Computer Networks Theory + Lab

Summary

Contents

« Previous (Lab07: IP Packet Format, DHCP and ICMP) Next (Lab09: DNS and Web) »

Lab08: Instructions

Lab08: Socket and TCP Basics

Objective:

1. Learn ss (netstat) to collect socket statistics and netcat for reading/writing to sockets
2. Learn how TCP works via interpreting tcpdump based traces

Reference Material:

1. SS: https://computingforgeeks.com/netstat-vs-ss-usage-guide-linux/
2. Netcat: https://sansorg.egnyte.com/dl/Rop1b0ElWo; https://jameshfisher.com/2018/12/31/how-to-make-a-webserver-with-netcat-nc/ (fundamentals and file transfer are the only headings of interest in the first URL, the next URL shows how to set up a webserver)

Part-1: SS / Netstat and Netcat

Go through the references, understand the commands and play around with SS. Note that SS supersedes netstat, so you can focus just on it (MAC users may need to use netstat though). Generate your own tcp/udp traffic and check that these reflect in the ss command. For tcp, browse around, open teams, email apps etc and for udp, you can use something you have seen before, starts with a "t" :-)

Windows users, SS does work inside wsl, but you need to generate traffic internally to see any action. MAC users need to use netstat instead of ss.

Any one having any issues, can always visit sl2 lab or even ssh into one of the machines and play around (e.g. sl2-15.cse.iitb.ac.in)

Play around with netcat also. Set up a web server (we are yet to cover HTTP for you to understand all details, but give it a shot) and also do a file transfer locally (between two processes on localhost). Verify that these are correctly captured by ss.

For auto-grading purposes, we share below a simple set-up and corresponding files. Based on the provided files, attempt the quiz titled "Lab08-SS". Like before, try this experiment on your machine, though we are not going to grade it.

Scenario 1: No browser is open (in fact, all other applications are closed) Scenario 2: The chrome browser is open, but nothing typed Scenario 3: google.com is entered in the chrome browser

You are given three sets of three output files corresponding to commands "ss -s, ss -lt, ss -tp" for each of the above scenarios. In addition to the above, there are two other files, ss-tcp.txt and ss-udp.txt, which will be used in the quiz titled "lab08-ss".

Part-2: Real-life TCP

Real-life TCP is quite different from what we covered in class. This is not to say the core concepts are not present - slow start, AIMD, RTT estimation, etc., are very much there. Except that a few other optimizations have been added on top. We will explore them in this lab. The answer descriptions will provide more details about the optimization once the lab ends (you are not required to understand these in depth, just observe them as part of the lab).

You may want to first check which version of TCP you are using on your laptop. For this do "cat /proc/sys/net/ipv4/tcp_congestion_control" in Linux.

To understand TCP, one needs insight into the sending side, where most of the action is. For this, you need to upload a file."scp", "google-drive uploads", etc., come with a lot of security-related stuff which can distract you. While you can give them a shot (in fact you should, to see the difference), it will be cleaner if you upload a file to an HTTP-based website and trace the flow. You can try "http://www.csm-testcenter.org/" for this, select file upload and you can use it to upload a reasonably sized file, say 1MB.

For a cleaner trace, don't start tcpdump until you get to the point where you just have to click the upload button. After collecting the trace, open it in wireshark to examine it.

You should do a bit of further clean-up. For this, in wireshark, select what you think is a packet belonging to your file upload (upload a reasonably large file like 1MB and you will know what packets belong to this upload). Right-click and select "follow TCP stream". This isolates all packets belonging to this TCP connection. Focus on this connection henceforth.

As mentioned, some of the behavior you will see is very different from what was covered in class. Feel free to browse around to understand the reasons for such behavior. Note that if you try the same on your desktop/laptop, you may see very large segment sizes much beyond Ethernet MTU of 1500. This is because of a feature called TCP segment offloading. You can use ethtool to disable this feature before trying it out. (See https://forum.ivorde.com/linux-tso-tcp-segmentation-offload-what-it-means-and-how-to-enable-disable-it-t19721.html)

For statistics, in wireshark, under statistics, checkout TCP stream graphs. Look at this to understand some of the graphs generated by wireshark. http://packetbomb.com/understanding-the-tcptrace-time-sequence-graph-in-wireshark/

While you should collect your own trace and analyze it, we provide a trace upload.pcap, based on which answer the quiz titled "lab08-tcp". The version of TCP used here is TCP cubic. Be sure to filter the trace to identify the right connection (Right-click a relevant packet and select "follow TCP stream".)

View Document

Discussion Forum

Search here

Threads : 0

Create thread

Recent Earlier Popularity Importance Instructor