Computer Science 78

Computer Networks – or hacking the network, part II

In what follows, we discuss the course overview, grading, books, weekly schedule, laboratory assignments and group projects.

Course Overview

Welcome to the world on computer networks. Ever wondered what makes the Internet tick? Want to gain the skills that would allow you to implement the Internet? Read on ....

The ORC: This course focuses on the communications protocols used in computer networks: their functionality, specification, verification, implementation, and performance; and how protocols work together to provide more complex services. Aspects of network architectures are also considered. Laboratory projects are an integral part of the course in which networking concepts are explored in depth.

Prerequisite: CS 23 and CS 37. You need strong C and Unix (shell programming, gdb, cvs, process, threads) skills (taught in CS 23) to do this course. All programming assignments and project are in C and Unix.

Topics covered in this course include network applications (e.g., peer-to-peer applications such as BitTorrent), TCP and UDP transport protocols, flow control, congestion control, the IP protocol, routing algorithms, medium access layer (e.g., IEEE 802.11 wireless MAC) and Internet infrastructure such as domain name system (DNS) - these comprise the Internet's core components.

There is a significant amount of programming in this course requiring a significant time commitment on the part of the student.

The course breaks down into lectures, labs, and a course project - note, there are no exams for this class, i.e., no mid term and no final. We focus on network programming. The projects will run for the last two weeks of the course. There will be no lectures during the last week of the course so students can fully focus on their group projects. The projects will be organized around a design review, code review and demo or die event.

The course will include seven stepwise programming assignments based around building your own P2P IM system, implementing the Simple Reliable Transport protocol (SRT), Simple Network Protocol (SNP) (including packet forwarding and routing) and Overlay Network (ON). Collectively, we call the transport, network, and overlay software system DartNet.

Students will implement DartNet over the course of the programming assignments and then "put it all together" so that their IM application uses their transport and network layer running over an IP overlay network which students also code. Development of DartNet gives students a knowledge base and skill set that mirrors what computer protocol developers execute in industry. So each student will run their IM app over their transport and routed network that they build - huge insights will be gained in doing this. And, believe me when you tell people that you did this as an undergrad, even at Cisco, they'll be way impressed! Having worked in the software networking industry for a decade before falling into academia I'm a big believer in implementing networking software as the essential part of fully understanding complex computer networks - such as, the Internet - else, things remain too abstract. You will write approximately 3000 lines of C code for the assignments and 3500 lines for the group project (shared between the group members). Note, there is some code reuse for the DartNet set of labs. As a result you will tangible programming experience in building a network app, and coding transport and network

software. If you code it you'll understand it. Or as Internet pioneer David Clark famously put it: "We reject kings, presidents, and voting. We believe in rough consensus and running code.". David was referring to the working practice of how the Internet Engineering Task Force (IETF) went about developing the core Internet standards.

This is an exciting and demand set of experiments that will reinforce concepts taught in class through network programming assignments. This is the second time we have done this so we will cut you some slack and hope you will reciprocate as we "debug" any issues that come up.

The group project (approx. 4 people in each groups - the lecturer will set up the groups) will use the wireless Nokia Internet Tablets (below) which runs embedded Linux. You will program these devices using socket programming. We will provide each group with one or two tablets each and define a common project goal that students can build from. More on that as the course progresses. We plan to have you implement some research ideas we have developed in the SensorLab - bring research into the classroom, it's cutting edge.

Grading

10% - Class and lab participation - jump in, get involved, be a good CS78 citizen.

There will be a discussion in class regarding the course reading material. Active involvement in that discussion, the class, the lab - will help toward the class contribution part of the grade.

70% - Laboratory exercises (10% for each of the 7 labs)

There are 7 weekly laboratory assignments over the first 8 weeks. 10% is given for each lab. Some labs are harder than others but we have a flat grading scheme across all labs. These assignments are to be done individually. The schedule is online - plan a head. You need to be organized to get through the labs and stay on schedule.

We will provide source code solutions to all labs. The TA will grade your solutions and write up a grade sheet on the correctness, simplicity, and clarity of your code. The instructor will review the TA's grading and grade sheets. Your grade and grade sheet and our source code solution will be mailed to you one week after the lab assignment is turned in. If you have questions about the grade or grade sheet please talk to the TA first. If you are still have concerns see the instructor. Please do not distribute the source code solutions we provide you with (see honor code).

20% - Team project

The project is made up of a small team (approx four people) and requires strong collaboration and a problem solving mindset to get the job done. The instructor will put the teams together with each member being responsible to deliver against a part of the overall system design, implementation, testing and integration. The goals of this activity are to help you develop the confidence, skills, and habits necessary to write large computer programs while part of a multi-person team. You will become conversant in software engineering paradigms, and be exposed to various public-domain and open source tools that make the software development process easier. In addition, you will develop vital skills in self-directed learning, problem solving, and communication. The project will have a design and code review as well as the demo. A project report that captures the design and implementation will be submitted as part of the assessment.

Books

This is the course book (5h Edition now available at the bookstore):

Computer Networking: A Top-Down Approach (5th Edition) by James F. Kurose and Keith W. Ross

Not required but if I where to recommend a hands on book on Linux and shell programming it would be this one (lots of good stuff in this):

A Practical Guide to Linux Commands, Editors, and Shell Programming by Mark G. Sobell

Another really good book covering, debugging, processes, threads, and socket programming in clear and easy manner to grasp:

Beginning Linux Programming, 4th Edition by Neil Matthew, Richard Stones

Just to make things clear. Buy the Computer Network book for the course. But if you are interested in hacking C, shell scripts, Linux you might like the other books for your library. They are all really nice books - I love them.

Schedule

Week 1
Lecture 1 Computer networks in a nutshell -- Featuring the centralized and distributed "Roll Call" protocols
Lecture 2 Computer networks in a nutshell --- Real life demo of "Internet in a Box" - you saw here first!
Lecture 3 Socket Programming (could be a refresher for cs23 kids)

Week 2
Lecture 4 Socket Programming
Lecture 5 Applications: Starting with Instant Messaging (IM) (see also Internet Draft MSNP 1.0)
Lab Debugging, valgrind (refresher).
Lecture 6 P2P Apps: Gnutella, Kazaa, BitTorrent and Skype

Week 3
Lecture 7 P2P Apps: Gnutella, Kazaa, BitTorrent and Skype (continued)
Lecture 8 Classic Apps: Email and Web
Lab C threads, mutex, timers (refresher) for IM assignment.
Lecture 9 Domain Name System (DNS) and some network tools.

Week 4
Lecture 10 Transport layer (mutex/demutex). DartNet: Simple Reliable Transport (SRT)
Lecture 11 DartNet: Simple Reliable Transport (SRT)
Lecture 12 TCP: signalling, mux/demux, segment (header/data)

Week 5
Lecture 13 DartNet: Simple Reliable Transport (SRT): Data Transfer
Lecture 14 TCP: sliding window, retransmit and fast retransmit
Lecture 15 TCP: congestion control

Week 6
Lecture 16 DartNet: Overlay Network (ON) Design
Lecture 17 Internet Protocol
Lab IP Addressing and Forwarding
Lecture 18 Routing Algorithms

Week 7
Lecture 19 Routing Protocols, DartNet: Simple Network Protocol (SNP) Design
Lecture 20 Routing Protocols
Lab The MAC layer: WiFi 802.11

Lecture 21 Project and tutorial I on N810s

Week 8
No classes: time held over for project work.
Design review

Week 9
No classes: time held over for project work.
Code review

Week 10
Demo or die day.
Project reports due.

Lab Assignments

Lab1 - Packet sniffing
Lab2 - Reverse engineering sockets: a puzzle
Lab3 - IM application
Lab4 - DartNet: Simple Reliable Transport (SRT-SIG)
Lab5 - DartNet: Simple Reliable Transport (SRT-GBN)
Lab6 - DartNet: Overlay Network (ON)
Lab7 - DartNet: Simple Network Protocol (SNP)

This is an initial schedule that might change. You will be informed in advance if that is the case. All assignments are due at 12 midnight.

As part of getting you to complete we allow the weekend to work on the labs. Typically labs are given out at Monday @ class and due at midnight on Sunday. There are TA hours on Wednesday, Saturday, and Sunday in support of these assignments. You can mail the TA anytime - he is mostly always around and available.

Labs 4-7 are the DartNet set. As part of Lab7 you build the overlay network with SRT and SNP and then run your IM app over the network. This is challenging and very cool once you get it working. In essence you have built a simplified Internet stack. Now is that cool!

You will write approximately 3000 lines of C code for the assignments and 3500 lines for the group project (shared between the group members). Note, there is some code reuse for the DartNet set of labs.

Lab 2 (Sockets): 200
Lab 3 (IM): 483
Lab 4 (SRT-SIG): 371
Lab 5 (SRT-GBN): 542 (171 new lines)
Lab 6 (Overlay): 709
Lab 7 (SNP): 1169

Project: 3586 (shared across the group).

Group Project

There are no lectures during the last two weeks of the course.

During that time you will be working in teams programming N810s. These are embedded Linux devices that have a bunch of wireless interfaces and sensors (GPS for example). The Webpage

for last year's project including teams, news, announcements, resources, description can be found here.

Important Project Milestones

Project and tutorial I on N810s
Project and tutorial II on N810s

The following set of deadlines is important for the progress of the project. Please note that you need to provide documentation for the design/code reviews and the final project submission.

Notice that we will use cvs for code management for the project. All code needs to be signed as part of the submission. In addition, the final report has to be written using latex and the source and a pdf submitted in the project cvs directory.

The following set of deadlines is important for the progress of the project. Please note that you need to provide documentation for the design and code reviews and the final project submission.

We will use cvs for code management for the project. All code needs to be signed as part of the submission. In addition, the final report has to be written using latex and the source and a pdf submitted in the project cvs directory.

Design Review. The project review should include requirements, Design Spec (inputs/outputs, data flow, data structures, pseudo-code) and functional decomposition; How the project implementation breaks down who is doing what. The project review material is due 12 PM day before review Send tarball documentation to cs78@cs.dartmouth.edu

Code Review. The code review should include the Implementation Spec, unit tests and whatever code is written up until the review point. The code review material is due 12 PM day before review. Send tarball documentation and source tree to cs78@cs.dartmouth.edu

Demo or die day. Project presentation (design overview, lessons learnt, etc) and demo of project.

Project reports due. The report (max 10 pages) written in latex and include:

1) Thread design of client
2) Design Specs
3) Implementation Specs
4) GUI screen dump
5) Lesson learnt