COSC 465: Computer Networking
Lab 4: Network measurement experiments
Due: 9 March 2012, 5:00 pm

1 Overview

Note: this lab is due before you go on Spring break. There is no new lab starting next week. You're welcome to work in groups of up to three persons for this lab. Only one submission needs to be made for the group.

In this lab you will use five different measurement tools to characterize 5 different network paths. You should examine 2 paths on campus (we'll start on these during lab), and you'll need to run experiments on 3 wide-area Internet paths using the PlanetLab system. For the measurement tools, we'll use bitblaster, ping, traceroute, iperf, and a tool called Netalyzr.

As part of this lab we'll use a distributed environment called PlanetLab. PlanetLab consists of over 1,000 hosts located primarily at academic institutions around the world. We have two hosts located at Colgate, and we can get access to any one of the hosts located in PlanetLab. (All PlanetLab systems are Linux-based; the only interface available is a command-line via ssh.) We'll use some of these machines for running measurement experiments in order to learn something about wide-area Internet paths.

If you did not complete the bitblaster lab, an executable program is available to download for use on Linux systems (see below for links to all the tools).

There's bonus available in this lab. Read on.

2 Detailed Description

2.1 Paths to measure

You need to measure characteristics for 5 total paths. For each path you'll need to take some measurements in each direction (some measurement techniques only measure characteristics for one direction of the path).

Two paths must be on campus. One endpoint should either be `cs.colgate.edu`, `toucan.colgate.edu`, or one of the PlanetLab hosts at Colgate (`ebb.colgate.edu` or `flow.colgate.edu`). The other endpoints must be outside McGregory Hall, and can either be a wired or wireless location; they should be (very) different locations (e.g., other sides of campus). For example, you could choose the library as one endpoint (so the path would be between the CS department and the library), and the O'Connor Campus Center as the other endpoint (path would be between the Coop and the CS department). We'll take measurements for at least one of these paths during lab. (Note: a good choice for another location would be your residence. If you do that, and you're off campus, you should use the PlanetLab hosts as the other endpoint since they are not firewalled from the rest of the Internet.)

The remaining three paths should be over the wide-area Internet. For each path, one endpoint should be in North America and should be the same for each path (i.e., one endpoint of each path will be the same); this can either be a machine in the CS department, or some other machine accessible via PlanetLab. The other three hosts should be located on three different continents; one of these can be in North America (thus one wide-area path would be entirely within North America).

Some number of hosts have already been added to a PlanetLab configuration, but you're welcome to add others if you want. See below for instructions on getting on PlanetLab and configuring our slice. (Note that everyone *must* get an account on PlanetLab as part of this lab.)

2.2 Characteristics to measure

There are 7 characteristics you should measure or compute for each path. Some of these characteristics will need to be measured in both directions of the path. The following list refers to various measurement tools; these tools are described in more detail in the next section.

1. The (minimum) average round-trip time of the path. You can simply use the `ping` tool for this measurement. This measurement should be as close as possible to the (round-trip) propagation delay for the path.

2. The capacity of the "bottleneck link" for each direction of the path. This one is harder to measure, but two tools that can help are Netalyzr and iperf. You need to measure this in each direction, because some link layers can have asymmetric capacities. The hard part of measuring the capacity of the bottleneck link is that it is not easy to tell whether the limiting factor is the fundamental transmission capacity of the link, or a device that is imposing some rate limit. You'll need to consider this complication as you use the measurement tools and try to decipher their output. (Netalyzr is mostly useful for detecting bottlenecks close to the host from which you're running the tool, so you'll want to run both tools and come up with your best estimate.)

3. The number of "hops" (routers) along each direction of the path. This one is fairly straightforward to measure with the `traceroute` tool. We will use a fancy version of `traceroute`, called `paris-traceroute` (how could it not be fancy with "paris" in its name?). It's much better than the default traceroute installed on many Linux systems; there's a link at the bottom of this document to the source code and a prebuilt executable. Note that to get the number of hops in each direction, you'll have to run `paris-traceroute` twice for one path.

4. The number of "autonomous systems" (separate networks, or network service providers) along a path, and their network prefixes. The output of `paris-traceroute` shows a series of IP addresses of the routers along a path. You can use these addresses (and some information in the DNS names) to look up address allocation information using the `whois` tool. You should use the on-line version of `whois` made available by Cymru, at `http://asn.cymru.com/`, to determine ownership of an IP address.

5. An estimate of the packet loss ratio along each direction of a path. Given your estimate of the bottleneck link capacity for one direction of a path, you'll use your super-fancy bitblaster tool to send a uniform stream of packets at two rates: 1/2 the estimated bottleneck bandwidth, and 1/4 the estimated bottleneck bandwidth (these rates may need some tuning as you run your experiments). The packet size you use should be no smaller than 500 bytes, and no larger than 1000 bytes. You should send a long enough packet stream for the measurement to last about 60 seconds. From the bitblast receiver output, you should see whether any packets are lost (and hopefully some are!).

   If you are unable to successfully run bitblaster due to too many packet drops (leading to the inability of the receiver to detect the end of the stream), you can use iperf in a similar manner as bitblaster. See below for instructions.

   The final number you produce for this measurement should be an estimate of the "loss rate", which can be defined as simply the number of packets lost divided by the total number of packets sent.

6. Using information you've collected, compute an estimate for the throughput of a long-lived TCP connection. You'll plug in some numbers to a formula created by Mathis *et al.*:

   $$Tput = \frac{MSS \times C}{RTT \times \sqrt{p}}$$

   where MSS is the "maximum segment size" in bytes (you can simply think of this as the maximum packet size), RTT is the round-trip time in seconds, p is the loss rate, and C is either the constant 1.22 or 0.87. (If you assume that a TCP receiver sends an ACK for every packet, you should use 1.22, otherwise 0.87 for "delayed" ACKs. Delayed ACKs are more prevalent, but I don't care which constant you use.)

   See `http://dl.acm.org/citation.cfm?id=264023` for the original paper. After 15 years, it is still a pretty good high-order approximation of the expected throughput for a long-lived TCP connection.

   We'll learn much more about TCP soon. It is the most widely used transport protocol, and provides a reliable, connection-oriented, byte stream to applications. Generally speaking, about 80–90% of all Internet traffic is

TCP. As you might learn here, TCP can be quite sensitive to loss. (If you did not see *any* packet loss in your bitblaster experiments, you can assume a loss rate of 0.001).

7. Measure some aspect of the buffering "elasticity" of the path by injecting a long-lived bitblaster flow while running ping alongside it. Run your bitblaster tool again, using a fairly high rate (at least 1/2 the bottleneck bandwidth, but perhaps more), and configure the packet stream length so that it runs for at least 1 minute. Simultaneously, use the ping tool to measure round-trip delays. The measurements you collect should be the reported minimum, maximum, average, and standard deviation for the round-trip times from ping.

2.2.1 Writeup

For your lab results, you'll need to organize and present the measurements required from the list above. You can present the raw data in a tabular form, similar to below, or in some other reasonable format. You should accompany your measurements with a brief discussion of the results. Include in your discussion the following:

- Measurement problems (i.e., were there technical difficulties in obtaining various measurements? Why?),
- Some detail about the traceroute results, including the observed round-trip times for each hop, where you think trans-oceanic hops are likely to be, and your estimates of geographic locations of routers. For locations, you can either infer from the DNS name, or use an IP geolocation tool like `http://www.maxmind.com/`. (Note that maxmind imposes a limit on the number of lookups that can be done per day, but there are other geolocation tools out there, too. Use google.)
- Any results that you find counter-intuitive, surprising, or just "interesting".

An example of a tabular format you might use to display some raw results is as follows:

A: ebb.colgate.edu, B: planetlab3.canterbury.ac.nz

	RTT	Min capacity	BDP	Num hops	Num ASes	Loss rate	TCP tput estimate
A→B	200 ms	100 Mb/s	19.07 MB	17	5	0.013	5.26 Mb/s
B→A	210 ms	100 Mb/s	20.03 MB	16	5	0.021	3.10 Mb/s

2.3 Measurement tools

ping This is a good-old ICMP echo request/echo reply. Ping can tell you an estimate of the round-trip delay between two hosts. The tool should be available on all hosts. See `man ping` or `ping --help` for help using it.

The only required command-line argument is the IP address or DNS name of the target host.

paris-traceroute This is a modern variant of the classic router path discovery tool. The reference for paris-traceroute is at `http://www.paris-traceroute.net/`. I'd suggest you run the tool with the `--protocol=icmp` option; you'll generally have better success. (There's a paper comparing different uses of traceroute that I can refer you to if you want to learn the gory details.)

As with ping, the only required command-line argument is the IP address or host name of the target to trace to.

You can get a pre-built Linux binary for running on PlanetLab at `http://cs.colgate.edu/~jsommers/cosc465`. You can also find the source code at the main site linked above.

Netalyzr Netalyzr is a pretty cool tool by researchers at the International Computer Science Institute at UC Berkeley. It's written in Java and attempts to discover what it can about how the host it runs on is connected to the Internet.

You can download the JAR file either from the Netalyzr site at `http://netalyzr.icsi.berkeley.edu/`, or I've put one at `http://cs.colgate.edu/~jsommers/cosc465`. On PlanetLab hosts, you'll have to ensure that Java is installed (just type `sudo yum install java`), and then just type `java -jar NetalyzrCLI.jar`. When the tool finishes, it will show a URL that you can load to view the results.

iperf iperf is a standard tool for generating long-lived TCP flows, as well as constant bit-rate UDP traffic streams. You can install iperf on PlanetLab hosts by typing `sudo yum install iperf`, and the source code and documentation can be found at `http://iperf.sourceforge.net/`.

To start iperf in a long-lived TCP flow scenario, you have to start both a server and client. On the server (the machine that's generating traffic), type: `iperf -s -w 1M`. The `-w 1M` option says to create a 1 megabyte send buffer (if possible). We do that in order to try to create as much traffic as possible.

On the client, type `iperf -c <server ip> -w 1M -P 10 -i 10 -t 60`. The `-P 10` option says to start up 10 parallel TCP flows, and the `-i 10 -t 60` options say to run for 60 seconds and print status every 10 seconds. In this configuration, iperf can generate 1 Gb/s in a local area network pretty easily.

If you need to use iperf in a UDP mode, use the `-u` option in addition to the above, and use the `-b` option to specify the bitrate. (These options will make iperf work similarly to bitblaster.)

bitblaster Your excellent tool for generating a constant bit-rate UDP traffic stream. You already know how this works. If you need correctly working binary files for the sender and receiver, see `http://cs.colgate.edu/~jsommers/cosc465`.

As noted above, Linux binary executable files for `paris-traceroute` and bitblaster, and the Netalyzr Java JAR file are available at `http://cs.colgate.edu/~jsommers/cosc465`.

2.4 Using PlanetLab

Three of the paths for this lab involve the wide-area Internet. We'll use the PlanetLab infrastructure (`http://www.planet-lab.org/`) to get access to hosts around the world.

1. First, get your account set up on Planetlab. Go to `http://www.planet-lab.org` to create an account. When you create an account, you'll need to choose a home site. Choose Colgate.

2. Once you have an account, use the `ssh-keygen` program to create a public/private key pair for ssh authentication. At a Linux command line (on the host from which you'll be ssh'ing onto PlanetLab machines), type `ssh-keygen -t rsa -f ~/.ssh/id_rsa`. This will create a public key in the file `~/.ssh/id_rsa.pub`.

   Upload your public key to your Planetlab account. Once you're able to log in to the PlanetLab website, go to your account information and upload your ssh public key. I'll automatically add you to a "slice" for this class, called `colgate_cosc465`.

3. At this point, you can add new hosts to the `colgate_cosc465` slice (see the "My Slices" link on the PlanetLab website), or you can try logging into any one of the hosts that has already been added to our slice. (Note: a "slice" is a PlanetLab term that basically must means a collection of virtual machines. When you add a host to our slice, you're adding a virtual machine on a particular machine that we can then ssh in to.)

4. When you log into a PlanetLab host, you'll need to specify `colgate_cosc465` as the user name (so, your ssh command line will be something like `ssh colgate_cosc465@ebb.colgate.edu`). Note that after you add a new host, it may take a while (up to an hour or more) for the virtual machine to become active. Also note that some PlanetLab hosts are extremely slow (there may be heavy user load on them), so it may take some tinkering to get on to far-flung hosts out there.

5. Once you can ssh onto the host, create a directory for your group. Since the entire class is using the same slice, you'll probably want to avoid overwriting someone else's files.

   To move files on and off the machine, you'll have to use `scp`, and if you need to use various Linux command line tools, you may need to install them with the `yum` utility. You'll have `sudo` access on all hosts, so you can simply say `sudo yum install gcc`, if you needed to install gcc, for example.

   To copy `myfile` to a the `mydirectory` subdirectory in my home directory on the host `ebb.colgate.edu` using `scp`, the syntax is:

   ```
   scp myfile colgate_cosc465@ebb.colgate.edu:~/mydirectory
   ```

   There's a similar syntax for "pulling" a file from a remote machine. See the `scp` man page for details.

For further details on using PlanetLab hosts, see the official users guide at: `https://www.planet-lab.org/doc/guides/user`.

2.5 Bonus!

You can earn 1 bonus point *for each direction of each wide area path*, if you can convincingly (to me!) do the each of following:

- identify the *location* of the bottleneck link (i.e., between which routers),
- or derive an estimate of the available bandwidth ("unused capacity") along a path.

For two characteristics, three paths, and two directions for each path, that gives a maximum bonus of 12 points (i.e., more than 20% for a 50 point lab).

To earn each point you'll need to provide evidence that you've correctly measured the characteristic in question. This evidence will almost certainly be in the form of measurements from some kind of network measurement tool. There are many different tools that have been developed in the research community to estimate these characteristics (some that I've written). You're welcome to use any of those tools, but you'll need to be clear for each bonus point which tool you used, how it was configured, what the measurement was, and why you think it is accurate.

One place to start looking is Sally Floyd's page of tools for bandwidth estimation (and other measurement tools), located at `http://www.icir.org/models/tools.html`. Note that all the tools linked there are *research quality*, which is probably worse than *alpha* quality. Your mileage may vary.