Sarah Whelan
slw96
December 2, 2015
Networks 325

Project 2 Report

1. **Explain the technique you use to measure the hop distance from your machine to destination using a single probe.**

In order to use a single probe or packet to determine hop distance I utilized a single UDP packet and ICMP messages. On the single UDP packet I set the TTL field to a number I knew in advance – in this case 32. In theory any number would have worked (that fit within the bits of the field) but as Rami Al-Dalky pointed out and Professor Rabinovich relayed certain equipment will reset this field if the number is too high. This makes sense as a packet that stays alive for a long time without getting to its destination wastes resources. Professor Rabinovich also pointed out that it is rare to have hop counts that are larger than 32. This information is what I used to determine that 32 would be an acceptable value with which to set the TTL. With the TTL a known value I sent the UDP packet to the destination in the targets.txt file and to a port that would likely be unused. The port number I choose to use was an official traceroute port number as this is some variation on that. My reasoning was that most likely any port number would work as long as it was one that was not commonly used. Additionally in the interest of security or performance many proxies and servers may not wish to respond with ICMP messages on ports they are not using as it can be a drain on the network and server and not responding may protect against some more naïve attacks that just scan on ports. However some people may configure their equipment to respond to the Traceroute/Ping ports so as not to lose the ability to debug/check network on their own servers etc. Either way I tried my tool with several different ports manually and got similar response rates for every port I tried. I also set a message in the body of the packet an 8 byte string: abcdefgh. The message was just an additional precaution to ensure the responses I was getting were for the packets I was sending. I also determined my ip address and the ip address that I had send the packet to. Python uses the first DNS record that returns so I had to save the ip I used in order to check the returning packet against the original ip and not just a different first DNS record.

After sending my packet with known TTL to a port that should be unreachable I waited for a response. Initially I had a non-blocking socket I would only poll so many times but this had the same behavior as a blocking socket with a time out and the blocking socket code was cleaner so I went with that but both would have worked. Once I received my response I verified that it was in response to the packet I send (more on this in question 2) and knowing that what I was receiving was a packet with a set of IPv4 headers, then ICMP headers, then the IPv4 headers of the packet that caused the port unreachable ICMP message to be created at the destination and the first/all of the 8 bytes of my message. From this I pulled the TTL header field from the inner IPv4 headers (my original) and subtracted this from my known original TTL of 32 to get the number of hops that the packet took before it got to the destination. RTT was also measured with a single probe in that I got the time just before the packet was sent and again just after the response was received to determine how long the packet was travelling.

I noticed that in using my tool if I was on the case campus or it was a busy time for internet at my house (ie everyone else was awake too) that I'd get a lot of destinations not responding but if I tried the tool a bunch of times one after another I'd get a response so this tool was harder to test when a lot of the UDP packets were getting dropped. The internet activity also had a drastic effect on the RTT I was measuring but not on the hop count.

2. **Explain how you will match ICMP responses with the probes you are sending out**

To match ICMP responses to the probes I am sending out I checked specific header fields and the message field to ensure that the response was for the packet I originally sent.

Specifically I checked the IP source and destination addresses on both set of IP headers and if my message was returned I checked that as well (I was getting some packets where my message was not returned along with the headers but all of the other header fields were in the right spots). Additionally I checked the ICMP headers to ensure it was for the right error. Checked the source and destination IPs on the inner IPv4 packet for matching my ip and the ip I sent the original packet to. Checked the source and destination IPs on the outer IPv4 to match the ip I sent the original packet to and my ip address, respectively. If I got my message back I checked that it matched. Also checked if both the type and code of the ICMP message was 3 indicating it was host/port unreachable. I thought about setting and checking the identification field of the original packet but that should be used only for fragmentation and is apparently not allowed. (https://tools.ietf.org/html/rfc6864 and https://en.wikipedia.org/wiki/IPv4#Identification)

3. **list all possible reasons you can think of for not getting the answer when probing an arbitrary host**

- UDP packet was dropped on the way to the destination.
- ICMP message was dropped on the way back from the destination
    - Both of the above would be more likely in high traffic scenarios
- proxy dropped packet (either direction)
- firewalls
- Destination did not want to respond with ICMP
    - configured to not respond to ping/traceroute like things to avoid ping flooding/other attacks
- No internet connection
- Destination not found
- Server down
- For specifically my tool – timeout after 3 seconds
- There are many things that could go wrong I think that main culprits for lack of response for me were intermittent packet dropping and proxy/firewall/server settings to not respond

Data

List of original sites:

wikipedia.org
bongacams.com
wikihow.com
google.com.pe
kohls.com
souq.com
lemonde.fr
houzz.com
mi.com
tiexue.net

Only the ones in bold responded so to get at least 10 sites I substituted some other sites shown below in the table.

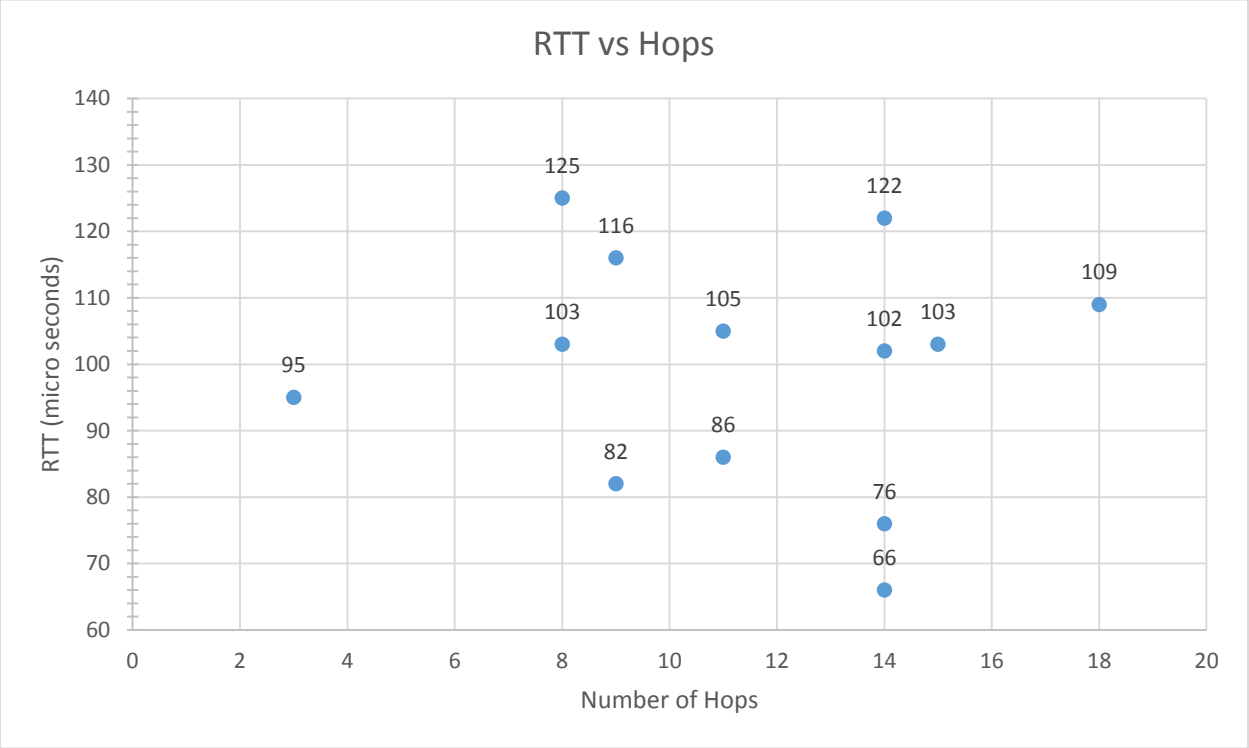
The following data was collected in one run of the tool at night and these were some of the best RTTs I received. In the runs the RTT would vary dramatically in periods of high traffic but the hops would not change as much. As this was one run the network conditions were the same for all of them and I choose to do this instead of averaging as I would have been averaging with much higher RTTS/possibly different hops (tangentially related to traffic but could change normally) and this shows the lack of correlation even in reasonable circumstances.

Data (Collected 11pm December 1st):

Host	RTT (microseconds)	Hops
google.com.pe	116	9
kohls.com	66	14
lemonde.fr	105	11
google.com	82	9
case.edu	95	3
news.ycombinator.com	125	8
callmedrew.com	103	15
facebook.com	102	14
twitter.com	103	8
espn.com	109	18
dropbox.com	76	14
target.com	86	11
etsy.com	122	14

The above table does not include the non-bolded sites from the list of original targets that did not respond (or the packet was dropped).

Graph/Scatter Plot:



Correlation Coefficient: -0.07488 ≈ 0

This correlation coefficient indicates that there is practically no correlation between RTT and number of hops. This is reasonable as the processing time at each hop is negligible compared to travel time across links. RTT would increase if a link was long or slow not how many different links.