Course Description: 6.852, Distributed Algorithms

1 People and places

Instructor: Prof. Nancy Lynch, 32-G668, MIT extension 3-7225, lynch@csail.mit.edu. Available by appointment; Tuesday and Wednesday afternoons are generally good, and late Thursday afternoons are sometimes good.

Teaching assistant: Mira Radeva, 32-G630, MIT extension 3-5971, radeva@csail.mit.edu. Office hours: Monday, Wednesday 3 - 4:30PM.

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

Class meetings: Tuesdays and Thursdays, 11:00AM - 12:30PM in Room 3-370.

Web site and mailing list: Through Stellar, registered students will automatically be set up on a mailing list. However, if you are not registered and would like to be on the list, please email Mira at radeva@csail.mit.edu, to join.

Website: http://stellar.mit.edu/S/course/6/fa13/6.852/

In addition to all the usual course-related material, this site will contain downloadable copies of course handouts and related research papers.

2 What is this course about?

Distributed Algorithms are algorithms that are designed to run on many processors, without tight centralized control. In general, they are much harder to design, and much harder to understand, then single-processor sequential algorithms. Distributed algorithms are important for most practical systems, including large-scale and local-area computer networks, data management systems, and multiprocessor shared-memory systems. Distributed algorithms also have a rich theory, which forms the subject matter for this course.

Theoretical results about distributed algorithms appear in 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). They also appear 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).

What do distributed algorithms researchers do? They (we) define various kinds of distributed system models, and identify important problems to be solved using those models. They design new algorithms to solve the problems, and analyze their behavior—which means correctness, performance, and fault-tolerance. Researchers also 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. Often, the results have impact on practical distributed system design.
This year, the course will focus mainly on basic distributed algorithms and impossibility results, as covered in Lynch’s book *Distributed Algorithms*. This will be supplemented by some updated material on topics such as self-stabilization, wait-free computability, and failure detectors. At the end, we will touch on algorithms for dynamic networks.

6.852 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.

### 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 or 6.840 would be fine for this.

### 4 Source material

The primary source will be the book *Distributed Algorithms* by Nancy Lynch, which can be purchased at the MIT Coop. This book has gone through many printings, but we have made no changes since the fourth printing, so fourth and later are just fine. Known errata are collected in an errata list, which is 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 some of these and read them.

Other books that you should 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, originally published at around the same time as the Lynch book. But this one 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 bit 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.
- Dilsun Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager, *The Theory of Timed I/O Automata*, Second Edition. This monograph presents a basic modeling framework, Timed I/O Automata (TIOA), for describing and analyzing distributed algorithms, especially those that can use timing.

These books are on reserve at Barton; see [http://library.mit.edu/F/?func=find-a&find_code=WCN&request=6.852&local_base=U-MIT30](http://library.mit.edu/F/?func=find-a&find_code=WCN&request=6.852&local_base=U-MIT30) for information about obtaining them.

In addition, some research papers whose contents are not covered in the textbook will be covered in class and on problem sets. 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

We will announce readings that cover the material for each class before that class. Most of the readings will
be sections from the main textbook. Some will be sections of the other books and research papers listed in
Handout 3. We expect that you will read the assigned material ahead of time, and come to class prepared to
discuss it. Reading a research paper generally takes more time than reading sections from the textbook—so
you should plan for that.

5.2 Problem sets

These are intended to help you to understand the material covered in class. Most problems will be about
results already covered; some will be designed to get you started thinking about ideas that will arise later.
Specifically, we will assign approximately 4-5 problems every Thursday. The problems will be batched and
due every two weeks, at the beginning of class on alternate Thursdays. (See the course schedule, Handout
2, for exact dates.) There will be a total of six (two-part) problem sets, plus a short (one-part) seventh
problem set. No late homeworks will be accepted. If you haven’t finished, just hand in what you have
completed. In case of a serious emergency, please talk to either Mira Radeva or Nancy Lynch.

When grading homework problems, we will try to give full credit to solutions that include all the important
logical steps and ideas. Your writeups should not be overly long and detailed. (However, we will sometimes
ask specifically for details, for instance, when we are studying formal correctness proofs for algorithms.)
We will hand out solutions to homework problems, usually the best student solutions. If you would like us
to use your writeups, you can help us by writing elegant and concise solutions and formatting them using
LaTeX (and the LaTeX style files we will provide). When you submit your homework, keep the .tex file since
you (or we) may need to edit it if your solutions are chosen. Problem sets will be graded by teams of students
in the class, led by the course staff.

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 3-5 students will be responsible for working with the course staff to grade
the solutions. 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 will
have to grade over the course of the semester should be one or two, depending 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 Term projects

This year students are required to carry out a term project related to distributed algorithms. We will provide
a handout with more details about this in a couple of weeks, but here is a summary.

Students should work in teams of two or three. Several sorts of projects are acceptable, including:

- Reading project: Choose a topic of current interest in the distributed algorithms research community
  (we will make some suggestions). Read the most important papers on the topic and write an expository
  report explaining the key theoretical ideas.

- Theoretical research project: Find a research problem you are interested in, and devise your own new
  algorithms and/or prove your own new lower bound result. Write a theoretical research report about
it. For this, it’s better to consider current research topics, so you will need to do some background reading for this type of project as well.

- **Experimental research project**: You can simulate known distributed algorithms and perform experiments to determine how they behave under various assumptions. Write an experimental research report about it. This kind of project will need a clear question and clear conclusions.

A project proposal will be due around the middle of the term, which will give us a chance to approve the project and make suggestions. The final report will be due at the last regular class, on Tuesday, Dec. 10. Students will present summaries of their projects at the last regular class. If this is not enough time, we will continue at the same time on Thursday, Dec. 12. Attendance at this extra meeting is optional, and if you cannot attend it will not affect your grade. The main purpose of the presentations is to give you a chance to tell everyone what you worked on.

### 5.5 Exams

There will be no exams. No midterm, no final. However, we are not quite done after the last class—we will have one optional meeting soon after the last class to finish up any project presentations that we do not complete during the last regular class.

### 5.6 Course grade calculation

Your course grade will be based on problem set grades, term project grades, and grades for your class participation and problem set grading. Here is how we will calculate it:

- Problem sets: 60%
- Term projects: 25%
- Class participation and grading: 15%