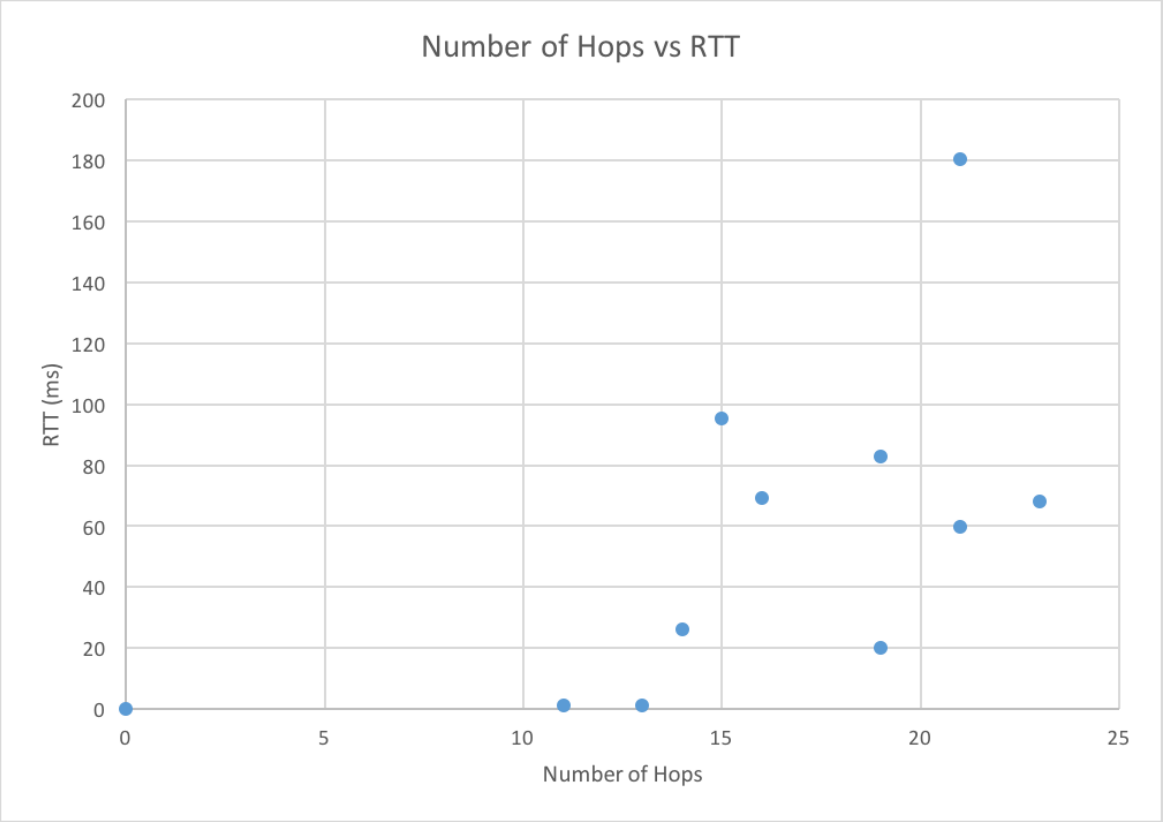
Ellis Saupe ems236
Project Report

A table and graph of my results is shown below. Many of the top 100 Alexa sites did not send an ICMP response, so I choose 10 arbitrary websites. I tried to pick newspapers and universities from other countries to get a good variety of RTTs and numbers of hops.

Name	Hops	RTT (ms)	Original datagram included in ICMP response(B)
localhost	0	0.037	548
google.com	13	1.089	28
mit.edu	14	26.109	548
titech.ac.jp	21	180.436	28
lemonde.fr	15	95.475	548
mg.co.za	11	1.179	548
spotify.com	19	82.892	28
360.cn	23	68.15	548
thelocal.de	19	20.153	28
heraldsun.com.au	16	69.213	548
santiagotimes.cl	21	59.635	548



As the graph shows, there is a generally positive correlation between number of hops and RTT, but there is no strong linear relationship between the two. Ignoring the data from localhost as an outlier, the linear correlation coefficient between number of hops and RTT is 0.574. These results are reasonable because the delay associated with each hop is expected to be very different. Leaving Cases local network infrastructure make take 3-4 hops but only account for 2-3ms to the RTT between two hosts, while crossing the trans-Atlantic cable is only 1 hop, but accounts for 90ms of the RTT. Assuming there was no major queuing delay along any route, the most significant source of delay to all of these targets was the propagation delay. Because the propagation delay has the strongest influence on RTT, I expect that the relationship between geographical distance and RTT is much stronger than the relationship between the number of hops and RTT as shown here. The weakly positive correlation shown here is likely a result of the number of hops being a weak approximation of geographical distance.

Every ICMP response shown included either the first 28B of the original IP datagram, or the first 548B. While testing my program, the website bilde.de included 68B of the original message in the ICMP response. However, this is not included in my results because their server stopped sending me ICMP messages before I was recording results. Because there are only two different values for the length of the ICMP payload other than bilde.de response, one behavior may be a Windows response and the other a Unix system response. Send the UDP datagram to the loopback address of the Linux VM resulted in an ICMP message including 548 bytes of the original datagram, so I would guess Windows systems include 28B of the original datagram and Unix systems include 548B.

You will need to think how you will match ICMP responses with the probes you are sending out (list all ways you could think of in your report and use the one that you found to work for you).

My program is single threaded and only sends one probe at a time, and I found that it was functional to not check for any errors when matching probes with responses; every ICMP message I received was a response to my single unanswered probe. To account for stray ICMP messages or using multiple threads to probe and reduce wait time, it would be useful to match responses to probes.

The most reliable was to match responses with probes is to match data with the 28 B, the IP header and UDP header, of the original datagram that is included with the ICMP response. Redundancy could be added by checking the ICMP type is 3, destination unreachable, but this shouldn't be necessary.

My program matches the UDP destination port from the UDP header included with the ICMP response with the port I sent the UDP datagram to. I also match the source IP from the original IP header included with the ICMP message with the IP address of the VM. This method should be sufficient, but I could also match the source port from the UDP header with my source port or the UDP checksum if I wanted a more reliable match.

If I was sending these messages from a raw socket or had access to the ID field of the IP datagram I sent, I could match the 16 bit ID of the original IP header with the ID field of the IP header included in the ICMP response.

For the servers that include more than 28 bytes of the original IP datagram, I could include a message id in the payload of the datagram before the disclaimer message and match that with the first few bytes of the payload included with the ICMP response.

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 when probing an arbitrary host)

I only probe one port so that it doesn't look like I'm port scanning. If a host happens to be using that UDP port for something else, I will not get a destination unreachable ICMP response because the destination will be reachable.

I wouldn't receive an ICMP response if my UDP datagram is lost, the host I'm probing is down, or if the ICMP response is lost.

Hosts my not send destination unreachable ICMP responses to prevent malicious UDP datagrams. If a host always sends destination unreachable messages, I can contact any UDP ports and know if the host is listening on that port. If there is some security hole associated with some process that uses UDP communication, it is much harder for me to identify a target if don't know what UDP ports the host is listening on.

I found that some hosts stopped sending me replies after a few probes were sent. They may have a rule that to stop sending destination unreachable responses if too are sent to one IP. This would be a simple rule to prevent port scanning while also allowing some functionality from destination unreachable messages.