EEL-4736/EEL-5737 Principles of Computer System Design
Syllabus – Fall 2017

Text: "Principles of Computer System Design", Jerome E. Saltzer and M. Frans Kaashoek, ISBN 9180123749574, Morgan Kaufmann 2009. Readings will also be based on a collection of relevant technical papers.

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 assignment and project will require 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 40% for homework/project assignments, 60% for exams/quizzes.

Course topics:

Overview of computer systems: sources of complexity and design principles

- Modularity, Abstraction, Layering, Hierarchy

Elements of computer system organization

- Memory, interpreters, communication links

Layering and naming in computer systems

- Case study: UNIX file system

Enforcing modularity

- Clients and servers; virtualization

Designing for performance

- Metrics; latency and throughput; queuing
- Exploiting workload properties, concurrency; addressing bottlenecks

The network as a system and as a system component

- Network layers; end-to-end
- System design issues

Fault tolerance

- Concepts and metrics
- Systematically applying redundancy; software and data

Atomicity