EDI: First Lab Report

Aiman Al Masoud - 502044

April 30, 2022

Abstract

In assignment 1, the goal has been to determine the average daily peak time of a server, using an active monitoring technique. Whereas assigment 2 was about performing different experiments with the DNS.

1 Monitoring of a Server's Peak Time

1.1 Methodology and experimental setup

1.1.1 Hypotheses that were formulated

When a server is experiencing too many requests, the following things are expected to happen:

- H1) Latency Surges: Because the server has to deal with a higher amount of packets per unit time, queueing is expected to happen, and the round trip time of packets sent to the server is expected to increase.
- H2) Number of hops increases: At peak time, congestion may take place on the normally optimal routes, and thus alternative routes, which may include more hops, will be sought by the packets, increasing the average number of hops to reach the server.

1.1.2 Experimental Setup

A script that performs traceroute on a target IP at regular intervals (every 5 minutes) was executed on a vantage point for just over a week. Subsequently, the collected data was processed offline.

The vantage point in question was a virtual machine on GCP (Google Cloud Platform), located on a server in the US. The target website was: www.google.it. Assuming that www.google.it is mostly visited by users in Italy, the timestamps were interpreted as times in CEST, aka: GMT+2, which happens to be the Italian local time in April.

As a side-note, peak time traditionally occurs from late evening to early night, more precisely from 6 PM to 11 PM. [1]

The assumption that I made, in accordance with H1[1.1.1], was that latency (RTT) would've been proportional to the amount of traffic on the server.

1.2 Experimental results

After letting the script run for more than 183 hours (almost 8 days), collecting measurements at regular intervals of 5 minutes, I was left with 2204 readings; 4 of which were discarded as outliers.

The remaining 2200 measurements were processed and arranged in a table of this format:

timestamp	hops	reached	latency_ms	hour
2022-04-17 16:09:56	15	True	0.919000	16

Table 1: Table 1

The records were grouped by hour of the day, the mean and standard error were computed, resulting in this final table:

hour	latency_ms	latency_ms_std_err	hops	hops_std_err
0	0.885481	0.332606	12.326316	1.124600
1	0.911547	0.274709	12.784946	1.405343
2	0.923648	0.284790	12.562500	1.709494
3	0.883117	0.318890	12.436170	1.492335
4	0.951461	0.249570	12.702128	1.457865
5	0.915111	0.314960	12.252632	1.406432
6	0.955569	0.283697	12.138298	1.603766
7	0.914733	0.285439	12.357895	1.529259
8	0.876304	0.241643	12.463158	1.137471
9	0.909583	0.303091	12.361702	1.605441
10	0.870383	0.259852	12.241758	1.628501
11	0.877788	0.285976	12.559524	1.491754
12	0.882918	0.348783	12.592593	1.376388
13	0.897655	0.310639	12.180723	1.562866
14	0.888898	0.280578	12.469880	1.417120
15	0.827652	0.270974	12.271605	1.449244
16	0.872984	0.279684	12.728261	1.597410
17	0.846570	0.306079	12.505376	1.449315
18	0.823714	0.301251	12.437500	1.159060
19	0.870302	0.276282	12.800000	1.242852
20	0.864380	0.286903	12.677419	1.286732
21	0.892767	0.337298	12.505263	1.521993
22	0.860239	0.356526	12.468085	1.300968
23	0.933872	0.347573	12.606383	1.254824

Table 2: Table 2

Two bar charts were obtained from said table.

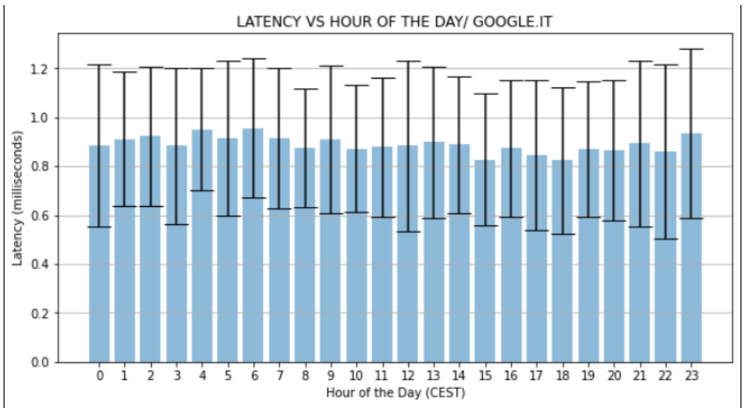

Figure 1: Latency Graph

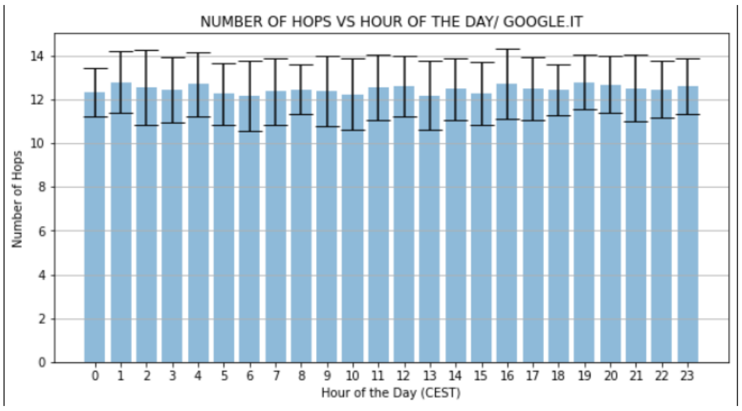

Figure 2: Hops Graph

It doesn't appear that there is much confirmation for the suspected trend in either graph, although it does seem, in Figure 1, that the latency (assumed to be proportional to the traffic towards the server) grows a little during the late evening (18 - 23).

For a more decisive answer, this experiment has to be repeated for a longer time interval, and perhaps also using many vantage points located at different parts of the globe.

2 DNS Experiments

In the second lab I experimented with the DNS using an array of cli tools designed to test the functioning and performance thereof.

2.1 Methodology and experimental setup

All of the following tests were performed at home from my local network.

2.2 Part 1

2.2.1 Query the DNS to obtain the IP addresses of ercole.unipv.it and of another website in the domain it; are the answers authoritative? Why?

Using the dig command, the ips are: 193.204.34.13 and 142.250.180.163 (google.it), and the answers aren't authoritative as in both cases (AUTHORITY flag = 0). This is because said answers aren't provided by a nameserver onwed by the owner of the domain.

2.2.2 Query the DNS to obtain the name(s) of the mail servers associated with the domain universitadipavia.it and berkeley.edu. How many servers provide this service? Is there anything specific associated with the RRs?

Using the dig command with the -t MX option (for MX RRs), there are 7 mail-servers associated to Pavia and 5 associated to Berkley. From their names you can tell that they're all actually Google servers, since the managment of email in these organizations has been outsourced to Google.

2.2.3 Query the DNS to obtain the IP address of a Web server in Asia; is the answer authoritative? Why? How many RRs did you obtain? What is their type? Does the domain sign any RR type using DNSSEC?

I queried akagiice-global.com (183.181.85.161), and the answer wasn't authoritative (AUTHORITY flag = 0) because it wasn't provided by a nameserver owned by that same organization. I queried for A type RRs and obtained one in the answer. The domain doesn't make use of DNSSEC, as running the following command:

```
dig akagiice-global.com +dnssec
```

yields a response devoid of any RRSIG records and lacking the AD (Authentic Data) flag.

2.2.4 Query the DNS to obtain the IP addresses of the authoritative Name Servers of a company outside Europe; how many queries did you execute? What type(s) of queries? How many Name Servers are associated with the company? Do they belong to the same domain? Can you identify the primary Name Server? Why? Who registered the domain? When will it expire?

I queried for the nameservers associated to the domain duolingo.com, like so:

```
dig duolingo.com -t NS
```

I obtained 4 names in the answer section:

duolingo.com.	172800	IN	NS	ns-1020.awsdns-63.net.
duolingo.com.	172800	IN	NS	ns-1117.awsdns-11.org.
duolingo.com.	172800	IN	NS	ns-1904.awsdns-46.co.uk.
duolingo.com.	172800	IN	NS	ns-247.awsdns-30.com.

And I had to perform a query for RRs of type A on each single name to obtain the IP associated to each name, so in total I had to execute: 1 type NS and 4 type A queries. From their names you can tell that the nameservers are owned by Amazon (AWS, Amazon Web Services), and each of them is registered under a different TLD (Top Level Domain). The following are the IPs associated to each of the names:

ns-1020.awsdns-63.net.	205.251.195.252
ns-1117.awsdns-11.org.	205.251.196.93
ns-1904.awsdns-46.co.uk.	205.251.199.112
ns-247.awsdns-30.com.	205.251.192.247

Running whois on the ...

2.2.5 Query one of the Name Servers identified in the previous experiment to obtain the IP address of the Name Servers of the domains unipv.it and samsung.com. How many IP addresses did you get?

Zero, the queries returned a status REFUSED flag, and no answer. I believe this is due to the fact that the name servers that I queried were not public, and hence wouldn't respond to a query coming from without their domain about another external domain. On the other hand, they did respond when I queried them for the domain duolingo.com.

2.3 Part 2

2.3.1 Measure the performance of a Name Server when processing multiple queries. Did you notice any variability? Any expected/unexpected behavior?

I used the tool dnsperf for this task, which for some reason isn't available on aptitutde as of writing this report, so I had to build it from source following the steps on their repository [2]

I measured the performance of Cloudflare's public name server (1.1.1.1), with the five following queries repeated 1000 times each:

www.usa.gov	A
www.usa.gov	NS
www.usa.gov	MX
www.usa.gov	HTTPS
www.usa.gov	RRSIG

Among the 5000 total queries that were executed, 4863 returned with a NOERROR code, and 137 with a "T", which is dnsperf's way of saying that the query timed out.

I used the verbose option to obtain the latency and response status of each single request. I then processed those with Python, and I obtained these average (on the query type) results for the latency in milliseconds.

A	0.056626
HTTPS	0.056006
MX	0.056162
NS	0.056179
RRSIG	0.056231

From the table it seems that these times appear to be all very similar, albeit the number of queries needed to establish that for sure is many order of magnitude greater than the total queries that I managed to run.

2.3.2 Measure the performance of different Name Servers when processing the same set of queries. Did the response time vary with the Name Server? Does it depend on the type of query or on the geographic location of the Name Server?

I used the tool dnseval for this task, and I queried the following nameservers: Cloudflare, Google, Cisco, Quad9; 1.1.1.1, 8.8.8.8, 208.67.222.222 and 9.9.9.9, respectively.

For each of the aforementioned nameservers, I queried the domain google.com for each of the following five RR types: ['A', 'NS', 'MX', 'HTTPS', 'RRSIG'], setting the -c option on dnseval to 50.

The first thing that I noticed is that -t HTTPS isn't considered a valid RR by dnseval. In terms of speed (average latency in ms), Cisco's nameserver was consistently the slowest. I don't think this depends on the geographic location of the name servers, as all of the servers are located in the US, with the exception of Cloudflare's, located in Australia, according to Keycdn [3]

Another thing worth noting is that only Cisco's name server didn't respond at all when queried for the RRSIG records associated to the domain google.com.

I conclude that the difference in performance in this experiment depend on the query type, as well as the geographic location of eventual CDNs (Content Delivery Networks), but probably not on the geographic location of the name servers themselves.

2.3.3 Check the path followed by your queries using different Name Servers; did you notice any expected/unexpected behavior?

I used dnstraceroute for this task. As expected, the routes started out pretty much identical, then they began diverging (approximately at hop 7).

It took 12 hops to reach Google's and Quad9's name servers, 11 hops to reach Cloudflare's name server and just 10 hops to reach Cisco's name server, albeit it scoring the largest latency on the previous experiment.

3 Conclusions

You can find all of the code employed in these experiments on Github [4]

4 References

1. `https://www.highspeedinternet.com/resources/why-does-my-internet-slow-down-at-night`
2. `https://github.com/DNS-OARC/dnsperf`
3. `https://tools.keycdn.com/geo?host=1.1.1.1+`
4. `https://github.com/aiman-al-masoud/edi_reports`