CSC364 2000S

Computer Networks

# Introductory Handouts

Monday, 24 January 2000

Samuel A. Rebelsky

*Grinnell College*

# Introductory Handouts

These materials are also available online. The online versions may not precisely match the printed ones, since I update my Web pages regularly. The online versions are the definitive ones.

# Catalog Description

This course focuses on the communications protocols used in computer networks: their functionality, specification, verification, implementation, and performance. The course also considers the use of network architectures and protocol hierarchies to provide more complex services. Existing protocols and architectures will be used as the basis of discussion and study. Includes formal laboratory work.

**Prerequisites**: Computer Science 211, Computer Science 213, or permission of instructor. *This semester, we've waived the prerequisites and are allowing CSC152 as a prerequisite. However, I won't go over things from the prerequisite courses.*

# Front Door

Welcome to the Spring 2000 session of Grinnell College's CSC 364, *Computer Networks*, which is described relatively well in the official blurb. My take on this course is that we'll study the theory of computer networks and common implementations. At the same time, we'll do some network programming (in C). This is the first time that CSC364 has been taught at Grinnell.

This is a systems course. This means that you will be doing a significant amount of programming in the course. It is also a project course, although I am still working out the details of the project.

In an attempt to provide up-to-date information, and to spare a few trees, I am making this as much of a ''paperless'' course as I can. You may also want to read the basic instructions for using this course web.

## Basics

**Meets**: MWF 2:15-3:05 p.m.

**Instructor**: Samuel A. Rebelsky, Science 2427. Office hours: Tu 2:15-4:15, W 3:15-4:15 (also feel free to stop by when my door is open)

**Grading**: Quizzes and labs: 10%; Assignments: 30%; Project: 30%; Exams: 30%.

This course has a midterm and a final. Both will be in-class exams.

**Project**: CSC364 has a project component. You should plan to work on the project in groups of size 4. I'm still working on the specifications, but it is likely that I will have you develop a simple architecture for transportable agents.

**Extra Credit**: I will occasionally give you quizzes to ensure that you're keeping up with the reading. Throughout the term, I may suggest other forms of extra credit.

## Books and Other Readings

Peterson, Larry L. and Davie, Bruce S. (2000). *Computer Networks: A Systems Approach*, Second Edition. San Francisco, CA: Morgan Kaufmann Publishers.

The primary text for the course and the best general book on computer networks I could find.

Stevens, W. Richard (1998). *Unix Network Programming, Volume 1: Networking APIs: Sockets and XTI*, Second Edition. Upper Saddle River, NJ: Prentice Hall PTR.

Your primary reference for doing network programming with sockets and TCP/IP. I expect this will stay as a reference on your shelves for years to come.

Kernighan, Brian W. and Ritchie, Dennis M. (1988). *The C Programming Language, Second Edition: ANSI C*. Upper Saddle River, NJ: Prentice Hall PTR.

An optional text. Since most of the programming in the course will be in C, you should benefit from having a reference. This is the standard reference on C. It's concise, clear, and written for real programmers.

Rebelsky, Samuel (2000). The CSC364 2000S Course Web.

The hypertext that you are currently reading. All of these materials are optional, but you may find them useful.

# Class 01: Introduction to the Course

**Held** Monday, January 24, 2000

**Overview**

Today, we begin our study of networks by considering some basic issues: what computer networks are, why we have computer networks, and what criteria we use in designing and building computer networks. We also begin the course by considering the normal range of administrative issues.

**Notes**

- *Assignments (for Wednesday):*
  - Read the introductory handouts.
  - Read chapter 1 of Peterson and Davie
  - Fill out the introductory survey
- Make sure to complete the Introductory survey for Wednesday's class! (A few students always seem to miss this.)
- On Thursday, January 27, at noon, I'll be giving a presentation on summer opportunities in computing and computer science. Some opportunities are available for non-majors.
- I know that many of you are prospective majors worrying about how to put together a major, so please feel free to come talk to me about scheduling. It is possible to be a CS major and still take a semester abroad.
- As usual, I'll start the course with a number of questions, some of which are answered in the course outline. Please don't refer to the outline when coming up with your answers.

**Contents**

- Defining Computer Networks
  - Why Build Computer Networks?
  - Aspects of Network Design
  - Goals for Network Design
- Building a Computer Network
- Administrative Issues

**Summary**

- About computer networks
- Some key issues: reliability, routing, resources, round-trip, and rate
- Network architecture, from bottom to top
- Course overview
- *Handouts:*
  - Course Overview (taken from the Course Web

# Defining Computer Networks

- I believe that we should begin the study of any subject by considering how we (and others) define the subject and what that tells us about what we expect to study.
- So ... *What is a computer network?*
- My ‘‘quick and dirty’’ definition is that

  > A computer network is the combination of hardware, software, and protocols that allows computers to communicate. Like computers, computer networks are general-purpose devices.

- In his textbook, Tanenbaum says

  > The old model of a single computer serving all of the organization’s computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These systems are called computer networks. The design and organization of these networks are the subjects of this book.

- Tanenbaum also says

  > Throughout this book we will use the term "computer network" to mean an interconnected collection of autonomous computers.

- Peterson and Davie said (in their first edition)

  > Before we can understand how to design a computer network, we should first agree on exactly what a computer network is. At one time, the term network meant the set of serial lines used to attach dumb terminals to mainframe computers. To some, the term implies the voice telephone network. To others, the only interesting network is the cable network used to disseminate video signals. The main thing these networks have in common is that they are specialized to handle one particular kind of data (key strokes, voice, or video) and they typically connect to special-purpose devices (terminals, hand receivers, and television sets).
  > &nbps; What distinguishes a computer network from these other types of networks? Probably the most important characteristic of a computer network is its generality. Computer networks are built primarily from general-purpose programmable hardware, and they are not optimized for a particular application like making phone calls or delivering television signals. Instead, they are able to carry many different types of data and they support a wide, and ever growing, range of applications.

- They say something fairly similar in this edition

## Why Build Computer Networks?

- This question may be as simple (and as difficult) as ‘‘Why Build Computers?’’. However, it is useful to consider what outcomes we hope from having computer networks.
- *Why are networks useful?*
- Networks permit the sharing of resources. Computers can share computation, data, storage, peripherals, ....

- Networks can support increased reliability
- Networks support human-human communication, such as e-mail, WWW, chat, videoconferencing, telephony, radio, ...
- It may also be helpful to contrast modern networks to the old ‘‘one large server plus many dumb terminals’’ model.
    - Why use the new model?
    - Why are people seeming to return to the old model?
    - Are they really returning?

## Aspects of Network Design

- What are the goals, requirements, and other issues we must consider when designing a network to support all of the applications above (and others we’ve not yet thought about)?
- How do I make sure the network is *reliable* or measure reliability?
- How do I *route* information between computers that are not directly connected?
- How do I ensure that *resources* are used fairly, efficiently, and appropriately?
- What is the *rate* I can transmit data?
- What is the *round-trip* time for my network? (More precisely, the latency of a network is an imporant factor to consider).
- [This is probably the first of many attempts to shoehorn related issues into words that with the same letter.]

## Goals for Network Design

- *What Makes a computer network good?*
    - Well, it depends on your perspective
- Users:
    - Broad range of services
    - Predictable/reliable
    - ...
- Designers
    - Good use of resources
    - Fair allocation
    - Scalable (?)
    - ...
- Implementers
    - Simple / understandable
    - Small
    - ...
- Providers
    - Easy to manage
    - Faults detectable
    - Scalable
    - ...

- ...

## Building a Computer Network

- Let's sketch the steps involved in building a computer network. Along the way (or afterwards), we'll try to identify key issues we might consider.
- We start by connecting computers by a wire.
  - How do we represent data?
  - How do we deal with errors?
- As we add more computers, not all are connected. We need to think about how to deal with groups of computers?
  - How do we route information?
  - How do we deal with errors (and what kinds of errors)?
- Eventually, we want to treat the group of computers as a logical whole. We need to think about how to connect the groups.
  - How do we route information?
  - How do we deal with errors (and what kinds of errors)?
- Then we need to think about how to provide different views of the network to applications, and what views to provide.
- We'll return to this issue on Wednesday.

## Administrative Issues

- Please refer to the course web site and the introductory handout for details.
- Teaching philosophy: I support your learning
- Policies
  - Attendance: I expect you to attend every class. Let me know when you'll miss class and why.
  - Grading: I'm a hard grader. I don't grade everything.
  - Course web
  - Etc.
- The exams
  - Midterm and final
  - In-class
- The books
  - One general text
  - One on Unix Network Programming
  - One optional reference
- The project
  - Still under development
  - Likely to emphasize transportable agents
  - Design first half of semester, implement second half

## How to Use the Course Web

For a number of reasons, I have chosen to make many of the handouts for this course available only in electronic format on the World-Wide Web. I will not go over basic use of the Web, since you should know about it from other courses. You should make sure to ask me if you have any questions about using the World-Wide Web.

The course web can be found at
`http://www.math.grin.edu/~rebelsky/Courses/CS364/2000S/` You may want to bookmark that page.

A number of important pieces of information are in the course web, including assignments, readings, requirements, syllabus, and office hours. I assume that if I put information on the Web, you will (eventually) read it.

- At the bare minimum, you should read all the pieces of basic information about the course. Of particular interest is the syllabus, which lists all the readings. I will also hand out a copy of this information on the first day of class.
- I prepare a rough outline for each class. Most students find these useful, and you should feel free to refer to them before, during, and after class. Since this class is becoming more lab-based, it is likely that in laboratory sessions there will be more information in the outlines than I will cover in class.
- Each outline begins with some notes. You can find just the notes in a separate news page.

At the top and bottom of every page are a series of links to important components of the course web. These are

- Instructions. This set of instructions.
- Search. A simple search facility for the course web.
- Current. The outline of the current or next class. You may need to reload the page to get the appropriate version.
- News. The course news, taken from the outlines.
- Syllabus. The course syllabus.
- Glance. An abbreviated version of the syllabus.
- Links. A collection of links that you might find useful.
- Handouts. Handouts for the class.
- Project. Information on the course project.
- Outlines. The outlines of classes that have been held. You can sometimes access other outlines through the syllabus.
- Labs. Laboratory assignments. When appropriate, we'll do labs in class (or as take-home assignments).
- Assignments. A list of the assignments for the class, accompanied by their due dates
- Quizzes. The quizzes and surveys for this course. I don't always grade these.
- Examples. A list of examples generated for this class.

# Administrative Information

## On Teaching, Learning, and Grading

- Introduction
- My Role
- Grading
- Your Role
- Lecturing
- Favoritism
- Summary

### Introduction

I like to begin each course with a metacommentary on teaching and learning. Why? Because I care about the learning process, because I seem to have a different teaching style and personality than some students expect, because I want you to think not just about *what* you are learning, but also *how* you are learning, and, unfortunately, because in one of the first two courses I taught at Grinnell some students were clearly dissatisfied with the way I teach. (As people are getting used to me and my teaching style, this seems to be less of a problem.)

From my perspective, you are here to learn and I am here to support that learning. What will you be learning? The subject matter of the course, certainly. However, I expect that (or hope that) you will also be discovering new ways to think and learn or sharpening existing skills. In terms of subject matter, I tend to care more about the processes and concepts that you learn than about the ‘‘basic facts’’.

Learning is an interactive process. You learn by asking, discussing, and answering questions, by playing with ideas (in computer science, you also learn by playing with programs), and by working with others. I know from experience that computer science cannot be learned passively: you need to experiment with ideas (in your head, on paper, or on the computer) in order to fully grasp these ideas.

### My Role

How do I try to support this learning? In a number of ways.

I *assign readings* to give you a basis for understanding the subject matter. Sometimes these readings will be from the textbook, sometimes I will distribute appropriate supplements.

I *lecture*, *lead discussions*, and *conduct recitations* on the topics of the course. Sometimes these will be based on readings and assignments, sometimes they will vary significantly from your readings. Why? Because I feel it wastes your time and mine to simply reiterate the readings. If you let me know that you’re confused about a reading, I will spend time going over that reading (either in person or in class).

To stimulate discussion and thinking, I regularly *call on students* in class. I know that not all of you are comfortable answering questions publically, but I strongly believe that you need to try. Please feel free to say ‘‘I’m not sure’’ when I call on you. At times, I’ll step through the class, asking each student in turn. At others, I’ll call on you individually. I tend to call more on students I interact with regularly.

I *assign work* because I find that most people learn by grounding concepts in particular exercises that allow them to better explore the details and implications of those concepts. I expect you to turn in work on the day it is due and will impose severe penalties on late assignments (including refusing to accept some late assignments).

Some of my assignments may involve *public presentation* of your work. Sometimes, the best way to learn a topic is to have to discuss it or present it to someone else. In addition, I've found that many students need some work on their presentation skills. Most often, presentations will be of papers that you've read.

**In general, I expect you to spend about ten hours per week on this class outside of class time.** If you find that you are spending more than that, let me know and I'll try to reduce the workload.

I *grade assignments* to help you identify some areas for improvement. Note that I believe that you learn more from doing an assignment than from receiving a grade on that assignment. This means that you may not receive a grade or comments on all your assignments. I will tell you when an assignment won't be graded, but not until after you hand it in. I will do my best to be prompt about returning grades on assignments. At times, I will use a grader to help speed the process.

I *give examinations* because I find that many students only attempt to master a concept when preparing for an exam. Because I care more about processes and concepts than about facts, I almost always give open-book examinations. Because I do not feel that time limits are helpful, this semester I will give you only take-home exams.

I *give quizzes* to ensure that you are doing the reading and that you are understanding what I expect you to understand from the readings and assignments. At times, I will give quizzes to help illustrate a particular point. This semester, all of my quizzes can only affect you positively: good work on quizzes will lead to extra credit.

I *build course webs* to organize my thoughts, to give you a resource for learning, and to help those of you who need to work on your note-taking skills. I do my best to make my notes for each lecture available on the Web, in outline format. In general, these notes will be available approximately five minutes before class. Warning: these are rough notes of what I expect to talk about; the actual class may not follow the notes. I will also attempt to update the notes after each class.

I *make myself available* to discuss problems and questions because I know that some of you will need personal attention. In general, if I'm in my office you should feel free to stop in. Most of the time, I'll be willing to help. Once in a while, I'll be working on a project and will ask you to come back later. Students always have first priority during office hours. You should also feel free to send me electronic mail, which I read regularly, and to call me. This semester, I am on partial parental leave, which means that I will be less available than normal. In particular, I will not be in the office on Tuesdays and Thursdays. Feel free to give me a call at home on those days, but understand that I may be busy.

At times, I *survey* my students to better understand how the class is going. Because I do research on the effects of computers on learning, I sometimes give surveys to gather data.

## Grading

At the same time that you learn and I try to help you learn, Grinnell and the larger community expect me to assign a grade to your work in the class. I base grades on a number of components, but primarily on *assignments*, *examinations*, and *involvement in classroom discussions*.

Because I understand that not everyone gets everything right the first time, I will occasionally allow you to *substitute* an extra assignment for one that you did poorly on. Unfortunately, the time pressures of the semester are significant enough that I will not be able to permit you to make up assignments except through this mechanism.

In computer science, it is often possible to do the same problem in multiple ways. Hence, I typically reserve class days to discuss particularly significant assignments. This semester, each exam will be followed by a day of discussion relating to the exam. We may also take time from some classes to discuss particulars of assignments.

I will admit to a fairly strict grading scale. Grinnell notes that A and A- represent exceptional work. To me, ‘‘exceptional’’ means going beyond ‘‘solid’’, correct work. Exceptional work entails doing more than is assigned or doing what is assigned particularly elegantly. Work limited to mastery of the core materials is B-level work. To help you demonstrate exceptional understanding, I will occasionally suggest *extra credit work* (although truly exceptional students will often suggest such work on their own).

To help eliminate biases, I typically use a numerical grading scale. 94-100 is an A, 90-93 is an A-, 87-89 is a B+, 84-86 is a B, and so on and so forth.

## Your Role

How should you participate as a member of my class? (Or, how do you do well in my class?) By being an active participant in your own learning. In part, this means doing all the work for the class. It also means a number of other things.

**Come talk to me** when you have questions or comments about subject matter, workload, or how the course is going in general. I will also set up an anonymous comment page for those who are uncomfortable talking to me directly.

**Do the readings** in advance of each class period and come prepared with a list of things that you don’t understand. I will try to spend time at the beginning of each class session answering these questions or will restructure the lecture to accommodate them.

**Ask and answer questions and make comments** during class periods. I consider active participation during class a particularly important part of the learning process.

**Begin your assignments early**. Students who begin assignments early have more opportunities to ask for help, to make sure that the assignment gets completed, and to sleep at night. Such students also do better in general.

### Lecturing

I seem to have a different ‘‘lecturing’’ style than some students expect. As I mentioned earlier, I don’t think it is the purpose of lecture to reiterate the readings. I do, however, think lecture and readings can provide alternate perspectives on the subject matter. At times, I will also discuss issues not covered in any readings.

I see no point in going on with a lecture or example if many students don’t understand what’s going on. You are the first line of defense: stop me when you are confused. In addition, I will occasionally stop the class and ask for a show of hands to see who is confused. Don’t be embarrassed to raise your hand; if you are confused, it is likely that someone else is also confused. I realize that this show of hands leads to some ‘‘pressure for understanding’’. However, you won’t get much out of a class if you’re confused (and therefore just copying down what I’m writing without thinking about it).

I deem it important for students to be active participants in lecture. This means that I will often ask you to help develop algorithms, solve problems, and even critique each other’s answers. If I call on you and you’re not sure of an answers, feel free to say ‘‘I don’t know’’ or to venture a guess. I consider it very important for all of us to see the problem solving process, warts and all. Note that I often generate examples of discussion ‘‘on the fly’’ so that we can all be involved in the problem solving or development process.

### Favoritism

For various reasons, I often get to know some students better than others. I tend to call on the students I know better, and sometimes respond slightly better to their questions because I have more context for the questions. I’m happy to make all of you ‘‘favorite’’ students. If you come to see me regularly and work enthusiastically on the material, you’ll probably end up being a student I know well.

### Summary

As the prior discussion suggests, I expect a great deal from my students. I also use many different strategies to get the best out of you. Feel free to discuss any of this with me (anything from concerns about this perspective to suggestions on improving teaching and learning).

## Academic Honesty

I expect you to follow the highest principles of academic honesty. Among other things, this means that any work you turn in should be your own or should have the work of others clearly documented. When you explicitly work as part of a group or team, you need not identify the work of each individual.

You should never ‘‘give away’’ answers to homework assignments or examinations. You may, however, work together in developing answers to most homework assignments. Except as specified on individual assignments, each student should develop his or her own final version of the assignment. On written assignments, each student should write up an individual version of the assignment and cite the discussion. On non-group programming assignments, each student should do his or her own programming, although students may help each other with design and debugging.

When working on examinations, you should not use other students as resources.

If you have a question as to whether a particular action may violate academic standards, please discuss it with me (preferably before you undertake that action).

### Citing Program Code

Note that computer programming shares with normal writing a need to cite work taken from elsewhere. It is certainly acceptable practice to borrow other code for your assignments. However, you must cite any code that you use from elsewhere. Each piece of code you take from elsewhere must include a comment that specifies:

- the author of the original code;
- the date the original code was written and the version of the code (if available);
- the date you incorporated the code into your program;
- a summary of the modifications (if any) you made to the code;
- instructions for getting the original code.

This applies not only to the code you get from the Web and elsewhere; it also applies to code you get from me and from the textbook.

You do not need to cite the classes and libraries you use, as the command to include classes and libraries within a program provides sufficient citation.

## Disabilities

I encourage those of you with disabilities, particularly ‘‘hidden disabilities’’ such as learning disabilities, to come see me about the accommodations that I can make to make your learning easier. If you have not already done so, you should also discuss your disability with academic advising. If you think you may have an undocumented learning disability, please speak to me and to academic advising.

In my experience, some learning difficulties can make computer science more difficult because of computers’ emphasis on small details. I also know that many of my favorite and best students have some learning disability, and have certainly succeeded. We’ll all do better if you talk to me about disabliities early.

Note that I generally feel that the ‘‘accommodations’’ that we are asked to make for those with learning disabilities are often appropriate for all students. Hence, I rarely give timed exams and I typically allow students to use computers during exams.

# CSC364 At A Glance

| | | |
|---|---|---|
| **Week 01: Background**<br>Reading: Peterson and Davie, Chapter 1 | | |
| Monday, January 24, 2000<br>**Introduction to the Course** | Wednesday, January 26, 2000<br>**Requirements and Network Architecture** | Friday, January 28, 2000<br>**Network Building Blocks** |
| **Week 02: C Programming**<br>Reading: K&R | | |
| Monday, January 31, 2000<br>**The Basics** | Wednesday, February 2, 2000<br>**Header Files and the Preprocessor** | Friday, February 4, 2000<br>**Pointers** |
| **Week 03: C Programming, Continued**<br>Reading: K&R | | |
| Monday, February 7, 2000<br>**Formatted I/O** | Wednesday, February 9, 2000<br>**Structs and Arrays** | Friday, February 11, 2000<br>**Memory Management** |
| **Week 04: Direct-Link Networks**<br>Reading: Peterson and Davie, Chapter 2 | | |
| Monday, February 14, 2000<br>**Framing** | Wednesday, February 16, 2000<br>**Error Detection** | Friday, February 18, 2000<br>**Common Structures** |
| **Week 05: Packet Switching**<br>Reading: Peterson and Davie, Chapter 3 | | |
| Monday, February 21, 2000<br>**Switching and Forwarding** | Wednesday, February 23, 2000<br>**Bridges and Switches** | Friday, February 25, 2000<br>**ATM** |
| **Week 06: Internetworking**<br>Reading: Peterson and Davie, Chapter 4 | | |
| Monday, February 28, 2000<br>**IP** | Wednesday, March 1, 2000<br>**Routing** | Friday, March 3, 2000<br>**Multicast** |
| **Week 07: Pause for Breath**<br>Reading: None | | |
| Monday, March 6, 2000<br>**Review for Exam** | Wednesday, March 8, 2000<br>**Examination** | Friday, March 10, 2000<br>**Special Topic** |
| **Week 08: End-To-End Protocols**<br>Reading: Peterson and Davie, Chapter 5 | | |
| Monday, March 13, 2000<br>**TCP** | Wednesday, March 15, 2000<br>**TCP, Revisited** | Friday, March 17, 2000<br>**RPC** |
| *Break* | | |
| **Week 09: Programming Unix Sockets**<br>Reading: Stevens, Parts 1 and 2 | | |
| Monday, April 3, 2000<br>**Socket Basics** | Wednesday, April 5, 2000<br>**Simple Client/Server Pairs** | Friday, April 7, 2000<br>**Handling Multiple Clients** |
| **Week 10: Advanced Unix Sockets**<br>Reading: Stevens, Part 3 | | |

| | | |
|---|---|---|
| Monday, April 10, 2000<br>**Nonblocking I/O** | Wednesday, April 12, 2000<br>**Broadcasting and Multicasting** | Friday, April 14, 2000<br>**Signals and Threads** |
| **Week 11: Allocating Resources**<br>Reading: Peterson and Davie, Chapter 6 | | |
| Monday, April 17, 2000<br>**Sharing Resources** | Wednesday, April 19, 2000<br>**Congestion Control** | Friday, April 21, 2000<br>**Quality of Service** |
| **Week 12: Network Security**<br>Reading: Peterson and Davie, Chapter 8 | | |
| Monday, April 24, 2000<br>**Cryptographic Algorithms** | Wednesday, April 26, 2000<br>**Security Mechanisms** | Friday, April 28, 2000<br>**Firewalls** |
| **Week 13: Internet Protocols**<br>Reading: Peterson and Davie, Chapter 9 | | |
| Monday, May 1, 2000<br>**Domain Name Service** | Wednesday, May 3, 2000<br>**FTP** | Friday, May 5, 2000<br>**Mail Protocols** |
| **Week 14: Wrapup**<br>Reading: None | | |
| Monday, May 8, 2000<br>**HTTP** | Wednesday, May 10, 2000<br>**Course Evaluation** | Friday, May 12, 2000<br>**Wrapup** |