Introduction to Distributed Computing

Prof. Sanjeev Setia
Distributed Software Systems
CS 707
Spring 2000

About this Class

- Distributed systems are ubiquitous
- Focus: designing and writing moderate-sized distributed applications
- Prerequisites:
  - CS 571 (Operating Systems)
  - 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
  - CORBA
  - Javaspaces (JINI), XML, Javabeans
    - Depending on time available

Logistics

- Grade: 70% projects, 30% exams
- Slides, assignments, reading material on class web page
  http://www.cs.gmu.edu/~setia/cs707/
- 3 or 4 small (2 week) programming assignments +1 larger project all to be done individually
- Use any platform; all the necessary software will be available on IT&E lab computers

Schedule

- Client-server application design
- Sockets; Application-level protocols
- RPC
- CORBA
- Other middleware technologies
- Designing Reliable and Scalable distributed applications – case studies, research articles

Distributed systems

- "Workgroups"
- ATM (bank) machines
- WWW
- Multimedia conferencing
- Computing landscape will soon consist of ubiquitous network-connected devices
  - "The network is the computer"

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
- But the next few years may see a movement towards more "truly" distributed applications

Benefits

- Performance
  - Parallel computing a subset of distributed computing
- Scalability
- Resource sharing
- Fault tolerance and availability
- Elegance

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
  - Applications that require reliability
- Function-shipping
  - Postscript, Java

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

Transparency

- Access – local and remote objects are accessed using identical operations
- Location – no knowledge of location of resource
- Concurrency – several processes can operate concurrently on shared objects without interference
- Replication – no knowledge of replicas
- Failure – graceful degradation
- Parallelism – tasks automatically parallelized

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

Readings

- Chapters in Tannenbaum's "Modern Operating Systems" or "Distributed Operating Systems"
- "A Note on Distributed Computing" – Waldo, Wyant, Wollrath, Kendall
  - Link on class web page