

Wright State University

CORE Scholar

Computer Science & Engineering Syllabi

College of Engineering & Computer Science

Fall 2009

CEG 730-01: Distributed Computing Principles

Prabhaker Mateti

Wright State University - Main Campus, prabhaker.mateti@wright.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/cecs_syllabi



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Repository Citation

Mateti, P. (2009). CEG 730-01: Distributed Computing Principles. .

https://corescholar.libraries.wright.edu/cecs_syllabi/921

This Syllabus is brought to you for free and open access by the College of Engineering & Computer Science at CORE Scholar. It has been accepted for inclusion in Computer Science & Engineering Syllabi by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.



CEG 730: Distributed Computing Principles

Prabhaker Mateti

Catalog Description: [4 Credit Hours] Communicating sequential processes, clients and servers, remote procedure calls, stub generation, weak and strong semaphores, split-binary-semaphores, and distributed termination. Example languages: SR, Linda. **Prerequisite:** CEG 634, 633

Home Page <http://www.cs.wright.edu/~pmateti/Courses/730/> Bookmark this link and visit it periodically. All course related announcements will be posted on this page.

Source Materials

- Text Book: Gregory R. Andrews, *Concurrent Programming: Principles and Practice*, Benjamin/Cummings, 1991. The book is now in its fifth printing. Errata sheets are available for the first <http://www.cs.arizona.edu/sr/cperr1.pdf> and second <http://www.cs.arizona.edu/sr/cperr2.pdf> printings.
- A book of your choice on client/server programming in Java, and
- Other references listed below.

Course Content

This course is about the foundations and principles involved in distributed systems with an introduction to recent systems and languages. The next course (CEG 830) concentrates on systems related issues, and on the design of distributed systems and applications.

I lecture on only the conceptually difficult sections of Andrews' book, leaving the rest as reading assignments.

The following is tentative. The order of topics will be roughly as indicated, except for project related discussions. The numbers in parentheses at the end of each topic is an estimate of the number of (75-minute) lectures I intend to devote.

Fundamentals (4)

Programming logic. Pre-, post conditions, and loop invariants. Starvation, deadlocks, livelocks. Interference, fairness, safety and liveness properties. Fine-grain and coarse-grain parallelism. Review of classic problems in concurrency. Review of semaphores, and conditional critical regions. Review of monitors. Proper definition of semaphore. Weak and strong semaphores. Split-binary semaphores. Starvation-free mutual exclusion. The technique of "passing the baton."

1. Andrews, Chapters 1, 2, 3.1, 3.5, 4, 5, 6, and Section 10.2.
2. Jan Tijmen Udding, "Absence of Individual Starvation Using Weak Semaphores," *Information Processing Letters*, Vol. 23, 1986, 159-162.
3. Allen B. Downey, "The Little Book of Semaphores", freely downloadable book, 2008, <http://www.greenteapress.com/semaphores/>

Remote Invocation (4)

Remote Procedure Calls (RPC), and Clients and Servers. Implementation of RPC. Stub generation, marshaling arbitrary data structures. Sun XDR and RPCGEN. Study of an RPC example. Remote Method Invocation (RMI) and Object Serialization of Java.

1. Programming with ONC RPC, a tutorial by Digital (Compaq), available on the web (use a search engine to find links).
2. Java tutorial, <http://java.sun.com/docs/books/tutorial/>

Distributed Algorithms (4)

CSP, communicating sequential processes. Asynchronous and synchronous message passing. Logical clocks, ordering of events. Heartbeat, probe/echo, and broadcast algorithms. Distributed mutual exclusion, Distributed implementation of semaphores, and distributed termination detection.

1. C. A. R. Hoare, "Communicating Sequential Processes," *Communications of the ACM*, 1978, Vol. 21, No. 8, 666-677. <http://www.usinccsp.com/> has an entire book by Hoare.
2. Andrews, Chapters 7 and 8.

Languages and Systems for Distributed Programming (7)

SR: Multiple primitives, *Linda*: Distributed data structures, and Java, JavaSpaces, and Hadoop.

1. Andrews, Chapter 10.
2. Greg Andrews and Gregg Townsend, The MPD Programming Language, www.cs.arizona.edu/mpd/
3. Sudhir Ahuja, Nicholas Carriero and David Gelemter, "Linda and Friends," *IEEE Computer* (magazine), Vol. 19, No. 8, 26-34. <http://www.lindaspaces.com/> has an entire book.
4. JavaSpaces™ http://en.wikipedia.org/wiki/Tuple_space
5. Hadoop, <http://hadoop.apache.org/core/docs/current/>

Grading

Class and Newsgroup Participation 5%

Concurrent algorithms and concepts are often subtle. Please participate in the class room discussions. Feel free to discuss openly, to ask what may appear to be dumb questions, and to catch any errors that I make. Be alert, study the backup material right away, and solve the exercises promptly.

No problems will be assigned specifically as "home work." However, you are expected to have studied problems that appeared in the old exams of this course, and the exercises of our textbook.

There is a local discussion group (see the link on the course home page) that you should read and participate regularly as a vehicle for discussing your solutions. I will read this group regularly, and note silently who is contributing what. I expect other students to be able to find flaws and fix them. Only when the most fundamental errors are being made will I break my silence and contribute a response.

Exams 30+40%

There will be two exams contributing 35% plus 35% to the final grade. The date for the midterm exam will be announced in class. The date for the final exam is as set by the Registrar. Both exams are closed-book, and comprehensive. Many of the old exams for this course are available at the course home page.

Projects 15+10%

It is necessary that you are comfortable in the *use* of networked machines based on Unix and X11. It is also assumed that you are fluent in Java, C/C++ and Unix process-related system calls. It is not assumed that you are familiar with either RPC or RMI.

There are two programming projects using

1. C/C++ and RPC,
2. Java RMI and/or Jini, JavaSpaces

that implement a multi-workstation oriented *WhiteBoard* (WB) client-server program. In each of these, you will start from a working and well-documented program written by Mateti. Working Versions of this program written in C and RPC, and also in Java can be downloaded from the links given on the home page.

The projects are *required*. Not producing a demonstrable (even if faulty) project will award you a grade no better than an F in the whole course regardless of your performance in the exams. There is an Acceptance-Test

The projects must be work done by *you* as a *single individual*. The projects are evaluated both by a demo and by skimming the source code and associated documentation.

The due dates for the projects are announced on the course home page.

Project P0: Weight == 0%; Your first task is to run the C/X11/RPC version so that the server is on one node, and there are at least four clients on four different nodes and two different whiteboards. Turn in printouts of the X11 screen.

Project P1: Weight == 15%; Extend the P0 server as follows so that it now responds to messages from a new "client" that we would call `wbadmin`.

- Server must respond to a query with full information (name of the board, names of the host machines where the boards are in use, etc.) regarding all whiteboards.
- Server must respond to a request to initiate a new server on a given node. The newly given node may be the same node as that of an existing server. The newly created server begins its life with no boards to serve.
- A server must respond to a request to transfer all clients on a given whiteboard from one server to another. This transfer should be transparent to the clients.

Project P2: Weight == 10%; Re-design the P1 above but using Java, RMI, Jini and/or JavaSpaces. Whether we will use Jini + JavaSpaces or limit ourselves to Java RMI will depend on how our class is progressing.

Copyright © 2009 pmateti@wright.edu (937) 775 5114