Welcome!

- We’ll start by walking through the syllabus.
Logistics

- Meeting time: Tues/Thurs, 1200-1320
- Course web page:
  - [http://www.cs.uoregon.edu/classes/09F/cis630/](http://www.cs.uoregon.edu/classes/09F/cis630/)

- Office hours:
  - Rm. 203 Deschutes
  - Monday/Wednesday, 3:00-4:30pm
    - Or by appointment – just e-mail me or drop by.
  - My e-mail: matt@cs.uoregon.edu
You will want to read the chapters well. We will follow the text carefully.

Many figures used in slides are provided by book authors to be consistent with text.
Lectures

- Slides posted weekly.
- Interactive discussion important. Participation helps your grade if you’re “on the borderline.” Speak up, ask questions, participate.

- In my experience, more interactive classes are more interesting. Unidirectional lectures are dull.

- Some material will be pulled from sources other than the textbook.
  - I will post PDFs for papers, links for web pages, and put relevant books on reserve at the library.
Grade breakdown

- Problem sets (5%)
  - Not graded.
  - Third and seventh week.
- Programming exercise(s) (10%)
  - Java RMI Application
  - Hands-on to get a feel for building a basic D.S.
- Reading summaries (10%)
- Term exam (in class) (25%)
- Term paper (25%)
- Term project (25%)
Reading Assignments / Summary

- You will be given four papers on topics in distributed systems.
  - Weekly, starting third week.
  - These will not be identical to the 2008 papers – I try to update readings to stay current.

- Turn in two-page summaries.
- The goal is to give practice reading research papers.
  - This will be useful in getting ready to write the term paper.
- We’ll have discussion time in class when summaries are due to talk about anything you found interesting.
Term Exam

- One and only one exam in the course.
- Will take place before Thanksgiving.
- All content up to exam fair game.
  - Lectures
  - Assigned reading in the book
  - Papers that have been assigned

- This is not intended to be terribly painful. If you do the assignments and the readings, you’ll be on track.
Term Paper

- Give you an opportunity to explore some topic of interest.
  - Topics outside the scope of lectures/assignments are OK.

- Provide an experience digging through research literature to learn about a topic.

- Your paper will be presented in class during dead week (week 10).

- See course web page for paper requirements.
  - These will be posted soon.
Term Project

- Real hands on experience with distributed systems.
- Performed in teams.
  - Skills survey helps me balance teams.
  - Individual contributions **must** be documented in deliverable.
    - This is to prevent individuals who do not contribute from unfairly benefiting from the efforts of their teammates.

- Deliverables
  - Written report of accomplishments.
  - Demonstration to me during finals week.
Term Project (2)

- You will be able to use any of the usual CS department resources. These include:
  - Your office workstations
  - Workstations in rm. 100 (don’t be disruptive to users if you’re running in the background though…)
- Languages and technologies are your choice.
Experience survey.

- Please fill out the experience survey.
- This will be used for me to gauge the level of experience in the class for programming and project assignments.

- Be sure to put down your preferred email address!
  - For mailing list so I can distribute important information outside class periods.
Scheduled down-time

- We will have one class that will be cancelled in mid-November.
  - 11/17/2009
  - http://sc09.supercomputing.org/
    - Somewhat related to the course actually.

- I have to be at a conference that day up in Portland.
  - I would encourage people to try to attend one day if you have time.
  - If anyone here is interested in trying to attend, let’s get names down now. I can check with the dept. about covering student registrations.
    - Please indicate interest on survey sheet.
Changes

- Changes from previous years
  - Cutting out distributed shared memory topic.
  - Increasing amount of discussion on P2P systems.
  - Eliminating project presentations.

- These are tweaks to:
  - Respond to previous years feedback on workload.
    - Other tweaks may be made as the term evolves w.r.t. workload.
  - Keep the class up to date with current technology trends.
Wrap up syllabus topics

- Late policies
- Sick policy
  - Take away point on this one: **sick days are OK.**
    - Getting your peers (or instructor) sick if you know you are sick is not cool.

- Academic integrity
  - By grad school, you should know the drill. Don’t cheat.

- **Questions?**
A bit about your instructor

- BS: Mathematics/Computer Science
- MS: Computer Science
- PhD: Electrical and Computer Engineering (2006)

- BS/MS here at UO; PhD at Univ. of New Mexico

- PhD area: Analysis of external interference on the performance of very large scale parallel programs.

- Prior to coming to UO, I was a staff scientist at the Los Alamos National Laboratory for ~6.5 years.
- Member of DOE Advanced Computing Laboratory and LANL Continuum Dynamics group.
This week

- We’ll start with the fundamentals.
- Reading assignment this week: chapters 1, 2, and 3.
  - Characterization of distributed systems
  - System models
  - Networking and internetworking
Objectives

- Basics of distributed systems with examples.
- Challenges: What issues arise that make distributed systems an interesting topic?
- Common distributed system architecture models.
- Requirements: What requirements drive the design of distributed systems?
- Fundamental models: What submodels describe the fundamental properties of distributed systems?
- Refresh our understanding of the networking infrastructure upon which distributed systems are built.
What is a distributed system?

- A distributed system is one in which hardware and/or software components located at networked computers communicate and coordinate their actions by exchanging messages.

- What are the consequences of this?
  - Concurrency
  - No global clock
  - Independent failures
Consequences

- **Concurrency**
  - Operations can execute at the same time at each computer. Coordination of these concurrently executing activities is a core topic of this class.

- **No global clock**
  - Programs communicate by passing messages and occasionally close coordination requires a shared idea of what time things occur. How to deal with time is a fundamental part of distributed systems.

- **Independent failures**
  - Computers in the system will fail for a variety of reasons. How to deal with this is critical in real distributed systems.
Motivation to build dist. systems

The key motivation behind distributed systems is resource sharing, where resource is a generic term.

Examples: CPU time, storage space, bandwidth, software, services, etc…

Various reasons are behind this sharing.

- Redundancy and robustness
- Performance
- Scalability
- Economic factors
Examples

- The internet
- Intranets
- Mobile and ubiquitous computing
Intranets
Mobile and Ubiquitous computing
Terminology

- **Service**: A distinct part of a computer system that manages a collection of related resources and presents their functionality to users and applications.
  - E.g.: File service, print service.

- **Server**: A running program on a networked computer that accepts requests for services and returns the result of the service (if any) to the client. The client is the computer that submits the request to the server.
Challenges

- **Heterogeneity**
  - More than one type of hardware or software.

- **Openness**
  - Extensibility through defined standards and APIs.

- **Security**

- **Scalability**

- **Failure handling**
  - Gracefully deal with components failing.

- **Concurrency**

- **Transparency**
Transparency

- Transparency is a consistent theme in this course.

- From a software engineering perspective, it might be considered one of the fundamental topics of distributed systems.

- So what is transparency?
Transparencies

- **Access transparency**
  - Local and remote resources accessed with identical operations.

- **Location transparency**
  - Resources accessible without knowledge of physical (room number) or network location (IP address).

- **Concurrency transparency**
  - Multiple processes can operate on resources without interfering with each other.

- **Replication transparency**
  - Multiple instances of a resource can be accessed for performance or reliability reasons without knowledge of the replicas by users or application programmers.
Transparencies (2)

- **Failure transparency**
  - Hide faults and allow users and application programs to complete their tasks despite hardware/software failures.

- **Mobility transparency**
  - Allow movement of resources and clients in a system without affecting the operation of users and programs.

- **Performance transparency**
  - Allows the system to be reconfigured for performance reasons as loads vary.

- **Scaling transparency**
  - Allows the system and applications to expand in scale without change to the system structure or application algorithms.
Transparency

- Transparency is critical to building real systems.

- Transparency means designing a system so that one “sees through”, or doesn’t notice some property of it.

- In practice, it boils down to making inconvenient or difficult features of the system invisible to the user or application.

- When we go distributed, a world of issues arises that we wish to make transparent. Hence, this course.