CSC 458 Computer Networks, Fall 2017

Department of Computer Science, University of Toronto
Course Information Sheet

Section: L0101	Section: L0201	Section: L5101
Location: MP 134	Location: GB 220	Location: GB 303
Lecture: Thurs, 1-3pm	Lecture: Tues, 1-3pm	Lecture: Tues, 6-8pm
Tutorial: Fri, 11-12, MP134	Tutorial: Fri, 1-2, GB220	Tutorial: Tues, 8-9, GB303
Instructor: AbdulAziz Al-Helali	Instructor: Sajad Shirali-Shahreza	Instructor: Joe Lim
Email: a.alhelali@mail.utoronto.ca	Email:shirali@cs.toronto.edu	Email: joe.lim@utoronto.ca
Office: BA3219	Office: BA3219	Office: BA3219
Office Hours: Thurs,3:15-5:15	Office Hours: Fridays, 2-3	Office Hours: Tues, 4-5:30pm

Class Web Page: www.portal.utoronto.ca

Course Description

This is an introductory course on computer networks. Topics covered in this course include packet switching systems, socket programming, network software, hardware, and protocols, network naming and addressing, congestion control schemes, software-defined networking, network security, and wireless networking. The emphasis of the course is network programming and applications.

Prerequisites

CSC209H1, CSC258H1, CSC263H1/CSC265H1, STA247H1/STA255H1/STA257H1/ECO227Y1

You need to have a basic understanding of probability theory, a strong background in C, a good understanding of Python, 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 your instructor an e-mail.

Teaching Assistants

Information will be posted once the schedule is set.

Bulletin Board and Class Mailing List

Please use the bulletin board https://bb-2017-09.teach.cs.toronto.edu/c/csc458 to ask questions. 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 no guarantee on when you will get a reply to e-mails. We really want you to use the bulletin board. :-)

If you have any questions that cannot be posted on the bulletin board (e.g., questions about your grades), you can e-mail the instructors directly.

Exams

Midterm Exams will be on ***Tuesday, Oct 17, 2017 from 6:00 - 8:00pm***. This applies to ***all three sections*** of this course. If you have a conflict with this date and time, please **notify** your instructor as soon as possible, ***before September 30th***. There is also a final exam. For date, and location of the final exam, please check the Arts & Science website. As soon as we get the exam timetable, we will post it on Blackboard.

Textbook

• *"Computer Networks: A Systems Approach"*, (5th Edition), Peterson, Davie, 2011.

Recommended Books

• "*UNIX Network Programming, Volume I: 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

We will use a combination of slides and blackboard. Please take notes when we are using the blackboard. The handouts will be available on Blackboard, so you don't need to take notes.

Grading

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

Assignments

All assignments will be posted by the 2nd week of class with their deadlines. You should manage your time efficiently.

There will be two problem sets (PS1 and PS2), both based on the textbook, and the material covered in the class.

There will also be two programming assignments. The two assignments are related in that you will need your PA1 code for PA2. We will not make any solutions available. You will need to fix your PA1 code based on the feedback you get before you can start working on your PA2. Here is a summary of the requirements for all submitted programming assignments:

- We will use a virtualized network environment (MiniNet) for programming assignments.
- Each student will get access to a virtual machine (on CDF servers) where she/he can complete the programming assignments.
- To ensure compatibility with our marking scripts, please make sure you only use the VM provided through CDF.
- All programs must be written in ANSI "C". No other programming languages are accepted.
- Additional information and requirements will be specified in each assignment.

Late Submission Policy

2.5% per 6 hours of the mark will be deducted past due date, up to 20%. Assignments will not be accepted after two days.

Academic Offenses "Briefly, an academic offense is a bad thing done to get marks you don't deserve. Slightly more formally, an academic offense is an action by a student or course instructor that breaks the rules about academic credit at the University of Toronto." 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.
- 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.