Course Syllabus
COMP 737 – Real-Time Systems
Fall 2015

Meeting Place: SN115
Meeting Time: 4:00–6:45, Tu (We are having two 75-minute meetings back-to-back with a short break in between on Tuesday evenings. I will adjust the schedule so that this translates into an appropriate number of meetings for the semester when considering holiday breaks.)
Course Web Page: http://www.cs.unc.edu/~anderson/teach/comp737. The powerpoint slides and other things can be found here.

Instructor: Prof. Jim Anderson
Telephone: 590-6057
Office: FB112
E-mail: anderson@cs.unc.edu
Office Hours: 2:00 - 3:00, MW

Grader: Kecheng Yang
Office: SN126
E-mail: yangk@cs.unc.edu
Office Hours: None. Kecheng is not a TA. He will be grading your homeworks. I'll be grading your exams and project. You should contact him only if you have a homework grading issue.

Goal of the Course: To study issues related to the design and analysis of systems with real-time constraints. The problem of ensuring such constraints is ultimately a scheduling problem, so much attention is devoted to such problems. This is a “must” course for anyone wanting to do real-time systems research in this department.


We will discuss in class how to get access to this text. We will also use some papers from the literature. These papers and some additional “classic” papers can be found on the course web page. The papers and the powerpoint slides will not be copied for you.

Prerequisites: COMP 530 (Undergrad Operating Systems). In addition, one of the things researchers in this area need to be comfortable with is computational-complexity issues pertaining to validating timing constraints. Because of this, for a class or two, some basic knowledge of NP-completeness, as covered in COMP 750 and some undergrad classes, will be useful. That said, a deep understanding of NP-completeness is not necessary. No NP-completeness-related questions will be asked on exams (though some easy ones may come up on homeworks). Please note that I do not intend to zealously enforce the prerequisites. Anyone with a decent background in algorithms and operating systems should be able to handle the material.

Grading:
Homework 20%
Project 30%
Midterm Exam 20%
Final Exam 30%

We will probably have five or six homework assignments. Some of these will involve programming. All homework assignments must be completed individually. These assignments are designed to make sure you’re keeping pace: you should not find them extraordinarily time-consuming. You must include a signed honor statement with each homework submission indicating that it is your own work.

Each student must complete a class project. You are responsible for defining your own project. Your project can be either an experimental investigation or a survey or research paper. The project must be a fairly significant piece of work. While survey papers are OK, I’d really like to see projects that are
more research-oriented. It is perfectly fine to use research from an RA position as the basis for your class project. However, your project may not be based on work from another course without the permission of me and the instructor for that course (permission will be granted only if the total work involved is commensurate with the amount of effort expected in both courses combined). Two-person projects may be permitted, provided the total work involved is about twice that of the typical single-person project.

The final exam will cover the entire course.

Note: I do re-use old homeworks and exam questions. Any attempts to access old homeworks and exams from the files of students who have taken this course previously, or from other sources, will be considered an honor code violation.

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 off,
- and not using your laptop to browse the web.

Note: Posting materials from this class online without prior authorization is an honor code violation.

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.

Topics: The list of topics I plan to cover is given below. (Chapter numbers refer to Liu’s book.) Note: We have 29 total classes (where a “class” is defined to be 75 minutes), one of which will be used for the midterm.

Part I: Uniprocessor Scheduling of Independent Tasks.

- **Introduction to real-time systems (1 week).**
  - Chapter 1: Example real-time applications.
  - Chapter 2: Hard vs. soft real time.
  - Chapter 3: Reference model (includes lots of definitions used in later chapters).

- **Classic uniprocessor scheduling results (4 weeks).**
  - Static scheduling.
    - Chapter 5: Cyclic executives.
  - Dynamic scheduling.
    - Dynamic-priority scheduling:
      - Chapter 4, Section 6: Optimality of EDF and LLF.
      - Chapter 6, Section 3: Utilization-based schedulability test for EDF.
Nonpreemptive EDF from:

* Static-priority scheduling:
  · Chapter 6, Section 4: Optimality of RM and DM.
  · Chapter 6, Section 7: Utilization-based schedulability test for RM. (Skip 6.7.3 – 6.7.5.)
  · Chapter 6, Sections 5 and 6: Demand-based scheduling conditions for static-priority systems.

* Dealing with complexities arising in real systems.
  · Chapter 6, Section 8: Practical considerations. (Skip 6.8.6 – 6.8.7.)
  · Timing analysis, from:

• Intractibility results (1.5 weeks).
  – Preemptive systems.
    * Dynamic-priority systems, from:
        · The last two papers are a little complicated: their results will be stated without proof.
    * Static-priority systems, from:
        · This paper is a little complicated: its results will be stated without proof.
  – Nonpreemptive systems.
    * Dynamic-priority systems, from:

* Static-priority systems (no good reference here).

Part II: Beyond Uniprocessor Independent Task Models.

• Resource sharing (1 week).
  – Chapter 8: Priority inheritance and priority ceiling protocols, stack resource protocol. (Skip 8.7 – 8.10.)
Lock-free approach (very briefly), from:

**Multiprocessor schedulability analysis (1 week).**

- General overview, mostly from:
- Hard real-time analysis for global EDF, from:
- Soft real-time analysis for global EDF, from:

**Synchronization in multiprocessors and distributed systems (1 week).**

- Multiprocessor locking protocols, from:
  B. Brandenburg and J. Anderson, “Optimality Results for Multiprocessor Real-Time Locking,”
- Chapter 9, Section 4: End-to-end scheduling.

**Mixing real-time and non-real-time (2.5 weeks).**

- Chapter 7, Section 1: Introduction.
- Chapter 7, Section 2: Deferrable servers.
- Chapter 7, Section 3: Sporadic servers.
- Chapter 7, Section 4: Constant utilization and total bandwidth servers. (We will skip weighted fair queuing, since we are covering proportional-share scheduling, which is similar.)

**Fairness (1 week).**

- Proportional-share scheduling, from:
- Pfair scheduling, from:

**A quick look at some real systems (or, how much of this theory really gets used anyway?) (1 week).**

- Chapter 12, Sections 1 and 2: Basic operating-system functions needed for real-time computing.
  - The above will be supplemented with some material from Chapter 2 of Björn Brandenburg’s Ph.D. dissertation: *Scheduling and Locking in Multiprocessor Real-Time Operating Systems*, The University of North Carolina at Chapel Hill, 2011.
- Chapter 12, Sections 6 and 7: A brief survey of commercial real-time and non-real-time operating systems.
* The above will also be supplemented with some material from Chapter 2 of Björn Brandenburg’s Ph.D. dissertation.

Part III: Beyond This Class. We have a weekly “real-time lunch” meeting where ongoing research in our real-time group is discussed. You are welcome to attend these meetings. Also, Sanjoy periodically teaches special seminar courses on some aspect of real-time systems. Watch for those.