Network Performance Definitions and Measurement Exercises

Notes:

* Commands preceded with "$" imply that you should execute the command as a general user - not as root.
* Commands preceded with "#" imply that you should be working as root.
* Commands with more specific command lines (e.g. "GW-RTR>" or "mysql>") imply that you are executing commands on remote equipment, or within another program.
* If a command line ends with "\" this indicates that the command continues on the next line and you should treat this as a single line.

Exercises Part I

0. Log in to your PC/VM or open a terminal window as the sysadm user.

NOTE: During these exercises if you find that the apt-get command complains that some archives cannot be found, then you may need to update your apt package database. To do this type:

```
$ sudo apt-get update
```

Network Performance Metrics

1. ping revisited

ping is a program that sends ICMP echo request packets to target hosts and waits for an ICMP response from the host. Depending on the operating system on which you are using ping you may see the minimum, maximum, and the mean round-trip times, and sometimes the standard deviation of the mean for the ICMP responses from the target host. For more details see:

http://en.wikipedia.org/wiki/Ping

Blocking ping is generally a bad idea.

With all this in mind, try using ping in a few different ways:

```
$ ping localhost
```

Press ctrl-c to stop the process. Here is typical output from the above command:

```
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.006 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.006 ms
64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64 time=0.006 ms
64 bytes from localhost (127.0.0.1): icmp_seq=5 ttl=64 time=0.006 ms
64 bytes from localhost (127.0.0.1): icmp_seq=6 ttl=64 time=0.009 ms
64 bytes from localhost (127.0.0.1): icmp_seq=7 ttl=64 time=0.007 ms
^C
--- localhost ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5994ms
rtt min/avg/max/mdev = 0.006/0.008/0.020/0.005 ms
```

Question: why did the first ICMP response take 20ms while the remaining responses were much quicker? This is a type of delay. What kind is it?

2. traceroute revisited

You may have used traceroute before, but have you really looked at what it is doing? If not, read this:

http://en.wikipedia.org/wiki/Traceroute

You may need to install the traceroute command first. To do this do:

```
$ sudo apt-get install traceroute
```

Once installed try:

```
$ traceroute nsrc.org
```

Here's sample output from traceroute to nsrc.org (lines wrapped due to length):

```
traceroute to nsrc.org (128.223.157.19), 64 hops max, 52 byte packets
```

```
 1  gw.ws.nsrc.org (10.10.0.254)  1.490 ms  1.069 ms  1.055 ms
 2  192.248.5.2 (192.248.5.2)  2.741 ms  2.450 ms  3.182 ms
 3  192.248.1.126 (192.248.1.126)  2.473 ms  2.497 ms  2.618 ms
 4  mb-t3-01-v4.bb.tein3.net (202.179.249.93)  26.324 ms  28.049 ms  27.403 ms
 5  sg-so-06-v4.bb.tein3.net (202.179.249.81)  103.321 ms  91.072 ms  91.674 ms
 6  jp-pop-sg-v4.bb.tein3.net (202.179.249.50)  168.948 ms  168.712 ms  168.903 ms
 7  tpr5-ge0-0-0-4.jp.apan.net (203.181.248.250)  172.789 ms  170.367 ms  188.689 ms
 8  losa-tokyo-tp2.transpac2.net (192.203.116.145)  579.586 ms  284.736 ms  284.202 ms
 9  abilene-1-lo-jmb-702.lsanca.pacificwave.net (207.231.240.131)  303.736 ms
    284.884 ms  530.854 ms
10  vl-101.xe-0-0-0.core0-gw.pdx.oregon-gigapop.net (198.32.165.65)  328.082 ms
    305.800 ms  533.644 ms
11  vl-105.uonet9-gw.eug.oregon-gigapop.net (198.32.165.92)  336.680 ms  617.267 ms
    495.685 ms
12  vl-3.uonet2-gw.uoregon.edu (128.223.3.2)  310.552 ms  421.638 ms  612.399 ms
13  nsrc.org (128.223.157.19)  309.548 ms  612.151 ms  611.505 ms
```

Do you understand what each item means? If not, see the Wikipedia page and type:

```
$ man traceroute
```

for more information. What does it mean if you see lines like this?

```
15  * * *
16  * * *
17  * * *
```

When you see this it means that the remote device does not reply to icmp echo requests, or it uses a private network address (RFC 1918).

As you can see traceroute can be used to determine where problems are taking place between two endpoints on a network.

Try running traceroute again to the same host (nsrc.org). It will likely take considerably less time.

3. mtr

The mtr tool combines ping and traceroute in to a single, dynamically updating display. Before using mtr you may need to first install it:

```
$ sudo apt-get install mtr-tiny
```

Now give it a try:

```
$ mtr nsrc.org
```

The output of the command looks different on different Linux and UNIX flavors, but in general you'll see a summary of packet loss to each node on the path to the remote target host, number of ICMP echo request packets sent, last rtt (round-trip-time) to the host, average, best and worst rtt as well as the standard deviation of rtt's.

By showing the percent loss of packets in this format it makes it much easier to see where you may be having network issues.

4. lsof and netstat

See what services are running on your machine. You can use the presentation as a reference.

Or, utilize "man lsof", "man netstat", "lsof -h" and "netstat -h" to see the available options (there are a lot!). Remember to use sudo when using lsof and netstat to give yourself necessary permissions to view everything.

You may need to install lsof. To do this type:

```
$ sudo apt-get install lsof
```

* Using lsof, what IPv4 services are listening on your machine?
* Using netstat, what IPv4 and IPv6 services are listening on your machine?

When you run lsof and netstat you should run them as root:

```
$ sudo lsof
$ sudo netstat
```

Remember - you will need to specify options to answer what IPv4 and IPv6 services are running on your machine.

Optional Exercises

Network Analysis

1. ping with variable packet size

By default, ping sends out IP datagrams of size 84 bytes:

* 20 bytes IP header
* 8 bytes ICMP header
* 56 bytes data padding

However, you can send out larger packets using the -s option. Using `-s 1472` will give you a 1500-byte IP datagram, which is the maximum for most networks before fragmentation takes place (MTU = Maximum Transmission Unit)

This simple mechanism can be used to debug all sorts of problems, and even distinguish between transmission delay and propagation delay.

Let's find a host that is a few hops away from us. First do:

```
$ traceroute nsrc.org
```

Now look for a machine that is more than two hops away and make a note of the IP address. Why? Because one hop is your virtual router and this exercise will not work reliably using virtual hardware. The second hop is the gateway router on our private network. It is too close and the difference in ping times are likely to be too small to be useful. We'll refer to the machine you choose to ping to as PING_MACHINE.

Send 20 standard pings to that address:

```
$ ping -c20 PING_MACHINE
```

Make a note of the *average* round-trip time seen (t1).

Now send 20 maximum-sized pings:

```
$ ping -c20 -s1472 PING_MACHINE
```

Again, make a note of the *average* round-trip time seen (t2).

The propagation delay is the same in both cases, so the larger round-trip time must be due to transmission delay.

You can now estimate the transmission delay and hence the bandwidth of the link between two points

```
increase in transmission time   =  t2 - t1
increase in bits sent           =  (1500-84) * 8 * 2  = 22656
```

(multiply by 2 because the round-trip time involves sending the packet twice)

Divide the bits by time to get an estimate of bits per second. Remember to convert milliseconds to seconds first.

Example:

t2 = 1.71
t1 = 1.14

t2-t1 = 0.57

0.57 ms = 0.00057 sec

22656 bits / 0.00057 sec = 39747368.42 bps

You could then convert this to Kbps, Mbps, etc.

By doing this for subsequent hops, it's possible to estimate the bandwidth on each hop, even those remote from you. There is a tool available which does this automatically - it's called "pathchar" but you have to build it from source. A few OS-specific binaries are available at:

ftp://ftp.ee.lbl.gov/pathchar/

The web page, including documentation is available here:

http://www.caida.org/tools/utilities/others/pathchar/

2. tcpdump and tshark

First we need to install both these programs:

```
$ sudo apt-get install tcpdump tshark
```

Use tcpdump like this:

```
$ sudo tcpdump -i lo -A -s1500 -w /tmp/tcpdump.log
```

Now, generate some traffic on your lo interface in another terminal. That is open another ssh session to your pc/vm.

For example:

```
$ ping localhost
$ ssh localhost
```

etc. Afterwords press CTRL-C to terminate the tcpdump session.

Note: ssh generates much more "interesting" output. Now let's read the output from tcpdump using tshark:

```
$ sudo tshark -r /tmp/tcpdump.log | less
```

What do you see? Can you follow the SSH session you initiated earlier?

Next we'll use ftp. First we need to install an ftp client:

```
$ sudo apt-get install ftp
```

Now try something like this:

```
$ sudo rm /tmp/tcpdump.log
$ sudo tcpdump -i eth0 -A -s1500 -w /tmp/tcpdump.log
```

In another terminal do:

```
$ ftp limestone.uoregon.edu

Connected to limestone.uoregon.edu.
220 FTP Server ready.
Name (limestone.uoregon.edu:sysadmin): anonymous
Password: <anything you want>
ftp> exit
```

End the tcpdump session in the other terminal (CTRL-C). Now view the contents of the log file:

```
$ sudo tshark -r /tmp/tcpdump.log | less
```

Can you see your password? If you have a lot of traffic on your network, then the tcpdump.log file may be fairly large. You can search for your FTP session by typing:

```
"/FTP"
```

in the output screen. Since you piped your shark command output to the "less" command using the "/" to search for strings works. Now press the "n" key for "n"ext to follow the FTP session. You should see a line with the string:

```
"FTP Request: PASS PasswordYouTypedIn"
```

Sniffing unencrypted passwords on wireless lans is very easy with a tool like this.

Rememer to clean up after yourself:

```
$ rm /tmp/tcpdump.log
```

3. tcpdump part II

You can use tcpdump as a forensic tool in real-time as well. To completely cover tcpdump would take several hours of class time, but let's get started with another practical example.

Let's watch a dhcp request from your PC and the responses that it receives.

First connect to your PC image and become root:

```
    $ sudo bash
```

Next we are going to use a utility called screen:

```
    # apt-get install screen
```

Now run screen:

```
    # screen
```

At this point you can have multiple terminal sessions open in a single ssh window. Let's start
the tcpdump process listening for dhcp requests:

```
    # tcpdump -s0 -ni eth0 port 67 or port 68
```

Now use screen to open another "screen" in your ssh terminal window.

```
        Press ctrl-a c
```

To figure out what "-s0", "-n" and "-i" are doing you can read the tcpdump man page:

```
        # man tcpdump
```

Search for "-s" by typing a "/" and then "-s" and then press ENTER. Press "n" to see the next
occurrence of the string "-s".

Now make a dhcp request for a new address for eth0 on your machine:

```
        # dhcclient
```

Return to the previous screen to see what tcpdump displays:

```
        Press "ctrl-a p"               ("p" for previous, "n" for next" to cycle through screens)
```

You should see some output like this:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
18:03:05.003190 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 52:54:4a:5e:68:77, length 300
18:03:05.004349 IP 10.10.0.254.67 > 10.10.0.250.68: BOOTP/DHCP, Reply, length 300
```

```
        To stop the tcpdump session type "ctrl-c"
```

Do you know what this means? Why did we specify to listen on ports 67 and 68? If you look in the
file /etc/services you will find the following defintions for ports 67 and 68

```
bootps           67/udp     # Bootstrap Protocol Server
bootps           67/tcp     # Bootstrap Protocol Server
bootpc           68/udp     # Bootstrap Protocol Client
bootpc           68/tcp     # Bootstrap Protocol Client
```

You can return the screen where you ran dhcpclient and exit from the screen if you wish:

```
        ctrl-a-n
```

Then type:

```
        # exit
```

If you are interested in the screen utility and how it works see:

```
        http://www.howtoforge.com/linux_screen
```

for more information or ask your instructor.

4. Using iperf

First we need to install iperf:

```
        $ sudo apt-get install iperf
```

Use "man iperf" or "iperf -h" for help.

Ask your neighbor to run:

```
        $ iperf -s
```

Connect to your neighbor's machine using:

```
        $ iperf -c ipNeighbor
```

If you don't know the IP address of your neighbor's machine ask them to
do:

```
        $ ifconfig eth0
```

and tell you what IP address their machine is using.

How much throughput is there between your machines? You can repeat this exercise with any remote machine where iperf is installed and you have an account. This is a quick way to see what bandwidth looks like between two points.

To stop the iperf server where you ran "iperf -s" press CTRL-c.

If you have time continue playing with iperf options. If you have a remote PC running UNIX or Linux you might want to try installing iperf and testing your connection from the workshop lab to your remote machine.

Some more things to try...

* Test TCP using various window sizes (-2).

* Verify TCP MSS (-m). How does this affect throughput? What is Path MTU discovery?

* Test with two parallel threads (-P) and compare the totals. Is there any difference? Why?

* Test with different packet sizes and the TCP_NODELAY (-N) option.