Administrivia:
NO CLASS Next week, Feb 19, 2020

…Since last class ran out of time…
Quick recap, and then picking up where we left off:

For the network side of things, you want to look at things like:

- Connections
- Traffic (bps, pps)
- Packet loss / network routing

You'll want to use 'netstat' to check connections: are there more than expected? Are they lingering? Is a certain host trying to connect more than expected (malicious? unintentional?)? Check for sockets in SYN_RECV state, etc. One of the often-forgotten options for netstat is "`-s`", which gives you a summary of a LOT of useful IP, TCP, and UDP data. Here's an excerpt of the "TCP" section:

```
Tcp:

    6904 active connections openings
    55223 passive connection openings
    156 failed connection attempts
    4924 connection resets received
    6 connections established
    2057050 segments received
    1530644 segments send out
    1335 segments retransmited
    254 bad segments received.
    93295 resets sent
```

Some of the more useful numbers on the debugging side would be "failed connection attempts" and the "resets" info (well, and "bad segments", too :)). The "failed attempts" is a measure of how many times the local machine tried to connect to something else, and failed. Here's an example of how that increments:

```
root@ka:~# netstat -s | grep "failed connection attempts"
    156 failed connection attempts
root@ka:~# telnet localhost 9999
Trying ::1...
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
root@ka:~# netstat -s | grep "failed connection attempts"
    158 failed connection attempts
```

Even though I only attempted one telnet connection, the count incremented by two, because it first tried to connect to the IPv6 address (`::1`), and then to the IPv4 address (`127.0.0.1`).

Measuring actual traffic is a little harder with "standard" tools, but hopefully you can either install useful ones, or write some scripts to do the work for you. Strictly measuring interface speeds can be done with a script that looks at `/proc/net/dev` and calculates the difference in the numbers.

Programs like `iftop` and `iptraf` are a little more useful as they can show you exactly what connections there are, and which ones are doing the most traffic, etc. (These are both a little hard to show examples of, as they are interactive programs).

The program `tcpdump` is a bit more "raw" than `iftop/iptraf`, but that can be useful at times, to see actual packet-level information.

One other thing that is important is to see what the network looks like between you and your destination (assuming you're doing remote-host traffic). The most common utilities for this sort of thing are `traceroute` and `mtr` (mtr is often not installed by default). They both work similarly (sending packets with increasing TTLs [Time To Live] to determine the route), and the output is somewhat the same:

```
traceroute to turbo (69.36.231.41), 30 hops max, 60 byte packets
 1  23.92.24.2 (23.92.24.2)  0.473 ms  0.641 ms  1.017 ms
 2  10ge7-6.core3.fmt2.he.net (65.49.10.217)  0.289 ms  0.302 ms  0.301 ms
 3  10ge5-4.core1.sjc1.he.net (184.105.80.193)  0.896 ms  0.899 ms  0.896 ms
 4  layer42.net.any2ix.coresite.com (206.51.41.5)  0.976 ms  1.017 ms  1.208 ms
 5  turbo.boom.net (69.36.231.41)  10.125 ms  10.153 ms  10.152 ms
```

Compare this to a trace from SDSC to that same host:

```
traceroute to turbo.boom.net (69.36.231.41), 30 hops max, 38 byte packets
 1  ge2.lowpass.boom.net (132.249.239.201)  0.236 ms  0.187 ms  0.135 ms
 2  SDSC-gateway-vrrp1.sd.boom.net (132.249.239.194)  1.421 ms  3.212 ms  0.901 ms
 3  bwctl9.sdsc.edu (192.12.207.9)  1.236 ms  0.503 ms  1.035 ms
 4  dc-sdg-agg4--sdsc-10g-2.cenic.net (137.164.23.181)  0.953 ms  0.926 ms  0.832 ms
 5  dc-tus-agg3--sdg-agg4-100ge.cenic.net (137.164.11.8)  2.468 ms  2.525 ms  2.433 ms
 6  dc-lax-agg6--tus-agg3-100ge.cenic.net (137.164.11.6)  3.411 ms  3.268 ms  3.282 ms
 7  laiix.layer42.net (198.32.146.22)  3.745 ms  3.435 ms  3.279 ms
 8  xe2-3.core1.lax.layer42.net (65.50.198.213)  3.449 ms  3.361 ms  3.237 ms
 9  turbo (69.36.231.41)  3.045 ms  3.330 ms  3.381 ms
```

mtr: The same route, one as IPv4, the other IPv6:

```
                                              Packets               Pings
HOST: devnull                               Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 23.92.24.2                            0.0%    10    0.4   0.5   0.4   0.6   0.1
  2.|-- 10ge7-6.core3.fmt2.he.net             0.0%    10    5.4  13.0   0.3  66.4  19.2
  3.|-- 10ge5-4.core1.sjc1.he.net             0.0%    10    3.2   9.1   0.8  20.9   6.1
  4.|-- layer42.net.any2ix.coresite.com       0.0%    10    1.0   7.6   0.9  43.3  14.1
  5.|-- turbo.boom.net                        0.0%    10   10.8  10.2  10.0  10.8   0.3

HOST: devnull                                 Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 2600:3c01::8678:acff:fe0d:a641          0.0%    10    0.7   0.9   0.7   1.7   0.3
  2.|-- 10gigabitethernet8-8.core3.fmt2.he.net  0.0%    10    0.3   6.2   0.3  37.8  11.4
  3.|-- 10ge5-4.core1.pao1.he.net               0.0%    10    1.0   1.5   1.0   3.4   0.8
  4.|-- ipv6-paix.layer42.net                   0.0%    10    2.0   2.6   1.6   8.0   2.0
  5.|-- xe4-6.core2.mpt.layer42.net             0.0%    10    1.5   1.5   1.4   1.6   0.1
  6.|-- xe0-3-0-0.core4.sv1.layer42.net         0.0%    10    2.3   2.2   2.1   2.4   0.1
  7.|-- xe0-2-0-1.core2.lax.layer42.net         0.0%    10   10.3  10.1  10.0  10.3   0.1
  8.|-- xe2-3.core1.lax.layer42.net             0.0%    10    9.9  10.0   9.7  11.6   0.6
  9.|-- 2001:1868:a900:0:10::1                  0.0%    10   10.4   9.8   9.4  10.4   0.3
```

There are a couple things to be aware of, however, when using traceroute. They all work "essentially" the same, by sending out packets with increasing TTLs (Time To Live). The first packet has a TTL of 1, so the first hop will decrement the TTL and since it is then 0, the host (usually a router of some sort) will send back an ICMP (Internet Control Messaging Protocol) packet that tells your host that the TTL exceeded. The program then increments the TTL until it reaches its destination (or after a certain number of "hops" have been exceeded). Under UNIX, the default behavior is to send out UDP packets to high numbered ports (in the 33000+ range), with the expectation that the destination host is not listening on those ports, and will send back "ICMP Port Unreachable".

The default behavior for the Windows "tracert" program is to send out ICMP Echo Request packets instead of UDP. With ICMP, the destination host will respond with "ICMP Echo Reply", telling traceroute it's done. The side effect of this is that sometimes UDP might be blocked by a remote firewall, but ICMP isn't, meaning the trace might seemingly only complete under Windows and not UNIX (or vice versa).

You can force the UNIX traceroute to use ICMP by supplying the `'-I'` switch. Now, there can be (and are) firewalls that block both ICMP ECHO and random UDP, and so traceroutes won't complete. Using the `'-T'` (TCP) switch and/or setting the destination port number, you can generally penetrate these ;-)