Course Syllabus

Comp 735 – Distributed and Concurrent Algorithms

Spring 2019

Meeting Place: FB 007
Meeting Time: 2:00 – 3:15, TuTh
Course Web Page: http://www.cs.unc.edu/~anderson/teach/comp735

Instructor: Prof. Jim Anderson
Telephone: 590-6057
Office: FB 112
E-mail: anderson@cs.unc.edu
Office Hours: 11:00 – Noon, TuTh

Goals of the course: To present fundamental algorithms and impossibility results from the concurrent programming literature, and to cover techniques for formally specifying and verifying concurrent systems. Both message-passing and shared-memory models of concurrency will be considered. At the end of the course, students will have a general knowledge of the concurrent programming literature, and will be able to develop new concurrent algorithms and verify their correctness. Perhaps the most important skill to be developed is the ability to intuitively “see” how or why a concurrent program works (a skill most students probably take for granted when it comes to sequential programs). In other words, this class will teach you how to “think” concurrently.

Who should take this class: Anyone who is likely to engage in work or research involving concurrent or distributed systems.

Prerequisites: Comp 550 (undergrad algorithms) and Comp 530 (undergrad operating systems). If you haven’t had one or both prerequisites and would like to take the class anyway, please see me.

Grading: Take-Home Assignments 65%
Survey or Research Paper 30%
Class Participation 5%

Take-home assignments will be given approximately once every three weeks. These assignments are required to be individual efforts (in contrast to, say, Comp 750). Any violations of this policy will be considered cheating and will be referred to the Student Attorney General. Any questions regarding the assignments should be referred to the instructor. You should not discuss the assignments with each other at all. Any attempts to find solutions on the web, in the files of other students who have taken this or similar courses, etc., will be considered an honor code violation. Any attempts to post my solutions on the web will be considered an honor code violation. (Sorry if this sounds harsh—it’s better to be clear about these things up-front than to have misunderstandings later.)

Note: I do re-use old assignments. As noted above, any attempts to access old assignments from the files of students who have taken this course previously, or from other sources, will be considered an honor code violation.

Students will be required to investigate a research topic of their choosing in some depth, and to write a research paper or survey paper on that topic. Topics related to your research area are especially encouraged. Some example topics are given below. We will have a more in-depth discussion about the requirements for the paper later in the semester.

Class Etiquette: You are expected to maintain proper etiquette in class. This includes:
• not making a habit of arriving late, leaving in the midst of class, or skipping class,
• not talking, sleeping, reading newspapers, eating, etc. in class,
• keeping cellphones, pagers, etc. off in class,
• and not using your laptop to browse the web in class.

Class Participation: This class will be far more enjoyable for everyone if all students come to class ready and willing to discuss the material to be covered. I plan to reward those who participate in class by increasing their final grade by up to half a letter grade. I also reserve the right to add a similar negative “reward” for those who do not observe appropriate etiquette in class.

Text: The primary “text” will be research papers (see list below). We will also cover several chapters of the following books.


We cover just a few chapters from these books. You don’t need to buy them.

If you really would like to have a textbook that covers (most of) the material from this course, the following is probably your best bet. (I have always listed this book as a potential resource, but I don’t think any student has ever bought it, so I didn’t order copies. If you want a copy, I suggest you use Amazon or some other similar on-line service.)


Possible Paper Topics: Clock synchronization, transactions, distributed simulation, distributed shared-memory systems, leader election algorithms, synchronization algorithms for real-time systems, frameworks for specifying/verifying concurrent systems (especially frameworks that emphasize program composition), automated verification tools, distributed search, distributed graph algorithms, distributed algorithms in multimedia systems, randomized concurrent algorithms, routing algorithms, lower bound results, distributed hashing, state maintenance in dynamic distributed systems, issues in peer-to-peer systems, synchronization in multicore systems, security issues in distributed systems, distributed algorithms for cloud-based computing. Students may also cover some additional papers on some of the topics to be covered in class by the instructor. For a topic to be approved, it must have a significant formal or algorithmic component.

Probable List of Topics: The following is a probable list of topics to be covered, along with a list of papers for each topic. I am willing to make slight changes to this list based on the interests of the class. Papers that are background or supplemental reading are denoted BR and SR, respectively. These papers will not be covered in class, but have been included because they contain information that may prove useful. (You do not have to read these papers. They are included for completeness, and also because they may contain things like definitions that are useful in understanding the papers that will be directly covered.)

• Reasoning about distributed and concurrent programs (3 weeks).
  Topics: state assertions, temporal assertions, safety properties, partial correctness, invariants, liveness properties, total correctness, well-founded sets, fairness, proof rules, example correctness proofs.
  Papers:


- **Synchronization algorithms for shared-memory systems (4.5 weeks).**

  Topics: mutual exclusion, barrier synchronization, readers/writers, local-spin algorithms, wait-free and lock-free synchronization.

  **Papers — Mutual Exclusion with Read and Write Instructions:**


  **Papers — Mutual Exclusion with Read-modify-write Instructions:**


  **Papers — Barrier Synchronization Algorithms:**


  **Papers — Readers/Writers:**


Papers — Wait-free Synchronization using Read and Write Instructions:

**Note:** Most of the papers on this topic are rather long and quite formidable. I plan to merely hit the highlights when covering this material.


**Note:** When reading the next two papers, just read the introductory parts and informal algorithm descriptions. Skip the formal proofs.


Papers — Wait-free and Lock-free Synchronization using Stronger Primitives:


- **Synchronization algorithms for message-passing systems (1 week).**

  **Topics:** mutual exclusion, logical clocks, fundamentals of resource allocation (dining philosophers,
drinking philosophers).

Papers:


• Fault tolerance in message-passing systems (5 weeks).

Topics: Byzantine agreement (algorithms and impossibility results), distributed consensus (algorithms and impossibility results), atomic commit protocols (in particular, why they work in light of the impossibility results), synchrony (in particular, how “synchronous” does a system have to be to tolerate faults?), broadcast and multicast algorithms, active and passive replication, self-stabilizing systems.

Papers — Byzantine agreement and consensus:


Papers — Atomic commit protocols:


Papers — Broadcast protocols:

Chapter 5 (Fault-tolerant Broadcasts and Related Problems) of *Distributed Systems*.

Papers — Replication:

Chapter 7 (Replication Management using the State-machine Approach) of *Distributed Systems*.

Chapter 8 (The Primary-backup Approach) of *Distributed Systems*.

Papers — Self-stabilization:

• **Algorithms for detecting stable properties in message-passing systems (1 week).**

  Topics: deadlock detection, termination detection, diffusing computations, distributed snapshots.

  **Papers:**


