Course Description: 6.852, Distributed Algorithms

1 People and places

Instructors: Prof. Nancy Lynch, 32-G668, MIT extension 3-7225, lynch@theory.csail.mit.edu; Available by appointment (Tuesday and Thursday afternoons are good.)
Dr. Victor Luchangco, Sun Microsystems, 32-G628, MIT extension 3-0309, victor.luchangco@sun.com. Available by appointment (Tuesday and Thursday afternoons are best).

Teaching assistant: Calvin Newport, 32-G670, MIT extension 3-1922, cnewport@mit.edu
Office hours: T4:00-6:00; 32-G6 Lounge

Course secretary: Joanne Talbot Hanley, 32-G672A, MIT extension 3-6054

Class meetings: Tuesdays and Thursdays, 11:00AM - 12:30PM in room 4-149

Web site and mailing list: We will set up a course mailing list. Please send email right away to Calvin, at cnewport@mit.edu, to join this list.

The course Web site is at URL http://courses.csail.mit.edu/6.852/08/ This site will contain downloadable copies of course handouts and related research papers, and other information about the course.

2 What this course is about

The field of distributed algorithms has become a well-developed research area over the past 25-30 years. Its results appear in specialized research conferences such as PODC (Principles Of Distributed Computing), DISC (International Symposium on DIStributed Computing), OPODIS (International Conference on Principles of Distributed Systems), and SPAA(ACM Symposium on Parallelism in Algorithms and Architectures), in general theoretical computer science conferences such as FOCS (Foundations of Computer Science) and STOC (Symposium on Theory of Computing), and in broad conferences involving distributed computing, such as ICDCS (International Conference on Distributed Computing Systems).

Distributed algorithms researchers define various kinds of distributed computing environments, such as shared-memory or network-based environments, and identify problems to be solved in those environments. They design new algorithms for these problems, and analyze the correctness, performance, and fault-tolerance of their algorithms. They also sometimes prove lower bounds and other impossibility results, which explain why certain tasks cannot be carried out in certain kinds of distributed settings, or cannot be carried out within certain cost constraints. Researchers typically study problems that arise in practical distributed systems, including problems of communication, data management, resource management, synchronization, and distributed agreement. Sometimes, the results have impact on practical distributed system design.

This year, the course will be co-taught by Prof. Lynch and Dr. Victor Luchangco, a member of the Scalable Systems research group at Sun Microsystems Research. The "core" of the material will be the same as usual—basic distributed algorithms and impossibility results. However, we are adding a few new things this year:

1. A unit on scalable shared-memory concurrent programming.

2. Some supplementary material on topics such as self-stabilization, wait-free computability, and failure detectors.

3. The Tempo toolset, for writing distributed algorithm pseudocode.

6.852 is a graduate-level course that is intended to do two things: provide an introduction to the most important basic results in the area of distributed algorithms, and prepare interested students to begin independent research or take a more advanced course in distributed algorithms. Usually, the students who take the course are a mixture of PhD students and MEng students. Since the course is run at a PhD level, most MEng students should find it challenging (and time-consuming).

3 Prerequisites

To take 6.852, you should have:

- "Mathematical maturity". In particular, you should be very good at reading and writing mathematical proofs.
- General knowledge about some distributed systems. For instance, MIT's undergraduate course 6.033, Computer Systems Engineering, would be good background.
- Experience with sequential algorithms and their analysis. MIT's undergrad course 6.046 is sufficient.
- (Desirable, but not essential) Experience with formal models of computation. MIT's course 6.045 on automata theory would be fine for this.

4 Source material

The primary source will be the book *Distributed Algorithms* by Nancy Lynch. See the course Web site for some information on finding and purchasing the book. Quantum Books has ordered copies for us. This book has gone through many printings, but we have made no changes since the fourth printing, so fourth or later are just fine. (In fact, the fourth printing contains only a few corrections over the third printing, so using a third printing copy would also be OK.) Known errata for early printings are collected in errata lists, which are accessible from the course Web page. The book refers to many papers from the research literature on distributed algorithms; you might want to track down and read some of these.

The secondary source will be the new book, *T*he Art of Multiprocessor Programming, by Profs. Maurice Herlihy and Nir Shavit (who work at Sun Research). This book is expected to be published (by Elsevier) mid-semester. However, the authors have, in the meantime, made a pdf of a book chapter available for download at `courses.csail.mit.edu/6.852/08/papers/lists-book-chapter.pdf`. Publisher info at `http://www.elsevier.com/wps/find/bookdescription.cws_home/714091/`.

Other books that you will find useful are:

- Hagit Attiya and Jennifer Welch *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. John Wiley and Sons, Inc., 2004. Second Edition.
  This is another textbook on distributed algorithms, initially published a little after the Lynch book. It now has a second edition. The material covered overlaps quite a lot with the Lynch book, though Attiya and Welch do cover some topics, like clock synchronization, that Lynch doesn't cover. The style is a little less formal.
- Shlomi Dolev. *Self-Stabilization*, MIT Press, 2000. This gives a good description of self-stabilizing distributed algorithms. Self-stabilization is a strong kind of fault-tolerance, which we will study near the end of the course.

Both books are on reserve in the CSAIL Reading Room, 32-G882 and at Barker Engineering Library, 10-500.

In addition, some research papers that are not covered in the textbook will be covered in class and on problem sets. They will cover a variety of topics, for example, computability and relative computability results for different kinds of objects and services, in the presence of various numbers of failures. These papers are listed in Handout 3. We will put instructions for obtaining these on the course Web site.

5 Course requirements

5.1 Readings

Readings that cover the material for each class will be announced ahead of time. For most classes, these will be sections from the textbook. For some classes, however, these will be research papers from the reading list, Handout 3. Reading a research paper will generally take more time than reading sections from the textbook—so be prepared to put in the extra time. We expect students to read the assigned material ahead of time, and to come to class prepared with questions and discussion ideas.

5.2 Problem sets

These are intended to help you to understand the material being covered in class. Most problems will involve thinking about algorithms and problems already covered in class; some will be designed to get you started thinking about ideas to be discussed in later classes.

Specifically, approximately five problems will be given out every Thursday. The problems will be batched and due every two weeks, at the beginning of class on alternate Thursdays. (Note exceptions to these rules on the course schedule, Handout 2.) There will be a total of seven problem set due dates. **No late homeworks will be accepted.** If you haven't finished, just hand in what you have completed. However, in case of a serious emergency, please talk to either Calvin Newport or one of the instructors.

Homework is an important part of your grade. When grading homework problems, we will try to give full credit to solutions that include all the important logical steps and ideas. We consider it a minus for a writeup to be lengthy and overly detailed. An exception is when we specifically ask for details, for instance, in formal proofs of correctness of algorithms.

Solutions to homework problems will be handed out. Whenever possible, the best student solution will be used. Students who would like us to use their writeups can help us by writing elegant and concise solutions and formatting them using LaTeX (and the LaTeX style files provided by us). When you submit the homework, keep the .tex file since it may need to be edited if your solution is chosen. Problem sets will be graded by teams of students in the class, led by Calvin and/or the instructors.

We have software available to assist you in writing syntactically-correct distributed algorithms; we are requiring that you use it for your homeworks. This software consists mainly of the Tempo language processor, which allows specification of algorithms as interacting state machines. Information about how to download the software and how to get started using it appears at www.veromodo.com.

You will notice that the Tempo language has a home-grown simulator, and connections to the PVS theorem-prover and UPPAAL model-checker. We are not requiring you to use any of these tools, but of course, you are welcome to try them. You will probably find the User Guide and test examples useful.

The Tempo language is based on Timed I/O Automata, a mathematical modeling framework consisting of interacting state machines, possibly with timing constraints. To learn about the mathematical framework, you should consult the monograph *The Theory of Timed I/O Automata*, by Kaynar, Lynch, Segala, and Vaandrager. This is part of the Synthesis series of computer science monographs, published by Morgan Claypool. This monograph (and others) can be dowloaded in pdf format for free from any machine with an IP address in the MIT range (MIT owns a license).
See http://www.morganclaypool.com/doi/abs/10.2200/S00006ED1V01Y200508CSL001.

Policy on homework collaboration: You are *strongly encouraged* to discuss possible solutions with other class members. Many students in past incarnations of this course have formed homework discussion groups. However, *you must always write up the solutions entirely on your own.*

5.3 Problem set grading

For each problem set, a group of 3-4 students will be responsible for working with the course staff to grade the solutions. If possible, we would like the grading to be completed by the Monday afternoon after the homeworks are handed in, so we can record the grades and hand them back on Tuesday. The number of times you have to grade over the course of the semester will depend on the size of the class. Part of your grade will depend on the quality and promptness of your work on problem set grading.

5.4 Exams

There will be no exams. No midterm, no final. You can go home after the last class, on May 15.

5.5 Course grade calculation

Your course grade will be based on problem set grades, problem set grading grades, and class participation. Here is how your grade will be calculated:

- Problem sets: 70% (10% for each problem set)
- Class participation (attendance, quality and quantity of participation): 20%
- Grading (quality and promptness): 10%