

CMSC 417: Computer Networks

Neil Spring

Spring 2008

Instructor Neil Spring

E-mail nspring@cs (include “417” in your subject line)

Class TuTh 3:30-4:45 CSIC 1122

Office hours TBA AVW 4133

TA Bo Han: bohan@cs

Web <http://www.cs.umd.edu/class/spring2008/cmsc417/>

Textbook Peterson & Davie, *Computer Networks*, 4th ed.

Supplement Donahoo & Calvert, TCP/IP Sockets in C

1 Goals of this course

The goal of this course is to prepare you to use and develop the software of networks. We will learn about the architectural features of networked software, the problems protocols solve, and the problems left for end-host software to address. At the end of this class, you should be able to design and build a simple but complete network stack.

2 Summary

This course will use an unconventional, problem-focused organization that largely ignores the “layer” of each solution. It is a new and relatively-untested approach to organizing course content.

This course will cover the four main problems of networking:

error detection and reliability – framing, error correcting codes, retransmissions, flooding, routing.

sharing and multiplexing – CSMA, flow control, congestion control, quality of service, traffic engineering.

growth and evolution – layering, multicast, caching, addressing, naming, overlays.

cooperation and competition – security, attacks, pricing, inter-domain connectivity.

Unfortunately, no textbook uses this organization. The layer-based organization, in my experience, confuses more than it organizes. The unfortunate consequence is that it will sometimes be up to you to determine which sections of the book are appropriate.

3 Prerequisites

Experience in CMSC412 (operating systems) may help you.

CMSC351 – Algorithms.

CMSC311 or ENEE350 – Computer Organization.

CMSC330 – Programming Languages.

You must know what a function pointer is and how it is used. Find a book on C today if you do not.

You must know how to write basic data structures or have basic data structures on-hand: queues (heaps if you're advanced) and hash tables are necessary.

You should understand basic issues of concurrency. That includes the interactions between non-blocking sockets, user-level and kernel-level threads, locking, etc. Too many students seem to think that forking a thread will solve a simple problem without creating many more.

4 Grading

4.1 Participation: 5%

In a class so large, I can't expect each of you to speak; participation here is a negative grade, if I think you're doing poorly and it's your own fault for not being engaged with the material, you won't get the participation bump.

4.2 Homeworks and Cell-Phone Quizzes: 10%

A few assignments to take home and complete will be graded. Quizzes given when cell phones ring are included here.

Quizzes given in class when cell phones ring will also be graded here. You will dread the cell phone quiz if you allow yourself to fall behind.

4.3 Midterm Exam: 20%

4.4 Final Exam: 30%

The midterm and final exams will mix multiple choice, simple matching, short answer and long answer questions. The midterm will consume a lecture slot, the final during finals week as scheduled by the university. The exams will be longer than will allow all of you to finish the entire exam. You will have to learn and study before the exam.

4.5 Programming Assignments: 35%

I'm still revising the ideal programming assignment for this environment. It will likely be under-specified and quite exciting for those who want to play, quite frustrating for those who want clear milestones, metrics, and supplied test cases.

I will teach coding concepts in the programming assignments as if you are implementing in C. C is the one true language. C is beautiful. C is the language of networks. C function calls correspond directly to many system calls; the system calls involved form the sockets API. Other languages attempt to hide errors that occur in the network and do so poorly.

You will have the option of completing later programming assignments in Ruby.

5 Lateness

All programming assignments can be turned in electronically. I will permit one programming assignment to be turned in after the weekend (when due Friday, it can be turned in on Monday). I expect any data loss due to dogs, roommates, lightning strikes or FBI confiscating your machine can be dealt with over a weekend.

6 Administrative Cruft

I dislike this section greatly, but codifying each of these policies is important for keeping myself sane and making clear what my expectations are. I'd much prefer a section that said "treat me with respect and I'll do the same for you;" this section is intended mostly for those who would hope to game the system. Note that I copied verbatim some of these passages; I hope you appreciate irony.

6.1 Excused absences

Students claiming a excused absence must apply in writing and furnish documentary support (such as from a health care professional who treated the student) for any assertion that the absence qualifies as an excused absence. The support should explicitly indicate the dates or times the student was incapacitated due to illness. Self-documentation of illness is not itself sufficient support to excuse the absence. An instructor is not under obligation to offer a substitute assignment or to give a student a make-up assessment unless the failure to perform was due to an excused absence. An excused absence for an individual typically does not translate into an extension for team deliverables on a project.

6.2 Religious observances

I have made an effort to avoid deadlines 9/22–24 and 10/1–2 (Rosh Hashanah and Yom Kippur). Please inform me in advance of religious observances that will interfere with your ability to complete assignments on time.

6.3 Honor code

The University of Maryland, College Park has a nationally recognized Code of Academic Integrity, administered by the Student Honor Council. This Code sets standards for academic integrity at Maryland for all undergraduate and graduate students. As a student you are responsible for upholding these standards for this course. It is very important for you to be aware of the consequences of cheating, fabrication, facilitation, and plagiarism. For more information on the Code of Academic Integrity or the Student Honor Council, please visit <http://www.studenthonorcouncil.umd.edu/whatis.html>.

6.4 What constitutes cheating?

Copying other assignments, looking over someone's shoulder in the lab, emailing function code, using google to find a code fragment without understanding, looking for code in other people's directories, pulling code printouts off printers, not working with your partner, alternating assignments with your partner, and in any other way attempting to gain a grade without learning.

Note: cheating goes both ways; leaving someone your code because you want to help is just as bad as borrowing someone else's code. We can tell when code looks and acts too similar to be independent work; we can't (easily) tell which of two implementations was the original.

6.5 What constitutes legal collaboration?

Interaction via mailing list or discussion of problem and code solutions governed by the Gilligan's Island rule.¹ Collaborative interoperability testing, experimenting with whether two independent implementations can talk to each other, is also permitted, so long as code is not shared.

Using google is OK. Using wikipedia is encouraged. If you find a particularly good solution on either, please cite it; there is no penalty for citing sources and I'm more likely to consider answers that disagree with textbook or lecture legitimate. If you find a question far too easy because an answer is present on-line, please let me know.

¹You understand the concept only if you can watch one half-hour complete episode of Gilligan's Island and still retain the concept. You may then begin coding with your newfound knowledge safe that it is your own work. Without the thirty minute pause, it is not your work.