

CSC 458: Computer Networks, Fall 2009

Department of Computer Science, University of Toronto

Handout # 1 – Information Sheet

Date: Tue, September 15th

Class hours: Tue. 1-3 PM

Location: BA 2145

Instructor: Professor Yashar Ganjali

E-mail: yganjali@cs.toronto.edu

Office: BA 5238

Office hours: Tue. 3-4 PM,
or by appointments

Tutorials: Fri. 1-2 PM

Location: BA 1200

Teaching Assistants:

- Wesley George (wgeorge@cs.toronto.edu)
- Alireza Sahraei (sahraei@cs.toronto.edu)

Final Exam: TBA

Class web page:

<http://www.cs.toronto.edu/~yganjali/courses/csc458/>

Course Description

Computer communication network design and implementation. Packet switching systems; socket programming; network software, hardware, and protocols; network naming and addressing; congestion control schemes; traffic generation, and measurement; network security; wireless networks. Emphasis on programming and experimental analysis of real network components.

Prerequisites

You need to have a basic understanding of probability theory, a strong background in C programming, and familiarity with the Unix operating system. If you are not sure whether you have the background to take this course, please take a look at the first programming assignment (link available at the class web page) to get an idea of the type of work, and the amount of time you will need to spend on it. If you still are not sure, send me an e-mail.

Teaching Assistants

Please check class web page for information about TAs, and their office hours.

Bulletin Board and Class Mailing List

Please use the bulletin board to ask questions from TAs. By using the bulletin board, everyone in class can read the replies and the overall number of repeat questions is reduced. Please check the bulletin board before posting any new questions. We guarantee any question posted to the bulletin board will be responded within 48 hours.

There is also a class mailing list which will be used by the TAs and the instructor for announcements. *Please never use the class mailing list for questions.* There are many students in class and we'd all be flooded with e-mails.

If you have any questions that cannot be posted on the bulletin board (e.g. questions about your grades), you can e-mail TAs directly. There is no guarantee on when you will get a reply. We really want you to use the bulleting board. :-)

Exams

For undergraduate students, there will be an in class midterm exam on October 27th as well as a final exam. For date, and location of the final exam, please check the class web page. Graduate students enrolled in this course are required to do a project instead of midterm and final exams. The deliverables for the project are a one page proposal, an intermediate report (two pages), and a final report (six pages). I will provide a list of interesting topics for the project, or you can come up with your own project within the scope of this course.

Textbook

- “*Computer Networking: A Top-Down Approach*”, Kurose and Ross, Addison Wesley, 5th edition.

Recommended Books

- “*UNIX Network Programming, Volume 1: The Sockets Networking API*”, W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, 3rd edition, 2003.
- “*TCP/IP Illustrated, Volume 1: The Protocols*”, W. Richard Stevens, 1993.

Notes and Handouts

I will use a combination of slides and blackboard. Please take notes when I am using the blackboard. The handouts will be provided to you so you don’t need to take notes.

Grading for Undergraduate Students

- Assignments: 50%
 - Problem sets: 20%
 - Programming assignments: 30%
- Midterm exam: 20%
- Final exam : 30%

Grading for Graduate Students

- Assignments: 50%
 - Problem sets: 20%
 - Programming assignments: 30%
- Final project: 50%
 - Proposal: 10%
 - Intermediate report: 15%
 - Final report: 25%

Assignments

There will be two problem sets, both based on the textbook, and the material covered in class. There will also be two programming assignments.

Here is a summary of the requirements for all submitted programming assignments:

- All programs must be written in ANSI “C”. To make the grading uniform, we can’t except assignments in C++, Java, perl,
- All programs must compile and run correctly using *gcc* on the CDF machines.
- You are strongly encouraged to use *gdb* to debug your programs.
- Additional requirements will be specified in each assignment.

Late Submission Policy

You have *one* free late submission of 24 hours for one of the assignments (problem set, or programming, but not both). You should e-mail the TAs before the deadline to get the free late submission. This 24 hour limit is hard, and cannot be extended. For any late submission other than the free one, 10% of the mark will be deducted for each day late, up to 20%. Assignments will not be accepted after two days.

Academic Offenses

“Briefly, an academic offence is a bad thing done to get marks you don't deserve. Slightly more formally, an academic offence is an action by a student or course instructor that breaks the rules about academic credit at the University of Toronto.”¹ Cheating is considered a very serious offense. Please avoid it! We are all here to teach and learn after all, and concerns about cheating make an unpleasant environment for everyone.

Permitted Collaboration

The following items are encouraged and allowed at all times for all students in this class:

- Discussion of material covered during lecture, problem sessions, or in handouts
- Discussion of the requirements of an assignment
- Discussion of the use of tools or development environments
- Discussion of general approaches to solving problems
- Discussion of general techniques of coding or debugging
- Discussion between a student and a TA or instructor for the course

Collaboration Requiring Citation

Two students engaging in more detailed discussions must be careful to document their collaboration. Students are required to include the names of those who provide specific assistance to properly credit their contribution, in the same manner as one would cite a reference in a research paper. The expectation is that even with a citation, the author must be able to explain the solution.

Examples of Collaboration That Require Citation

- Discussing the “key” to a problem set or programming assignment. Problem set questions are often designed such that the critical concept takes careful thought and gaining that insight from someone else must therefore be documented.
- Discussing the design of a programming project. Design is a crucial aspect of the programming process and discussion can be valuable. Any design input received from others must be cited.
- Receiving assistance from another student in debugging code. While the TAs are the preferred source for advice, any detailed assistance from someone else must be credited.

¹ Jim Clark, “Advice about academic offenses”, <http://www.cs.toronto.edu/~clarke/acoffences/>.

- Sharing advice for testing. For example, if someone provides important information on lessons learned ("my program didn't handle the case where the value was 0") that source must be credited.
- Research from alternative sources. Researching related topics, such as through the Internet, must be documented if the solution submitted is derived from the research information.

Unpermitted Collaboration

All submissions must represent original, independent work. Some examples of activities that do not represent original work include:

- Copying solutions from others. In particular, do not ask anyone to provide a copy of his or her solution or, conversely, give a solution to another student who requests it. Similarly, do not discuss algorithmic strategies to such an extent that you and your collaborator submit exactly the same solution. Use of solutions posted to websites, such as at other universities, is prohibited. Be aware that we photocopy some of the exams prior to handing them back.
- Using work from past classes. The use of another student's solution or the posted class solutions from a previous class constitutes a violation.
- Studying another student's solution. Do not read another solution submission whether in electronic or printed form, even to "check answers."
- Debugging code for someone else. When debugging code it is easy to inadvertently copy code or algorithmic solutions. It is acceptable to describe a problem and ask for advice on a way to track down the bug.²

² Parts of this note are based on handouts from Nick McKeown, and Tom Fountain, who teach CSC244a and EE182 respectively at Stanford. Some portions are based on similar collaboration policies written by Eric Roberts, Julie Zelenski, and the Computer Science Department at Brown University.