Further Experience with Teaching Distributed Systems to Undergraduates

Gregory M. Kapfhammer
Department of Computer Science
Allegheny College

http://cs.allegheny.edu/~gkapfham/
The First Day of Class

- Are you ready? You must become comfortable with the theory and practice of the design, implementation, and analysis of distributed systems and algorithms.

- I’ll never go back to a single address space again: students realize that building distributed systems is challenging and fun!

- Ricart and Agrawala who? distributed mutual what? why is Lamport stamping time? epidemic algorithms?

- I think I just had a page fault: students must manage a variety of complexity sources.
Distributed System Challenges

- Students need to be familiar with (or willing to learn!) material about programming languages, operating systems, networks, theory, and algorithms
- Nuts and bolts: RPCs in C, RMIs in Java, Java virtual machines, Jini 1.2 and 2.0, JavaSpaces, and the CLASSPATH
- Include concepts from concurrency such as semaphores and monitors with real implementations in Java
- What are we measuring? students must be able to conduct experiments to evaluate the relative strengths and weaknesses of algorithm and implementation choices
Course Objectives

- Explore the *principles* and *paradigms* associated with distributed systems
- Principles: communication, naming, distributed scheduling, synchronization, mutual exclusion, consistency, replication, and fault tolerance
- Paradigms: become very familiar with object-based distributed systems using Jini and JavaSpaces
- Include a discussion of special topics such as distributed hash tables (DHTs), tuple spaces, and data stream management systems (DSMS)
Instructional Objectives

- Course difficulty should be worn as a badge of honor
- Create exciting laboratory assignments where the students actually implement interesting systems and measure their performance
- A “gloves off approach” with a safety net: try to hide some complexity while ensuring that students retain perspective and understand many low-level details
- Require students to keep laboratory notebooks where they record hypotheses, data, observations, and design choices
- Read many scholarly and a few popular presss articles
Topic: Remote Communication

- Discovery, code downloading, parameters, transparency
- Local, remote, and distributed objects
Topic: Distributed Scheduling

- Unshared state, resource matching, and process migration
- Case studies: Sprite, Condor, Frugal, ComputeFarm for scientific computation and distributed testing
- Load balancing vs. load sharing and the cost/benefit tradeoff
Space clients can write, take, and read Entry objects.

How do we measure performance and/or correctness?
Special Topic: OpenDHT

- Clients can put and get with Sun RPC or XML-RPC
- How do we measure performance and/or correctness?
Special Topic: Data Streams

- Continuous streams of data and queries
- Case studies: STREAM, TelegraphCQ, load shedding
Conclusions

- The design and implementation of distributed systems is challenging, fun, and rewarding

- Current distributed systems textbooks are generally very good (there are still surprises, though!)

- There is a wealth of well-written literature from good conferences and journals

- Most students enjoy laboratories with Jini, JavaSpaces and Python (OpenDHT) while RPC programming in C is seen as tedious
Your comments, suggestions, and participation are invited!

http://cs.allegheny.edu/~gkapfham/teach/cs441/