# CS 5204
# Operating Systems
# Lecture 6

Godmar Back

Virginia Tech

## Outline

- Case for distributed systems, definition, evolution of OS
- Examples of distributed systems
- Goals for distributed systems
- Scaling techniques & design principles for distributed systems
- Slides from Tanenbaum: *Distributed Systems: Principles and Paradigms*



## Case for Distributed Systems

- Compare 80s and now with respect to
  - Growth in processing power
    - Grosch's Law (double price, 4x power: today maybe 50%)
  - Network speed/capacity: computing on bytes used to be cheaper than shipping bytes – now bandwidth is free
  - Changes in software design from big systems to small connected components
- Conclusion: → Distributed Systems!



## Definition

- Collection of independent computers that appears as a single coherent system. (Tanenbaum)
- Connects users and resources; allows users to share resources in a controlled way
- Lamport's alternative definition:
  - *You know you have one if the crash of a computer you've never heard of prevents you from getting any work done.*



## Examples of DS

- Client-Server systems
- Peer-to-Peer systems
  - Unstructured (e.g., exporting Windows shares)
  - Structured (e.g., Kazaa, Bittorrent, Chord)
- Clusters
- Middleware-based systems
- "True" distributed systems



## Aside: Clusters

- Term can have different meanings:

(n.) A group of computers connected by a high-speed network that work together as if they were one machine with multiple CPUs.
docs.sun.com/db/doc/805-4368/6j450e60c

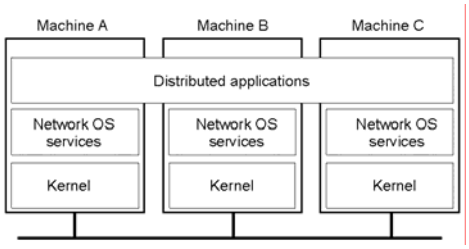
A group of independent computer systems known as nodes or hosts, that work together as a single system to ensure that mission-critical applications and resources remain available to clients.
www.microsoft.com/...serv/reskit/distsys/dsggloss.asp

## Dist'd Systems vs OS

- Centralized/Single-processor Operating System
  - manages local resources
- Network Operating Systems (NOS)
  - (loosely-coupled/heterogeneous, provides interoperability)
- Distributed Operating Systems (DOS)
  - (tightly-coupled, provides transparency)
    - Multiprocessor OS: homogeneous
    - Multicomputer OS: heterogeneous
- Middleware-based Distributed Systems
  - Heterogeneous + transparency



## Network OS

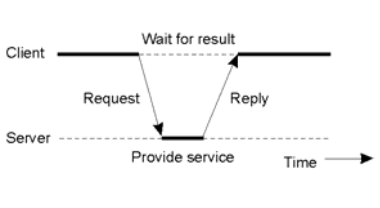



- Distributed system built on top of computers running a NOS: nodes export local services



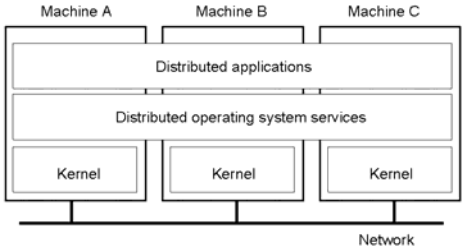
## Client/Server Paradigm

- Request/reply behavior: typical for NOS






## Distributed Operating System




- Can be multiprocessor or multicomputer

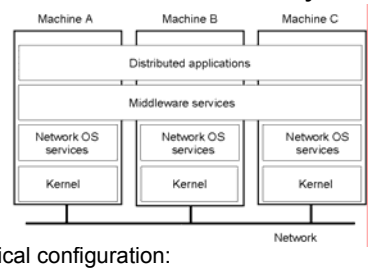


## "True" Distributed System

- Aka single system image
- OS manages heterogeneous collection of machines
  - Integrated process + resource management
  - Service migration
  - Automatic service replication
  - Concurrency control for resources built-in
- (Partial) examples: Apollo/Domain, V, Orca



## Middleware-based System



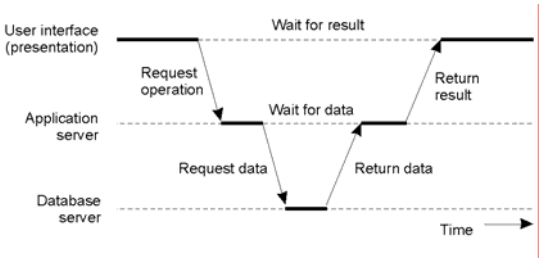

- Typical configuration:
  - Distributed system built on middleware layer running on top of NOS

## Middleware-based System (cont'd)

- Aka "three-tiered" architecture
  - Application/Middleware/OS
- Focus on
  - High-level communication facilities – paradigms
    - Distributed objects, distributed documents
  - Distribution transparency through naming
  - Persistence (file system/database)
  - Distributed transactions
  - Security
- Typically no management of local resources



## Three-tiered Architecture




- Observation: server may act as client



## Goals for Distributed Systems

- Transparency
- Consistency
- Robustness
- Scalability
- Openness
- Flexibility



## Kinds of Transparency

| 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 replicated |
| 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 |



## Degrees of Transparency

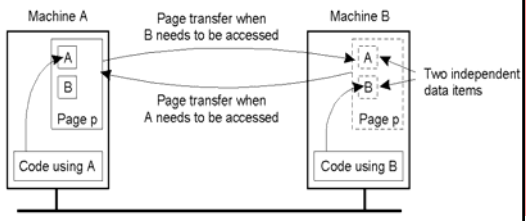
- Is full transparency always desired?
- Can't hide certain physical limitations, e.g. latency
- Trade-off w/ performance
  - Cost of consistency
  - Consider load imposed by attempts to provide full transparency



## Consistency

- Distributed synchronization
  - Time
    - Logical vs physical
  - Synchronization of resources
    - Mutual exclusion, election algorithms
- Distributed transactions
  - Achieve transaction semantics (ACID) in a distributed system
- Discussion:
  - Consistency in a distributed shared memory
  - Global address space: access to an address causes pages containing the address to be transferred to accessing computer

## False Sharing






## Robustness/Fault Tolerance

- In Client/server communication:
  - Retransmission of requests, RPC semantics
- Distributed protocols
  - Byzantine Generals Problem
- Software Fault Tolerance
- Resource control
  - E.g., DDoS



## Types of Scalability

- Size
  - Can add more users + resources
- Geography
  - Users & resources are geographically far apart
- Administration
  - System can span across multiple administrative domains
- Q.: What causes poor scalability?