EECS 325/425
Project 2.
Due December 5 at 10am.
EECS325: 40 points; eecs425: 50 points

(EECS 325 and 425 crowds: 40 points):

In this project, you will investigate the correlation between several common inter-host distance metrics: the hop count, the RTT, and – for 425 crowd -- the geographical distance. Before you do your measurements, you need to implement a program that measures hop distance and RTT to a remote host. To reduce the number of probes, this program must perform a binary search for the number of hops between you and the destination by sending by sending a UDP probe to an unlikely port. Specifically, start by sending a probe with TTL = 16; if it's too small **(how would you tell?)** double it until it's too high **(how would you tell?)**. Then, explore the TTL at mid-section between the last two TTL values (e.g., if TTL=16 was too small and TTL = 32 too long, send a probe with TTL= 24), and depending on the outcome concentrate on the appropriate half-section. The output of your tool must include (a) the number of router hops between you and the destination, and (b) the RTT as measured by the probe with the final TTL.

Going back to your hops/RTT measurement tool, you need raw sockets to implement this tool. Google for "raw socket tutorial" or something like that.

A few notes:

(1) You will need to use python to implement this program. If you do not know it, just find some intro tutorial with lots of examples. I do not need you to learn the whole language, just enough to implement the task at hand. Note: you can find on the Internet code that implements somewhat similar functions, such as ping and traceroute. You are welcome to read this code or even reuse parts of it as long as you document carefully what code is yours and what you adopted from elsewhere.

(2) Use Linux. Even if your machine is Windows, install a virtual machine (e.g., https://www.virtualbox.org) and Linux within it. I am told Window's support for raw sockets is spotty.

(3) You will need to have root privilege to work with raw sockets.

(4) The easiest way to create a datagram with a custom TTL is to create a socket of type socket.SOCK_DGRAM and then use setsockopt() to change the TTL field of your socket. Then you can send it with socket.sendto() without going through the trouble of building the rest of the headers yourself. Alternatively, can set an IP_HDRINCL socket option to prevent the IP layer from populating IP header for you, and then fill out the header.

(5) You will need to think **how you will match ICMP responses with the probes you are sending out.** Note that your socket may receive unrelated packets since there is no port number to distinguish "your" packets from someone else's (i.e., another process running on your host). Hint: look at the format of ICMP messages you might get back and think which fields you can have control over and use for request/response matching. Note that there is no guarantee that someone on the path (e.g., VPN if you run your program while on VPN) would not change your lower-level IP header fields.

(6) You need to allow for a possibility that you will get no answer **(list all possible reasons you can think of for not getting the answer)**; hence your program should not be stuck on reading from a socket forever. You will need to use either a "select" or "poll" call on the

socket (read about them – google for "socket select") before reading. WARNING: do not process this error by changing the port number – you may get a nasty call from network admins! This will be interpreted as port scanning. I suggest you just double the TTL and if you reach your maximum without an answer, give up on this destination and produce some error message and move on to measuring the next site. You can check manually if its your program's fault by trying to reach this destination using a standard traceroute that uses UDP messages ("traceroute –P UDP –p <the port you are using in your probes").

Once you have your tool, measure the hop count as well as RTT to each of the alexa sites you have been working with. (The debacle we had with inappropriate sites should hopefully not be an issue because you will be accessing those sites programmatically and not by hand, and only with UDP messages to a closed port; thus you will not access or see any content.) Produce a scatter graph to visualize the correlation: have hops on X-axis, RTT on Y-axis, and for each remote host, place a dot with corresponding coordinates on the graph. This is a typical technique to visualize correlation. Note, as mentioned, you do not get responses from many of the servers. In order to produce a meaningful scatterplot, pick some other sites that were not included in your original list of ten. Keep probing until you have around ten. You can say in your report that you are substituting these sites because your original sites did not respond.

425 crowd only (10 points; 10 bonus points for 325):

To measure geographical distance, use the web service at http://freegeoip.net/

Write a script that sends, e.g., the following HTTP requests to all of your 10 destinations:
http://freegeoip.net/xml/129.22.104.136
and parse the response (which in this case will come in XML format).

Produce two additional scatter graphs to visualize the correlation between hops and geo-distance and RTT and geo-distance. Compute the correlation coefficients between all three metric pairs.

Deliverables:

A single zip file with (a) all programs (make sure they are well commented and include instructions how to run them – arguments etc. underdocumented programs will be penalized); (b) Project report that includes all measurement results, graphs, correlation coefficients, and the answer to all questions in bold.