CS 485/ECE 440/CS 585 Fall 2013 Lab 1

Due 11:59pm on Thursday, 31 October 2013

Please submit your writeup for lab 1 as a PDF attachment of an email to "unmnetworkingclass@gmail.com". Do not submit your lab writeup to any other address.

Lab 1 is worth 200 points, based on the following rubric:

- 100 points for the technical aspects. Did you teach your classmates and I something interesting about network sockets and TCP/IP performance? Was what you did technically sound? Are your conclusions justified?
- 40 points for your experimental methodology and the related parts of the writeup. Did you take steps to avoid bias and distortion? Were you measuring the right thing? Did you account for all the different kinds of variation that might affect your results? Did you use the terminology correctly in the writeup? Could someone read your writeup and repeat all of your experiments to get the same data and draw the same conclusion for themselves? Think *repeatability*.
- 20 points for the presentation of your writeup. Did you use correct English grammar? Was your writing concise and readable? Did you use good visuals? Did you choose a meaningful title and remember to put your name on the first page?
- 20 points for your statistical analysis. Did you use the terminology correctly? Were the statistical tests you did based on the right assumptions?
- 20 points for a "poster" presentation, where you'll all post your writeup with the visuals on the wall and then people will walk around and discuss your results. I won't grade your presentation, I'll just give 20 points for participation if you take part in this or 0 if you don't. We'll probably do this in two halves in the first week of November, *e.g.*, on Tuesday the 5th half the class will present and half will be the audience, and on Thursday the 7th the halves will switch.

Your writeup can be at-most 2 pages of text in a reasonable font size. Figures and tables won't count towards the 2 pages, so put as many figures and tables as you want. Your writeup should include at least the following sections: abstract, experimental setup, results and discussion, and conclusion.

You are expected to do your own work, from implementation to experimental design to collecting data to doing the writeup, for all phases of this project you should do your own work. For the writeup, if you copy even a single sentence from an existing source without clearly attributing it to the correct authors, you will receive a 0 on this assignment. Also, I may ask you to reproduce a subset of the results for me using your own virtual machines. If you're not sure whether something will be considered cheating or not, ask me before you do it. Every student needs to do their own unique experiments. That's one good reason to run your ideas by me, to make sure you're not doing the same

experiment as one of your classmates, in which case I'll ask the two of you how you'd like to split up the total points out of 200 that you got.

You should use your own socket code. You can start from some basic socket C code downloaded off the Internet and build on top of it for the experiment you want to do. Don't use things like httperf, I want you each to have your own source code for the client and server sockets. Your experiments should be about TCP/IP sockets. If you want to compare TCP to UDP (*e.g.*, for streaming media), that's okay, but make sure what you're doing involves TCP/IP stream sockets that you're actually sending data over.

Using the virtual machine setup you've created, you should run some experiments to teach your classmates and I something really interesting about TCP/IP sockets and related performance, reliability, or security issues. You should have at least two factors in your experiment. You should follow the practices for experimental design and statistical analysis that we've gone over, and use the related techniques and terminology correctly. I'm expecting excellence in all aspects of this project, including the presentation and English grammar of the writeup and that your results actually teach us all something interesting we didn't know before. For this reason, I strongly recommend that you get started early and get your first round of results within the next week. Then you can either do another round of experiments to make the results more interesting or start on the writeup to make sure you have time to make it the best writeup you've ever written. Each of you already has a virtual machine setup in place, so getting results within the first week should be very doable.

Feel free to modify the virtual machine network setup if you like. For example, you can add more clients, another router or two, incorporate the Internet into your measurements, or whatever you like. Talk to me before you start routing packets from your experiment out to the Internet, though.

I strongly recommend that you add tc traffic shaping rules at different parts of your network. Without bandwidth limitations, delay, packet loss, *etc.* TCP is not very interesting.

I also strongly recommend that you bounce your ideas for an experimental setup and hypothesis off of me and I'll try to give you some feedback. You should also bounce your ideas off of the TAs, a couple of classmates, *etc.* Make sure you choose something fun that you're truly interested in. If you're having trouble thinking of interesting questions to test, come talk to me.

Here are some random thoughts to help you get your own thinking started...

- You could compare two different congestion control implementations, *e.g.*, Reno *vs.* CUBIC.
- There are all kinds of TCP, IP, and socket options you can look into, like selective ACKs, TCP window scaling, TCP_DEFER_ACCEPT, TCP_QUICKACK, TCP_CORK, Explicit Congestion Notification, *etc.* Try “man setsockopt” and “man 7 tcp”.
- The SYN backlog of Linux and the SYN cache of FreeBSD are both pretty interesting, but

mostly relevant to half-open connections. Still, if you're interested in security come talk to me and maybe we can come up with something interesting in this vein.

- Things you can control/vary with tc rules: delay, packet loss, packet reordering, and bandwidth. There are also some more advanced options like traffic shaping for fairness or to restrict a certain kind of traffic to smaller bandwidth, plus there are lots of option to play with for everything, like burstiness for token bucket filters, different statistical distributions for packet loss and delay that can be correlated and so on, and other ways to simulate interesting networks.
- Read as much as you can about tc, it can do some cool things like Random Early Detection that are relevant to Internet censorship (try using gmail in China, for example).
- The FreeBSD machine can be either the server or the client in your experiments, or you could make FreeBSD vs. Linux one of your factors, or you could even exploit the fact that TCP/IP connections are full duplex.
- You could try making a multi-process or multi-threaded server implementation, or even try playing with the poll() or select() API if you want to challenge yourself and do some serious UNIX networking programming.
- Optimistic acknowledgments come to mind as another potential security-related project.
- In Linux, do an ls on /proc/sys/net/ipv4 for a list of parameters in the Linux kernel you could play with. On FreeBSD do "sysctl -a".
- Wireshark can make various kinds of pretty graphs that you might find interesting, both for brainstorming what you'd like to do experiments on and also for making visuals for your writeup.
- You could maybe play around with injecting RSTs with some variable delay and/or reordering and see how easy/hard it is to reset live connections. See:

  http://www.cs.unm.edu/~crandall/icdcs2010.pdf
- I recommend keeping things simple and doing a two factor full factorial experimental design with two levels per factor. If you collect 30 data points per treatment, that should be plenty.
- You should be getting my feedback throughout the process, including discussing your ideas for what kind of experiments you want to do, discussing your experimental design, discussing the results, having me read drafts of your writeup, *etc.* It's also okay to discuss your project with other people, either in the class or people not in the class, just as long as the work you do is your own. You can also get feedback from the TAs.
- One approach would be to write some socket code and then just start playing with things like socket options and tc rules until you come across something really interesting that you'd like to explore further.
- Another approach would be to email me (or talk to me) about what your interests are. For example, if you're into web programming and Ruby on Rails and all that jazz, we could probably come up with something that would teach you more about your weird passion for HTTP. (Personally, my fetish is layer 3 network stack implementation details for commodity OSes and routers, like fragment reassembly policies, and a little bit of layer 4 half-open connection stuff. It would be hard to design a TCP/IP socket experiment about that kind of thing, but I'm happy to try if you're interested).