Introduction to Distributed Computing

Prof. Sanjeev Setia
Distributed Software Systems
CS 707

About this Class

- Distributed systems are ubiquitous
- Focus:
  - Fundamental concepts underlying distributed computing
  - designing and writing moderate-sized distributed applications
- Prerequisites:
  - CS 571 (Operating Systems)
  - CS 656 (Computer Networks)
  - CS 706 (Concurrent Software)
What you will learn

“I hear and I forget, I see and I remember, I do and I understand” – Chinese proverb

- Issues that arise in the development of distributed software
- Middleware technology
  - Threads, sockets
  - RPC, Java RMI/CORBA
  - Javaspaces (JINI), SOAP/Web Services/.NET, Enterprise Javabeans
    - Not discussed in class, but you can become more familiar with these technologies by

Logistics

- Grade: 60% projects, 40% exams
- Slides, assignments, reading material on class web page
  http://www.cs.gmu.edu/~setia/cs707/
- Two small (2-3 week) programming assignments to be done individually + one larger project that can be done in a group of two students
- Use any platform; all the necessary software will be available on IT&E lab computers
Schedule

- Introduction (today)
- Client-server application design
- Application-level protocols
  - Sockets
- Communication
  - RPC/RMI/CORBA
- Naming
- Synchronization
- Consistency & Replication
- Fault Tolerance

Distributed systems

- “Workgroups”
- ATM (bank) machines
- WWW
- Multimedia conferencing
- Computing landscape will soon consist of ubiquitous network-connected devices
  - “The network is the computer”
A typical portion of the Internet

A typical intranet
Portable and handheld devices in a distributed system

Distributed applications

- Applications that consist of a set of processes that are distributed across a network of machines and work together as an ensemble to solve a common problem
- In the past, mostly “client-server”
  - Resource management centralized at the server
  - “Peer to Peer” computing represents a movement towards more “truly” distributed applications
Web servers and web browsers

Goals/Benefits

- Resource sharing
- Scalability
- Fault tolerance and availability
- Performance
  - Parallel computing can be considered a subset of distributed computing
Challenges (Differences from Local Computing)

- Heterogeneity
- Latency
  - Interactions between distributed processes have a higher latency
- Memory Access
  - Remote memory access is not the same as local memory access
    - Local pointers are meaningless outside address space of process

Challenges cont’d

- Synchronization
  - Concurrent interactions the norm
- Partial failure
  - Applications need to adapt gracefully in the face of partial failure
  - Lamport once defined a distributed system as “One on which I cannot get any work done because some machine I have never heard of has crashed”
Communication Patterns

- Client-server
- Group-oriented/Peer-to-Peer
  - Applications that require reliability, scalability
- Function-shipping/Mobile Code/Agents
  - Postscript, Java

Clients invoke individual servers

![Diagram of client-server communication]

Key:
- Process: 
- Computer: 

Distributed Software Systems
A service provided by multiple servers

Web proxy server
A distributed application based on peer processes

Web applets

a) client request results in the downloading of applet code

b) client interacts with the applet
Thin clients and compute servers

Distributed Software Systems

Spontaneous networking in a hotel

Distributed Software Systems
Distributed Software: Goals

- Middleware handles heterogeneity
- Higher-level support
  - Make distributed nature of application **transparent** to the user/programmer
  - Remote Procedure Calls
  - RPC + Object orientation = CORBA
- Higher-level support BUT **expose** remote objects, partial failure, etc. to the programmer
  - JINI, Javaspaces
- Scalability

Software and hardware service layers in distributed systems

- Applications, services
- Middleware
- Operating system
- Computer and network hardware
Transparency in Distributed Systems

Access transparency: enables local and remote resources to be accessed using identical operations.
Location transparency: enables resources to be accessed without knowledge of their location.
Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them.
Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.
Performance transparency: allows the system to be reconfigured to improve performance as loads vary.
Performance transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.

Example: NFS

- A very successful distributed “application” based on RPC
  - Illustrates both arguments
- Interface for remote files same as interface for local files
- Soft mounts vs Hard mounts
  - Soft mounts expose network or server failures
  - Hard mounts force application to hang until server recovers
NFS cont’d

“Limitations on robustness and reliability of NFS have nothing to do with the implementation ... The problem can be traced to the interface upon which NFS is built, an interface that was designed for non-distributed computing where partial failure was not possible” – Waldo et al

Scalability

- Becoming increasingly important because of the changing computing landscape
- Key to scalability: decentralized algorithms and data structures
  - No machine has complete information about the state of the system
  - Machines make decisions based on locally available information
  - Failure of one machine does not ruin the algorithm
  - There is no implicit assumption that a global clock exists
# Computers in the Internet

<table>
<thead>
<tr>
<th>Date</th>
<th>Computers</th>
<th>Web servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979, Dec.</td>
<td>188</td>
<td>0</td>
</tr>
<tr>
<td>1989, July</td>
<td>130,000</td>
<td>0</td>
</tr>
<tr>
<td>1999, July</td>
<td>56,218,000</td>
<td>5,560,866</td>
</tr>
</tbody>
</table>

# Computers vs. Web servers in the Internet

<table>
<thead>
<tr>
<th>Date</th>
<th>Computers</th>
<th>Web servers</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993, July</td>
<td>1,776,000</td>
<td>130</td>
<td>0.008</td>
</tr>
<tr>
<td>1995, July</td>
<td>6,642,000</td>
<td>23,500</td>
<td>0.4</td>
</tr>
<tr>
<td>1997, July</td>
<td>19,540,000</td>
<td>1,203,095</td>
<td>6</td>
</tr>
<tr>
<td>1999, July</td>
<td>56,218,000</td>
<td>6,598,697</td>
<td>12</td>
</tr>
</tbody>
</table>
The difference between letting: (a) a server or (b) a client check forms as they are being filled.

An example of dividing the DNS name space into zones.
Readings

- Chapter 1 of textbook (Tannenbaum)
- “A Note on Distributed Computing” – Waldo, Wyant, Wollrath, Kendall
  - Link on class web page