Course Syllabus

- CMPSCI 677: Distributed Operating Systems
- **Instructor:** Prashant Shenoy
  - Email: shenoy@cs.umass.edu, Phone: (413) 577 0850
  - Office hours: Tuesday 12:30-1:30, CS 336, or by appt
- **Teaching Asst:** Nasreen Abdul Jaleel
  - Email: nasreen@cs.umass.edu, Phone: TBA
  - Office hours: TBA, CS 311
- **Course web page:** [http://lass.cs.umass.edu/~shenoy/courses/677](http://lass.cs.umass.edu/~shenoy/courses/677)
Course Outline

• Introduction *(today)*
  – What, why, why not?
  – Basics

• Interprocess Communication
  – RPCs, RMI, message- and stream-oriented communication

• Processes and their scheduling
  – Thread/process scheduling, code/process migration

• Naming and location management
  – Entities, addresses, access points

Course Outline

• Canonical problems and solutions
  – Mutual exclusion, leader election, clock synchronization, …

• Resource sharing, replication and consistency
  – DSM, DFS, consistency issues, caching and replication

• Fault-tolerance

• Security in distributed Systems

• Distributed middleware

• Advanced topics: web, multimedia, real-time and mobile systems
Misc. Course Details

• **Textbook:** Distributed Systems by Tannenbaum and Van Steen, Prentice Hall 2001

• **Grading**
  – 4-5 Homeworks (20%), 3-4 programming assignments (35%)
  – 1 mid-term and 1 final (40%), class participation (5%)

• **Course mailing list:** cs677@cs.umass.edu
  – You need to add yourself to this list! [ see class web page ]

• **Pre-requisites**
  – Undergrad course in operating systems
  – **Good** programming skills in a high-level prog. language

Definition of a Distributed System

• A distributed system:
  – Multiple connected CPUs working together
  – A collection of independent computers that appears to its users as a single coherent system

• Examples: parallel machines, networked machines
Advantages and Disadvantages

• Advantages
  – Communication and resource sharing possible
  – Economics – price-performance ratio
  – Reliability, scalability
  – Potential for incremental growth

• Disadvantages
  – Distribution-aware PLs, OSs and applications
  – Network connectivity essential
  – Security and privacy

Transparency in a Distributed System

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

Different forms of transparency in a distributed system.
Scalability Problems

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single on-line telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Doing routing based on complete information</td>
</tr>
</tbody>
</table>

Examples of scalability limitations.

Hardware Concepts: Multiprocessors (1)

- Multiprocessor dimensions
  - Memory: could be shared or be private to each CPU
  - Interconnect: could be shared (bus-based) or switched

- A bus-based multiprocessor.
Multiprocessors (2)

a) A crossbar switch     b) An omega switching network

Homogeneous Multicomputer Systems

a) Grid                                           b) Hypercube

(a)                                                     (b)
Distributed Systems Models

• Minicomputer model (e.g., early networks)
  – Each user has local machine
  – Local processing but can fetch remote data (files, databases)
• Workstation model (e.g., Sprite)
  – Processing can also migrate
• Client-server Model (e.g., V system, world wide web)
  – User has local workstation
  – Powerful workstations serve as servers (file, print, DB servers)
• Processor pool model (e.g., Amoeba, Plan 9)
  – Terminals are Xterms or diskless terminals
  – Pool of backend processors handle processing

Uniprocessor Operating Systems

• An OS acts as a resource manager or an arbitrator
  – Manages CPU, I/O devices, memory
• OS provides a virtual interface that is easier to use than hardware

• Structure of uniprocessor operating systems
  – Monolithic (e.g., MS-DOS, early UNIX)
    • One large kernel that handles everything
  – Layered design
    • Functionality is decomposed into N layers
    • Each layer uses services of layer N-1 and implements new service(s) for layer N+1
Uniprocessor Operating Systems

Microkernel architecture
- Small kernel
- user-level servers implement additional functionality

Distributed Operating System

- Manages resources in a distributed system
  - Seamlessly and transparently to the user
- Looks to the user like a centralized OS
  - But operates on multiple independent CPUs
- Provides transparency
  - Location, migration, concurrency, replication,…
- Presents users with a virtual uniprocessor
Types of Distributed OSs

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

Multiprocessor Operating Systems

- Like a uniprocessor operating system
- Manages multiple CPUs transparently to the user
- Each processor has its own hardware cache
  - Maintain consistency of cached data
Multicomputer Operating Systems

Network Operating System
Network Operating System

• Employs a client-server model
  – Minimal OS kernel
  – Additional functionality as user processes

Middleware-based Systems

• General
## Comparison between Systems

<table>
<thead>
<tr>
<th>Item</th>
<th>Distributed OS</th>
<th>Network OS</th>
<th>Middleware-based OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiproc.</td>
<td>Multicomp.</td>
<td></td>
</tr>
<tr>
<td>Degree of transparency</td>
<td>Very High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Same OS on all nodes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Number of copies of OS</td>
<td>1</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Basis for communication</td>
<td>Shared memory</td>
<td>Messages</td>
<td>Files</td>
</tr>
<tr>
<td>Resource management</td>
<td>Global, central</td>
<td>Global, distributed</td>
<td>Per node</td>
</tr>
<tr>
<td>Scalability</td>
<td>No</td>
<td>Moderately</td>
<td>Yes</td>
</tr>
<tr>
<td>Openness</td>
<td>Closed</td>
<td>Closed</td>
<td>Open</td>
</tr>
</tbody>
</table>