

CprE 450/550X  
Distributed Systems and Middleware

## Introduction

Instructor: Yong Guan  
3216 Coover  
Tel: (515) 294-8378  
Email: [guan@ee.iastate.edu](mailto:guan@ee.iastate.edu)

Jan. 16, 2002

## Definition of a Distributed System

---

A distributed system is:

A collection of independent computers that  
appears to its users as a single coherent  
system.

## Questions?

---

- Why we need distributed systems?
- What properties should a distributed systems have?
- What problems/issues should be addressed?

## Why we need distributed systems?

---

- Centralized System
- Network Operating Systems
- Distributed Operating Systems
- Middleware
- Computer Networks

## Definition of a Distributed System

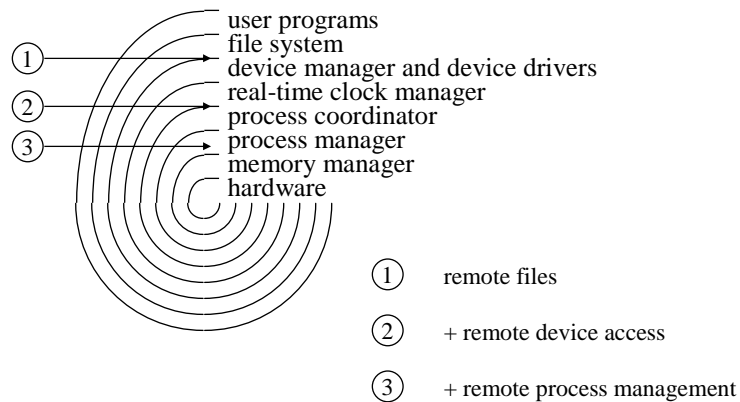
---

- Requirements:
  - Provide user with convenient virtual computer.
  - Hide distribution of resources.
  - Mechanisms for protecting resources.
  - Secure communication.
- Definition
  - **Distributed system looks to user like ordinary centralized OS, but runs on multiple, independent CPUs.**
    - Use of multiple processors is invisible.
    - User views system as virtual uniprocessor.

## The insider's view of a Centralized OS

---

- The insider's view of a centralized OS.
- (Roughly patterned after XINU [Comer 1984])



## Distributed vs. Centralized Systems

---

- Advantages of Distributed Systems:
  - Reliability.
  - Sharing of resources.
  - Aggregate computing power.
  - Openness/Scalability
- Disadvantages of distributed systems:
  - Security.
  - Physical distribution of resources vs. demand.
  - Computing power per node is limited.

## Distributed vs. Networked OS

---

### Transparency:

- How aware are users of the fact that multiple computers are being used?

- **Network OS:**
  - Users are aware where resources are located
  - Network OS is built on top of centralized OS.
  - Handles interfacing and coordination between local OSs.
- **Distributed OS:**
  - Designed to control and optimize operations and resources in distributed system.

# Network Operating Systems

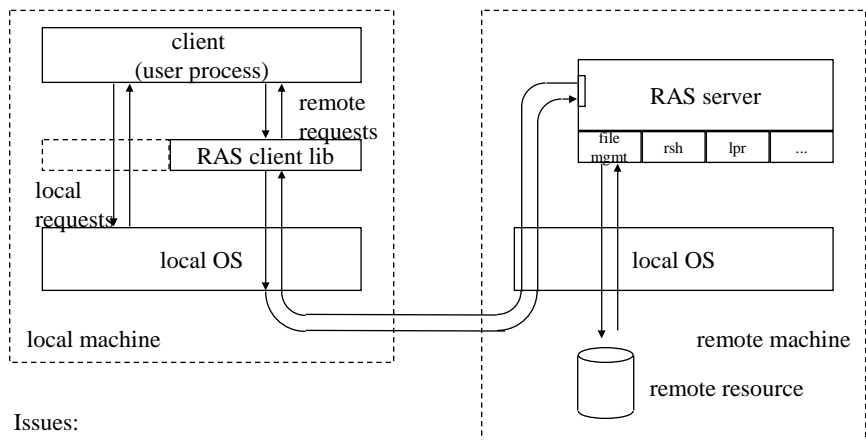
- Definition:

– A network OS is a collection of OSs of computers connected through a network incorporating modules to provide access to remote resources.

- Characteristics:

- Each computer has private OS.
- User works on his own machine and remotely logs in to other computers.
- Users are aware of location of files.
- Limited fault tolerance.

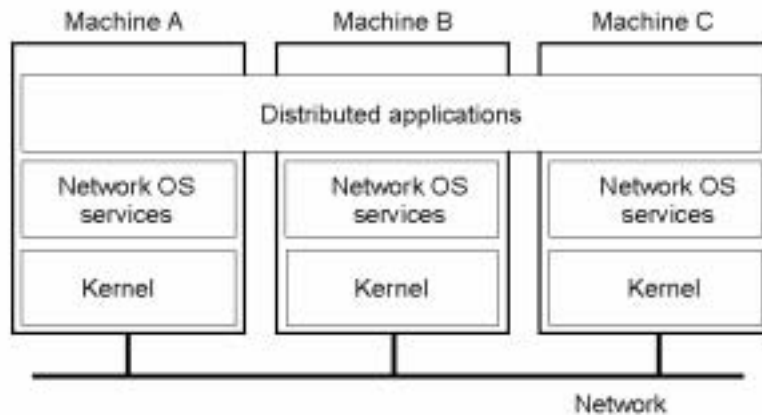
## A Vanilla Network OS (Remote Access System [Goscinsky '83])



Issues:

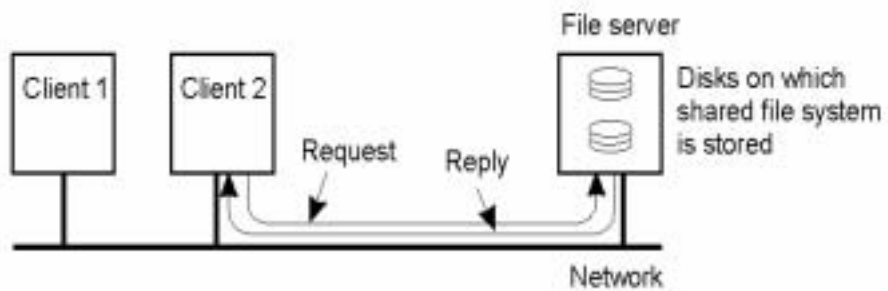
- Performance! (local and remote)
- Where is the state?
- Serialization of operations.
- Blocking operations

## Network Operating System (1)



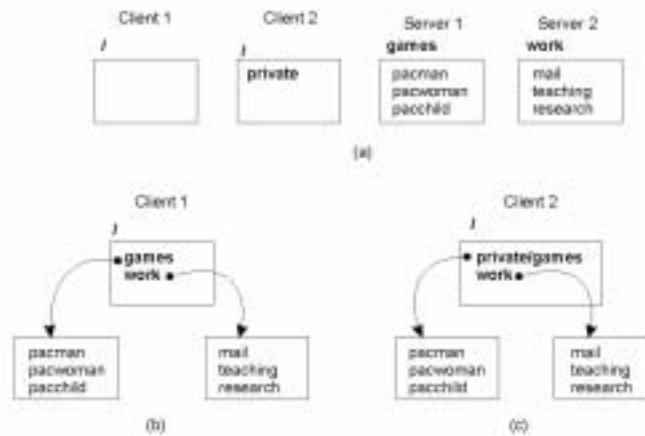
General structure of a network operating system.

## Network Operating System (2)



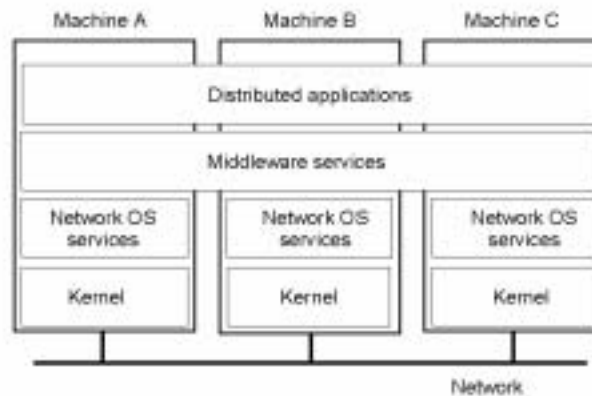
Two clients and a server in a network operating system.

## Network Operating System (3)



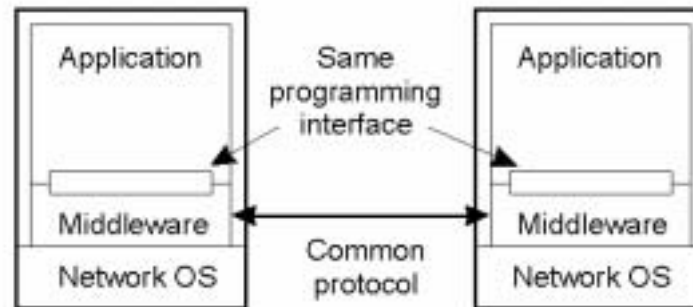
Different clients may mount the servers in different places.

## 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

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

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



## Research and Design Issues

---

- Communication model
- Paradigms for process interaction
- Transparency
- Heterogeneity
- Autonomy and/or interdependence
- Reliable distributed computing
- Replication

## Communication Model

---

- ISO/OSI Model
  - Physical
  - Datalink
  - Network
  - Transport
  - Session
  - Presentation
  - Application
- An alternative, e.g. *Functional*, Model
  - Physical
    - same as ISO/OSI
  - Datagram
    - connectionless service between source and destination process
    - location of services
  - Transport
    - reliable transport between client and server
    - “transaction level”
  - Binding
    - location of resources within the server
    - logical connection between client and server
  - User
    - request semantics

## Process Interaction: Client/Server

---

- **Server:** A subsystem that provides a particular type of service to a *priori* unknown clients.
- Control functionally distributed among the various servers in the system.
- Control of *individual* resources is centralized in a server. (localized?)
- Problems:
  - Reliability/Availability
  - Scalability
  - Replication?

## Process Interaction: Pipe Model

---

- **Pipe:** Communication facility to transfer data between processes in FIFO order. Can be used for synchronization purposes.
- Named/unnamed pipes
- Pipes for secure IPC
- Pipes across network?
- Multicast pipes?

## 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.

## Autonomy and Interdependence

---

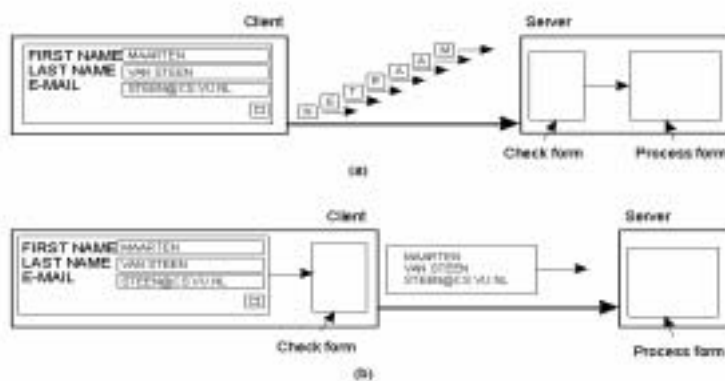
- Disadvantage generated by interdependence:
  - cannot work stand-alone
  - globally controlled
  - difficult to identify source of authority and responsibility
  - what about mutual suspicion?
- Reasons for autonomy:
  - policy freedom
  - robustness
  - cooperation between mutually suspicious users

## Scalability Problems

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

Examples of scalability limitations.

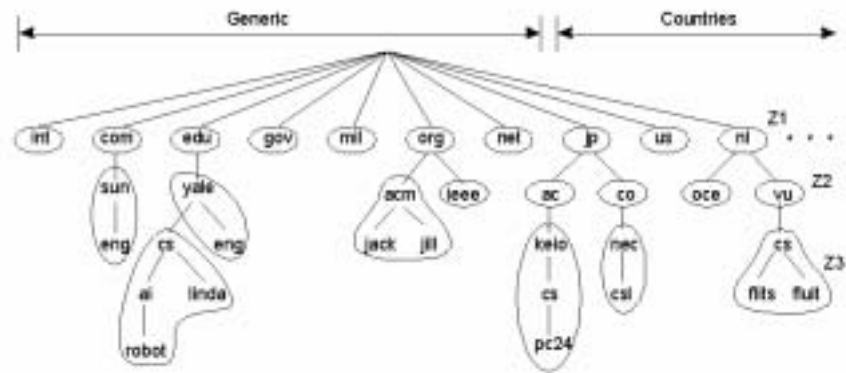
## Scaling Techniques (1)



The difference between letting:

- 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.

Any Questions?