Course Overview
CPS 212: Distributed Information Systems
Jeff Chase
Fall 1998

CPS 212 is a new graduate-level course on techniques for storing and sharing information in computer networks, large and small. We will cover a range of core distributed systems topics, with an emphasis on emerging technologies for Internet-based information services.

Who should take this course. CPS 212 is intended primarily for second-year graduate students who have already taken our courses in operating systems (CPS 210) and computer architecture (CPS 220). Although CPS 212 is a graduate course, the course is similar to CPS 214 (networks) in that advanced undergraduates are encouraged to participate, since we do not yet have a 100-level distributed systems course. If you are a motivated undergraduate and you did well in CPS 110, you will be safe in CPS 212. Whether you are a graduate student or an undergraduate, you will get more out of this course if you have taken CPS 214. You should also be confident of your ability to learn to use new programming tools (e.g., Java) quickly and independently. Please see me if you have doubts about whether or not you should take this course.

Readings. We will use the textbook Distributed Systems: Concepts and Design, along with a collection of research papers. Although the text is horribly dated in this fast-moving field, it covers the core material pretty well. The research readings are TBD but will be announced on the course web site early in the semester. Plan on reading 8-10 papers and writing a page of opinions about one or two of the systems studied. Several of the systems studied (e.g., Bayou) may consume a week or more of class time each. We will also draw from the book Java Distributed Computing by Jim Farley.

Workload and grading. Exams (probably a midterm and a final) will be worth 35% of your grade. You will also do some Java-based programming exercises (30%) and a semester project (20%). The project could be a continuation of the programming exercises, a significant study of some interesting technology from a list that I provide, or an implementation project of your choosing subject to my approval. You are encouraged to form teams for lab exercises and implementation projects.

The last 15% of your grade will be based on your reading report(s), an “exit interview”, and my personal discretion. I will be continuing my policy of giving an automatic A grade to any undergraduate who scores higher than all of the graduate students on the last exam.
Course material. The topics covered this semester will be similar to my previous seminar offerings of CPS 216 (database methodology). CPS 212 does have something to do with databases: we will spend a lot of time on transactional concurrency control, transactional update and recovery, logging and buffering, replication, and other topics important for commercial databases. However, we will spend little or no time on other topics important for databases, e.g., how to represent real-world information (the relational model) and how to organize data for indexed retrieval or query processing. These are topics for a “proper” offering of CPS 216.

Here is a draft outline of the topics to be covered. This outline will evolve (probably by shrinking) as the semester progresses.

1. Course introduction. Course overview and mechanics; Issues of trust, autonomy, and scale for three points along the continuum of distributed systems: scalable multiprocessors, switched gigabit LANs, and the global Internet. Basics: RPC and client/server, stream communication (TCP and HTTP), transportable code, and the importance of threads.

2. Clustering and Scalable Services. Motivation for clusters, high-speed messaging, distributed storage, building scalable Internet servers with clustering.


4. Distributed Objects. Object models for distributed computing. Naming, location, and invocation. Java Remote Method Invocation (RMI), Microsoft Component Object Model (COM), and CORBA.

5. Distributed Updates and Consistency. Logical time, the ordering of events, group communication, and the meaning of consistency. Eventual consistency and its importance for unreliable networks, mobile computing, and massively replicated services. Stronger models of consistency used in distributed shared memory and file caches.


7. Private Communication on Public Networks. Encryption using public and private keys, DES and RSA, authentication, key distribution, certificates and digital signatures, electronic commerce, and Java security.