Introduction to Networking

Syllabus

Web Page

http://www.cs.northwestern.edu/~pdinda/netclass-w03

Instructor

Peter A. Dinda
1890 Maple Avenue, Room 338
847-467-7859
pdinda@cs.northwestern.edu
Office hours: Thursdays, 2-4pm or by appointment

Teaching assistant

Jason Skicewicz
1890 Maple Avenue, Room 332
847-491-7150
jskitz@cs.northwestern.edu
Office hours: Mondays 3-6pm or by appointment

Location and Time

1890 Maple Avenue, CS Department classroom, MWF 2-2:50pm

Prerequisites

Required	CS 311 or equivalent data structures course
Required	Knowledge of C and C++
Highly recommended	CS 213 or equivalent computer systems course
Highly recommended	CS 343 or equivalent operating systems course
Highly recommended	Unix development experience (gcc, gdb, make, etc)
Recommended	Unix systems programming experience

Textbook and other readings

James Kurose and Keith Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, Second Edition, Addison Wesley, 2002 (Textbook)

- Buy the hardcover second edition, not the softcover "preliminary edition" or the first edition. The book includes access to the on-line version at http://www.awl.com/kurose-ross.

Richard Stevens, *TCP/IP Illustrated, Volume I: The Protocols*, Addison Wesley, 1994, (ISBN: 0-201-63346-9) (Required)

- Very detailed look at the protocols, including tcpdump from the wire. Essential to doing the projects well.

Internet Requests For Comments (RFCs) (Useful)

- Official specifications for the Internet which are available from http://www.ietf.org/rfc.html

Richard Stevens, Unix Network Programming (volumes 1 and 2), Prentice Hall, 1997, 1998 (ISBN 0-134-90012-X and 0-130-81081-9) (Useful)

- Describes the nitty-gritty details of socket programming and IPC on Unix

Richard Stevens, Advanced Programming in the Unix Environment, Addison-Wesley, 1992 (ISBN 0-201-56317-7) (Useful)

- Describes how to think like a Unix systems programmer

Bjarne Stroustrup, The C++ Programming Language, Special Edition, 2000, Addison-Wesley, (ISBN 0201700735) (Useful)

- Definitive reference to C++

Objectives, framework, philosophy, and caveats

This course introduces the underlying concepts behind networking using the Internet and its protocols as examples. There are three goals: (1) to give you an understanding of how networks, especially the Internet, work, (2) to give you experience in "programming in the large", and (3) to teach you network programming.

We will cover the first five chapters of Kurose in detail, working our way down the network stack from the application layer to the data-link layer. Concurrent with the lectures, you (in groups of two) will be building a functional TCP/IP stack and a small web server that will run on it. What you build will be "real" – your code will interoperate with other TCP/IP stacks and you'll be able to talk to your web server using any browser on any TCP/IP stack.

This is a learn-by-doing kind of class. You will get your hands dirty by examining parts of our Internet infrastructure and building other parts. It will be a lot of work, but it will also be a lot of fun, provided you enjoy this sort of thing. We will assume that you do and that you will make a good faith effort. We don't want to have to spend too much time measuring your performance. If you care about what we're teaching, you'll do a better job of that yourself, and if you don't care, then you should take some course that you do care about.

After finishing the course, you will be able to do the following.

- Understand the Internet protocols
- Build implementations of the Internet protocols
- Generalize this knowledge to other networking protocols.
- Be a competent network and systems programmer.
- Think like a networking practitioner
- Read and judge articles on networking in trade magazines
- Begin to read and judge research and technical articles on networking

- Create simplicity and reliability out of complexity and unreliability
- Structure and design software systems to achieve that simplicity and reliability

Project

Over the course of the quarter, you will implement a user-level TCP/IP stack and a small web server that runs on top of it. Your code will not implement the full functionality of HTTP or TCP/IP, but it will implement enough of it to be able to interoperate with other, complete implementations. In keeping with the top-down approach of Kurose and Ross, you will build this from the web server down instead of from the network card up. I will initially provide you with the whole stack (as object code) and you will implement the web server. Next, I will peal away the layers of the stack, leaving you to implement your own versions. Each layer will have well-defined interfaces that you will fill out.

Here are the layers, as well as each one's percentage of the project grade. Note that the layers in italics will be supplied to you and are included only for completeness.

20 %	**Web server (a)**
	Sockets
50 %	**TCP (b)** *UDP*
30 %	**IP (c)** *ARP*
	Ethernet

The implementation language will be C++ and the platform will be Red Hat Linux 7.3. We hope that you will use g++ 2.96 or later as your compiler, make as your build tool, and CVS as your version control system. You may also find that the C++ standard template library will make your life easier. You'll be using the PCs in the TLAB, which will be specially configured for this class. You are welcome to use other machines, but we must be able to compile and run your code on our machines. Note that the Ethernet layer of the code requires that your kernel supports the Berkeley packet filter interface and that you can run your Ethernet card in promiscuous mode to extract and inject raw packets.

To evaluate your project, we will spot-check your source code, compile it, and run randomized testcases on it. When appropriate, we will supply you with examples of such testcases.

In this iteration of the class, part (c) of the project will probably involve a separate routing simulation instead of an actual small implementation of IP. I have not decided yet which of these we'll be doing.

Homework

There will be four homework problems sets that will be periodically assigned to help you improve your understanding of the material.

Exams

There will be a midterm exam and a final exam. The final exam will not be cumulative.

Grading

50 % Project
20 % Midterm
20 % Final
10 % Homework

Final grades will be computed in the following way. A final score from 0 to 100 will be computed as a weighted sum of each of the project components, the homeworks, and the exams. Scores greater than 90 or greater than 90th percentile will be assigned As, scores greater than 80 or greater than 80th percentile will be assigned Bs, scores greater than 70 or greater than 70th percentile will be assigned Cs, scores greater than 60 or greater than 60th percentile will be assigned Ds, and the remainder will be assigned Fs. Notice that this means that if everyone works hard and gets >90, everyone gets an A. Please choose wisely where you put your time.

Late Policy

For each calendar day after the due date for a homework or a lab, 10% is lost. After 1 day, the maximum score is 90%, after 2 days, 80%, etc, for a maximum of 10 days.

Cheating

Since cheaters are mostly hurting themselves, we do not have the time or energy to hunt them down. We much prefer that you act collegially and help each other to learn the material and to solve development problems than to have you live in fear of our wrath and not talk to each other. Nonetheless, if we detect blatant cheating, we will deal with the cheaters as per Northwestern guidelines.

Schedule

Please don't let all this reading frighten you. Only KR and the HOs are required reading. RS is highly recommended and will help a lot in doing the projects. You almost certainly will end up reading RS as you work on the project, unless you are either a genius or insane. RFCs are quite long, so I recommend that you scan them.

A good progression is to read KR and HO, skim RS, listen to the lecture, work on the project until you're confused, and then take a deeper look at RS. A key thing that RS gives you is an example of what a correct implementation behaves like. When you have a question about the correctness of an implementation decision, it's time to look at the RFCs.

Lecture	Date	Topics	Readings	Homework/Project

	1/6 M	(classes begin)		
1	1/6 M	Mechanics, Introduction, Fundamental concepts: communication models, network, end-to-end philosophy, protocol, stacks, layering, TCP/IP	KR ch 1 RS ch 1, TLAB HO	Project Part (a) out Find partner (groups of up to 2) Install and play with Minet
2	1/8 W	Fundamental concepts: tools for exploration, much more on protocols and what they do, encapsulation, TCP/IP	KR ch 1 RS ch 1, 7, 8, A Tools HO	Homework 1 out
3	1/10 F	Fundamental concepts: latency, bandwidth, delay, throughput, goodput, naming, addressing, services	KR ch 2, RS ch 1, 14	
4	1/13 M	Application layer protocols, focusing on HTTP 1.0: client/server, text-based protocols, GET/POST, headers, static and dynamic content, HTTP 1.1 extensions	KR ch 2 RFC 1945	
5	1/15 W	Unix Systems Programming: error model, data model, file descriptors, open/read/write/seek/close, semantics, files, blocking and select	KR ch 2 USP HO, SP HO, MSI HO	
6	1/17 F	Unix Network Programming: select, IPC, connection concept, socket abstraction, socket creation	KR ch 2 USP HO, SP HO, MSI HO	
7	1/20 M	Unix Network Programming: socket programming, Minet socket interface	KR ch 2 USP HO, SP HO, MSI HO	
8	1/22 W	Project Help Day or Slack Day		
9	1/24 F	Application layer protocols: video and audio: data types, issues, compression, RTP, QoS networking	skim KR ch 6	
10	1/27 M	Transport layer: socket layer, logical connection, multiplexing, UDP	KR ch 3, RS ch 11	Project Part (a) in Project Part (b) out, Homework 1 in, Homework 2 out

11	1/29 W	Transport layer: the channel, error control and checksum	KR ch 3	
12	1/31 F	Transport layer: dealing with loss and re-ordering, pipelined protocols, stop-and-wait, go-back-n, selective-repeat	KR ch 3	
13	2/3 M	Transport layer: TCP: sequence numbers, segments, semantics, TCP as go-back-n with extensions	KR ch 3 RS ch 17-24, RFC 793	
14	2/5 W	Transport Layer: TCP: syntax and dataflow	KR ch 3 RS ch 17-24, RFC 793	
15	2/7 F	Transport layer: TCP: flow control and connection establishment	KR ch 3 RS ch 17-24, RFC 793	
16	2/10 M	Transport layer: TCP: congestion control and connection management	KR ch 3 RS ch 17-24, RFC 793	
17	2/12 W	Project Help Day or Slack Day		Homework 2 in Homework 3 out
18	2/14 F	Network layer: packet, circuit, virtual circuit switching, distributed graph algorithms, routing algorithms, flooding, hot potatoe (age-test: who is this an homage to?)	KR ch 4	
19	2/17 M	Network layer: routing algorithms: link-state & Djikstra, distance vector & Bellman-ford, hierarchical routing, IP routing	KR ch 4 Graph algs HO. RS ch 3, 9, 10, RFC 791	
Midterm, tentatively Monday, 2/17, 6-7:30pm (covers 1-16)				
20	2/19 W	Network layer: IP routing, ICMP, BGP, RIP, OSPF	KR ch 4 RS ch 3, 6 9, 10, RFC 791	
21	2/21 F	Network layer: IP Fragmentation, IPv6, broadcast and multicast	KR ch 4 RS ch 3, 9, 10, 12, 13, RFC 791	Project Part (b) in Project Part (c) out

22	2/24 M	Data-link layer: addressing, ARP, MAC, CS Dept network	KR ch 5, RS ch 2, 4	
23	2/26 W	Data-link layer: MAC, error control, NIC architecture, Ethernet	KR ch 5 RS ch 2, 4	Homework 3 in Homework 4 out
24	2/28 F	Ethernet: IP over Ethernet, ARP, different types of Ethernet, switches, hubs, etc, Gigabit Ethernet, 10 Gigabit Ethernet	KR ch 5 RS ch 2, 4 RFC 826, RFC 894	
25	3/3 M	ATM	KR ch 5	
26	3/5 W	Project Help Day or Slack Day		
27	3/7 F	Wireless	KR ch 5	
28	3/10 M	Project Help Day or Slack day		
29	3/12 W	Security	skim KR ch 7	Project Part (c) in
30	3/14 F	Networking Research topics	CNR HO	Homework 4 in
Final Exam: Wednesday, 3/19, 9-10:30am (covers 17-30)				

KR = Kurose and Ross
RS = Richard Stevens
HO = Handout
TLAB HO = The TLAB Cluster HO
USP HO= Unix Systems Programming in a Nutshell HO
SP HO = Sockets Programming in a Nutshell HO
MSI HO = Minet Socket Interface HO
MS HO = Minet TCP/IP Stack HO
CNR HO = Computer Networking Research HO