General Information
Lecture time: M, Wd 6:00-7:15 PM,
Lecture hall: ENG 0383

Instructor: Dan C. Marinescu
Office Hours: M, Wd 5:00-6:00 PM or by appointment.
Office Location: HEC 304.
Class Web site: http://cs.ucf.edu/~dcm/Teaching/COP5611-Spring2013/

Final Exam:

Textbook: “Principles of Computer Systems Design; An Introduction, Vol 2” by Jerome Saltzer and Frans Kaasahoek. Publisher: Morgan Kaufmann, ISBN 978-0-12-374957-4. The students are not required to buy any textbook for this class, the text is available online at http://ocw.mit.edu/resources/res-6-004-principles-of-computer-system-design-an-introduction-spring-2009/online-textbook/networks_open5_0.pdf (the first volume was published in July 2009).

The textbook reflects the experience of the two authors in teaching the subject at MIT for many years. One of the authors of the textbook, Prof. Jerome Saltzer, helped formulate the undergraduate curriculum in Computer Science, and developed the core subject on computer systems engineering at MIT. In mid 1960s he was involved in all aspects of the design and implementation of the Multics system and more recently in the design of the Kerberos authentication system.

The topics covered in Volume 2 are:

1. Chapter 7. The Network as a System and as a System Component
2. Chapter 8. Fault Tolerance: Reliable Systems from Unreliable Components
3. Chapter 9. Atomicity: All-or-Nothing and Before-or-After
4. Chapter 10. Consistency
5. Chapter 11. Security

Additional references: Class notes for several topics will be provided. Suggestions for the most relevant readings on the topics covered in the text are are given on pages 375-423 of Volume 1 and at the end of Volume 2.

Due dates: assignments, research papers, and project phases are due on Thursday of the week specified by 10 AM; all should be submitted by Email with the required subject. All submission should be in .pdf; the students are encouraged to use either Word or Latex for text and Visio for figures.

Assignments: There are two homework assignments each consisting of 5-6 problems from the textbook.

Research paper: The research paper should be focused on a specific aspect of reliability of computing and communication systems chosen by each student. Chapter 8 provides a wealth
of references the students could use. The paper should follow the traditional organization of a paper submitted to research publication: title, authors, abstract, introduction and motivation, basic concepts, several sections devoted to the subject, conclusions, literature. The paper should be written in Word or Latex and the drawings should be done in Visio. Timeline:

- Week 5 - Submit a pdf file containing the skeleton of the paper and including, the title, and the fully developed abstract, introduction, basic concepts, and literature.
- Week 11 - Submit the full paper.

The subject of the Email should be: your name - Research paper

Project The students are encourage to propose projects related to their own research interests; such projects require the approval of the instructor. Timeline:

- Week 2 - Submit by Email the project proposal including details regarding the implementation language and environment. Identify three main phases of the project, e.g., completion of the general design, completion of the basic implementation, and completion of the project.
- Week 6 - Submit by Email a progress report for the first phase.
- Week 9 - Submit by Email a progress report for the second phase.
- Week 12 - Submit by Email a progress report for the entire project.

The subject of the Email should be: your name - project phase 0/1/2/3.

Class attendance and grading: The students are strongly encouraged to attend every class and be active. Our policy is simple, based on professional standards: on exams you should not collaborate. On all other assignments you are welcome to work with anyone else on ideas and understanding, but your writing should be your own and you should carefully acknowledge all contributions of ideas by others, whether from classmates or from papers you have read. Acts of academic dishonesty will not be tolerated; when detected they lead to unconditional failure of the class. The grading scheme is:

- Homework: 15%
- Project: 40%
- Research paper: 25%
- Final: 20%

Syllabus: The schedule is tentative and will be adjusted to the needs of the class; when a topic seems more difficult we shall spend more time on it.

1. Weeks 1-2. Basic concepts

   (a) The Abstractions: Interpreter, Memory, Communication Links
2. Weeks 3-5. Introduction to Parallel and Distributed Systems

(a) Parallel Systems
(b) Distributed Systems
(c) Communication in a Distributed System
(d) Process Coordination
(e) Logical Clocks
(f) Message Delivery Rules
(g) Causal Delivery
(h) Runs and Cuts; Causal History
(i) Enforced Modularity, the Client-server Paradigm
(j) Consensus Protocols
(k) Models of Concurrency: Petri Nets


(a) Store and Forward Networks - the Internet
   i. Isochronous and Asynchronous Multiplexing
   ii. Packet Forwarding; Delay
   iii. Buffer Overflow and Discarded Packets
   iv. Duplicate Packets and Duplicate Suppression
   v. Damaged Packets and Broken Links
   vi. Reordered Delivery
   vii. Summary of Interesting Properties and the Best-Effort Contract
   viii. The Link Layer
   ix. The Network Layer
   x. The End-to-End Layer
   xi. Additional Layers and the End-to-End Argument
   xii. Mapped and Recursive Applications of the Layered Model

(b) Multiple access
   i. The model of a multiple access network
   ii. Scheduled and non-scheduled access
   iii. CSMA-CD the Ethernet
   iv. Collision resolution algorithms
   v. The stack algorithm

(c) Congestion control
   i. Managing Shared Resources
   ii. Resource Management in Networks
iii. Cross-layer Cooperation: Feedback
iv. Cross-layer Cooperation: Control
v. Other Ways of Controlling Congestion in Networks
vi. Delay Revisited
(d) Combining store and forward and multiple access networks, ARP


(a) Faults, Failures, and Fault Tolerant Design
   i. Faults, Failures, and Modules
   ii. The Fault-Tolerance Design Process

(b) Measures of Reliability and Failure Tolerance
   i. Availability and Mean Time to Failure
   ii. Reliability Functions
   iii. Measuring Fault Tolerance

(c) Tolerating Active Faults
   i. Responding to Active Faults
   ii. Fault Tolerance Models

(d) Systematically Applying Redundancy
   i. Coding: Incremental Redundancy
   ii. Replication: Massive Redundancy
   iii. Voting
   iv. Repair

(e) Applying Redundancy to Software and Data
   i. Tolerating Software Faults
   ii. Tolerating Software (and other) Faults by Separating State
   iii. Durability and Durable Storage
   iv. Magnetic Disk Fault Tolerance

5. Weeks 11-12. Atomicity: All-or-Nothing and Before-or-After.

(a) Atomicity
   i. All-or-Nothing Atomicity in a Database
   ii. All-or-Nothing Atomicity in the Interrupt Interface
   iii. All-or-Nothing Atomicity in a Layered Application
   iv. Some Actions With and Without the All-or-Nothing Property
   v. Before-or-After Atomicity: Coordinating Concurrent Threads
   vi. Correctness and Serialization
   vii. All-or-Nothing and Before-or-After Atomicity

(b) All-or-Nothing Atomicity - Concepts
   i. Achieving All-or-Nothing Atomicity
ii. Systematic Atomicity: Commit and the Golden Rule
iii. Systematic All-or-Nothing Atomicity: Version Histories
iv. How Version Histories are Used

(c) All-or-Nothing Atomicity - Pragmatics
   i. Atomicity Logs
   ii. Logging Protocols
   iii. Recovery Procedures
   iv. Other Logging Configurations: Non-Volatile Cell Storage
   v. Checkpoints
   vi. What if the Cache is not Write-Through?

(d) Before-or-After Atomicity - Concepts
   i. Achieving Before-or-After Atomicity: Simple Serialization
   ii. The Mark-Point Discipline
   iii. Optimistic Atomicity: Read-Capture (Advanced Topic)
   iv. Does Anyone Actually Use Version Histories for Before-or-After Atomicity?

(e) Before-or-After Atomicity - Pragmatics
   i. Locks
   ii. Simple Locking
   iii. Two-Phase Locking
   iv. Performance Optimization
   v. Deadlock; Making Progress

(f) Atomicity across Layers and Multiple Sites
   i. Hierarchical Composition of Transactions
   ii. Two-Phase Commit
   iii. Multiple-Site Atomicity: Distributed Two-Phase Commit
   iv. The Dilemma of the Two Generals


   (a) Cache Coherence
      i. Coherence, Replication, and Consistency in a Cache
      ii. Eventual Consistency with Timer Expiration
      iii. Obtaining Strict Consistency with a Fluorescent Marking Pen
      iv. Obtaining Strict Consistency with the Snoopy Cache

   (b) Durable Storage Revisited: Widely Separated Replicas
      i. Durable Storage and the Durability Mantra
      ii. Replicated State Machines
      iii. Shortcuts to Meet more Modest Requirements
      iv. Maintaining Data Integrity
      v. Replica Reading and Majorities
vi. Backup
vii. Partitioning Data

(c) Reconciliation
   i. Occasionally Connected Operation
   ii. A Reconciliation Procedure
   iii. Improvements
   iv. Clock Coordination


   (a) Introduction to Secure Systems
      i. Threat Classification
      ii. Security is a Negative Goal
      iii. The Safety Net Approach
      iv. Design Principles
      v. A High \( \frac{d(\text{technology})}{dt} \) Poses Challenges For Security
      vi. Security Model
      vii. Trusted Computing Base

   (b) Authenticating Principals
      i. Separating Trust from Authenticating Principals
      ii. Authenticating Principals
      iii. Cryptographic Hash Functions, Computationally Secure, Window of Validity
      iv. Using Cryptographic Hash Functions to Protect Passwords

   (c) Authenticating Messages
      i. Message Authentication is Different from Confidentiality
      ii. Properties of SIGN and VERIFY
      iii. Public-key versus Shared-Secret Authentication
      iv. Key Distribution
      v. Long-Term Data Integrity with Witnesses

   (d) Authorization: Controlled Sharing
      i. Authorization Operations
      ii. The Simple Guard Model