CS 485/ECE 440/CS 585 Fall 2017 Lab 3, part 2

Lab 3 part 2 is due by 11:59pm on Thursday, November 9th, 2017. Part 2 is worth 100 points, and part 1 was worth 100 points, so in total Lab 3 is worth 200 lab points.

You'll submit a single packet capture (PCAP) and two pieces of source code as attachments in an email to crandall@cs.unm.edu. Failure to follow these instructions will result in a zero on the assignment. Send real email attachments, when you send "SharePoint" or Google Drive links or whatever, I won't click on those links. Unlike part 1, both group members (you'll work in groups of 2 for this assignment) need to submit something, but cc your partner and put both of your names in the body of the email. So, I'll get two emails from each pair of partners. In your email should be one PCAP (it should match your partners, *i.e.*, be the exact same PCAP your partner also submits), one file that represents your client implementation, and one file that represents your server implementation. Source code files should be single files and should either be gzipped tar balls or single text files. For example, you can submit a *.c file, a *.py file, a *.tar.gz, a *.tgz, or whatever as long as either opening it in a text editor or running "tar xvzf ..." on it works, and each of the client and server are clearly named (*e.g.*, tictactoeclient.py and tictactoeserver.py) and represented by exactly one file.

You are expected to do your own work. From writing your socket code to capturing the PCAP, for all phases of this project you should do your own work. Any instance of not doing your own work will be considered cheating. Submitting source code that doesn't work and/or is not what you used for creating the PCAP with your partner will be considered cheating. If you're not sure whether something will be considered cheating or not, ask me before you do it. You are encouraged to discuss the assignment with your classmates (especially your partner) at a any level of abstraction you like, so long as three things are true: 1) nobody else but you is typing on the keyboard or doing anything to configure your VMs; 2) you're not typing anything or making any changes that you don't understand; 3) You and you alone write the source code for your socket implementations (client or server). As long as those three things are true, feel free to help each other out and give pointers for debugging problems with each other's source code. Exchanging tools, source code that existed before the assignment was assigned, and general thoughts about approaches to specific problems is okay. As a reminder of the course policy, if you cheat on any assignment in this class including this assignment (cheating includes, but is not limited to, representing somebody else's work as your own or having someone else do the assignment for you) you will receive an F in the class. If you want to share source code written for the assignment with a classmate, you should get my permission first and share it with the whole class.

For Lab 3 part 2, you'll each be writing a client and a server to play tic-tac-toe. You can make up whatever protocol you want, but it should be obvious to someone who follows the TCP stream of a single tic-tac-toe game what moves were made, and the result of the game (tie, or who won, basically). YOU AND YOUR PARTNER MUST IMPLEMENT YOUR CLIENTS AND SERVERS IN COMPLETELY DIFFERENT LANGUAGES. For example, you can write your client and server in Python and your partner can write theirs in C. You can use any language you like, but no sympathy from me if you choose languages that make side effects hard (like Haskel). The language you choose must be clearly distinct from the language of your partner. For example, C and C++ are not different enough to be considered two different languages. But don't worry about what languages other students who aren't your partner choose, so long as you write your own source code and don't cheat in any way.

You can use any code that existed before this assignment was assigned so long as you credit where it came from (a website, a student who took this class in the past, or whatever) clearly in a comment at the top of the source code file. For any language you choose, there is probably existing TCP/IP client

and server socket code examples on the Internet, and you should feel free to use those as a starting point for your own source code *so long as you understand what's going on and how your program will use the Linux socket API to listen for or make connections*. If you copy socket code without understanding what it's doing, you're setting yourself up for failure on the second midterm. You can also use whatever source code existed before the assignment was assigned for the tic-tac-toe logic.

You and your partner can make up whatever protocol you like for the tic-tac-toe game between client and server, subject to the following constraints:

1. The client should make the first move uniformly at random (i.e., choose one of the 9 spaces completely randomly), and be X while the server is O.
2. It should all be done in printable ASCII characters, and be obvious to a human looking at a TCP stream what moves were made and who won (or if there was a tie).
3. When it's machine *vs.* machine (which it will be for your PCAPs), the result of the game should always be a tie.
4. The server always listens on port 707.

For example, your TCP stream might look like this where green is client, red is server, '\n' is a newline, and the three rows are three rows of a tic-tac-toe game:

---snip---
X--|---|---\n
XO-|---|---\n
XO-|X--|---\n
XO-|X--|0--\n
XOX|X--|O--\n
XOX|X-O|O--\n
XOX|X-O|OX-\n
XOX|XOO|OX-\n
XOX|XOO|OXX\n
tie\n
tie\n
---snip---

Feel free to completely steal the above protocol, it'll be guaranteed to be easy for me to see what's going on if you do.

To create a PCAP to prove that your socket code (both client and server for both you and your partner) works properly for turning in, you should create the PCAP on either one of your eth3's on the middle router. You should never, for any reason, run your partner's or any other classmate's source code on your virtual machines or let anyone run your source code on their virtual machines, this will be considered cheating. You should have your outside machines (groucho and gummo for me) be the tic-tac-toe servers and your inside machines (harpo and zeppo for me) be the tic-tac-toe clients, and make sure your clients connect to your partner's servers and vice versa. Your single PCAP that you submit should contain at least twenty tic tac toe games **PER** pair of clients/servers that cross the interface between you and your partner. So each of your clients with each of their servers and vice versa is eight possibilities, and there should be at least 20 tic-tac-toe games per each of those eight possibilities for 160 total (at least). Your PCAP should be less than 1 MB.

All PCAPs might be shared with the entire class later in the semester. So, make sure no private data makes its way into your PCAPs somehow. My hope is that Lab 4 will be something about TCP congestion control using the entire class as one big network, and that Lab 5 will be a "do whatever tomfoolery you want short of breaking into other people's VMs" lab.