EEL-4736/EEL-5737 Principles of Computer System Design
Syllabus – Fall 2019


Introduction: The design of hardware and software in computer systems ranging from personal devices to large-scale distributed, networked computers is an increasingly complex undertaking and requires understanding not only of individual sub-systems, such as the micro-processor, but also the interactions among sub-systems. This class provides a broad introduction to the main principles and abstractions for engineering computer systems, and in-depth studies of their use on computer systems across a variety of designs, be it an operating system, a client/server application, a database server, or a fault-tolerant disk cluster.

Prerequisites: Digital design (EEL4712 or equivalent); introduction to programming or data structures/algorithms (EEL4834 or equivalent). Programming in a high-level language.

Computer usage: Student personal computers will be used in assignments. Students will be expected to use the Linux operating system, either natively on their computers, or within a virtual machine (VM).

Assignments: Homeworks and a project will be assigned in this class. The project entails an exploration of a topic related to the design of a computer system through implementation of a prototype. The assignments and project will require significant software programming using the Python high-level language.

Exams: There will be several in-class quizzes, two midterms and one final exam in this class. An approximate breakdown of the grade weights is 35% for homework/project assignments, 65% for exams. Quiz grades will not be used in computing your final grades - quizzes are going to be used to help you self-assess your class knowledge, and practice for exams.

Course topics:

Overview of computer systems: sources of complexity and design principles (chapter 1)
  Week 1: Lecture/slide set 1: systems and complexity
  Modularity, Abstraction, Layering, Hierarchy
Elements of computer system organization (chapter 1)
  Week 1: Lecture/slide set 2: fundamental abstractions
  Memory, interpreters, communication links
Layering and naming in computer systems (chapter 2)
  Week 2: Lecture/slide set 3: naming introduction
  Week 3: Lecture/slide set 4: names and layers
  Week 3: Lecture/slide set 5: UNIX file system case study
Enforcing modularity (chapters 4 and 5)
  Week 4: Lecture/slide set 6: client/service modularity
  Week 5: Lecture/slide set 7: case study – network file system
  Week 5: Lecture/slide set 8: virtualization abstractions
  Week 6: Lecture/slide set 9: virtual links
  Week 6: Lecture/slide set 10: memory modularity
  Week 7: Lecture/slide set 11: virtual memory
  Week 7: Lecture/slide set 12: virtual processor threads
Designing for performance (chapter 6)
Week 8: Lecture/slide set 13: designing for performance
  Exploiting workload properties, concurrency; addressing bottlenecks

Week 9: Lecture/slide set 14: scheduling

The network as a system and as a system component (chapter 7)
Week 10: Lecture/slide set 15: network properties
Week 10: Lecture/slide set 16: network layers
Week 11: Lecture/slide set 17: network case studies: ARP, IP, Ethernet

Fault tolerance (chapter 8)
Week 11: Lecture/slide set 18: fault tolerance
  Concepts and metrics
Week 12: Lecture/slide set 19: redundancy
  Systematically applying redundancy; software and data

Atomicity (chapter 9)
Week 13: Lecture/slide set 20: atomicity
Week 14: Lecture/slide set 21: atomicity logs
Week 15: Lecture/slide set 22: atomicity and locks