Universiteit van Amsterdam
System and Network Engineering

Research Project 1

Peeling the Google Public DNS Onion

Analyzing the Cache Coherency and Locality of Google Public DNS

Authors:
Tarcan Turgut
Rohprimardho

Supervisor:
Roland M. van Rijswijk-Deij

8 February 2015

Abstract

Google Public DNS is a global open DNS service that Google offers for free. Since its inception the service has become quite popular. Despite its popularity, revealing the working principle of the service does not seem within Google's future plans. The only official and very limited information is published on Google's official website. This project is mainly conducted in the light of this limited information. The goal of this project is to shed light on how Google Public DNS works.

In this paper, we explore two aspects of Google DNS: locality and cache coherency. A topology is built around the globe with 5 authoritative name servers and RIPE Atlas probes as DNS clients. In order to investigate the locality of Google DNS, lookups are initiated by probes from all around the world and BIND logs of authoritative name servers are observed to determine if the queries to an authoritative name server originate in the Google data center where the query is received or in the closest data center to the name server. This methodology then extended to discover whether Google maintains a globally single shared cache or not. Another methodology presents the analyzing simultaneously BIND logs and response TTL values in order to examine cache coherency in a single Google location.

Our experiments showed that the queries to the authoritative name servers originate in Google's data center closest to the clients. Regarding cache coherency, we found out that Google does not maintain a globally shared cache, each location has its own cache. Further analysis targeting a single location showed that each location may not also maintain a single shared cache by multiple resolvers, that is a hint to a possible cache fragmentation, in turn, a possible performance penalty. This paper also presents some routing anomalies of DNS queries and unexplained issues regarding cache coherency which have arisen during our experiments.

Contents

1. Introduction

In the last decade, there has been a dramatic growth in the number of internet users, as of July 2014, the total number reached 3 billions [1]. Having said that, the performance expectation of the users are also increased, especially on web browsing. Since web browsers are getting more complex, DNS lookups may create bottlenecks, thus, DNS resolution plays an important role in clients' experience. A user can either prefer its local ISP's DNS resolver or a public DNS service, such as Google Public DNS and OpenDNS.

Google Public DNS is one of the most preferred DNS providers around the world [2]. Google claims to offer free and fast DNS service to its clients. Despite its popularity, there is only little information about the underlying mechanism of the service. As a matter of fact, the only official explanation can be found on Google's official website [3]. For this reason, we can simply identify it as a "black-box".

1.1. Google Public DNS

Google Public DNS is a free global service that can be used as an alternative to local ISP resolvers. It uses global anycast addresses 8.8.8.8 and 8.8.4.4 to receive DNS queries from the clients. The IP address for the service is announced globally by Google's autonomous system (AS15169)and the traffic will be routed via the shortest announced route seen from the client's perspective. Google DNS servers are spread around the globe. As of February 2015, there are 13 locations as shown in Table 1.1. The IP subnets of those locations are also published by Google in [3].

Google basically uses three main methods to mitigate the DNS latency as using powerful servers, using the edns-client-subnet option and a high cache coherency as published in [3]. Since our focus is the cache coherency of Google Public DNS, here we mention only the cache mechanism.

Google Public DNS has 2 levels of cache. Level 1 cache, a small per-machine cache, contains the most popular domain names. If a query is not satisfied by Level 1 cache then it is forwarded to another pool machines where cache is partitioned by names. Each query for the same name are always handled by the same machine [4].

1.2. Research Questions

In light of the limited information on the working of Google Public DNS, we set out to learn more about its workings. To do so, we will address the following research questions:

City	Country
Taipei	Taiwan
Brussels	Belgium
Groningen	Netherlands
Morganton	USA
Atlanta	USA
Council Bluffs	USA
Charleston	USA
The Dalles	USA
Tulsa	USA
Lappeenranta	Finland
Santiago	Chile
Dublin	Scotland
Singapore	Singapore

Table 1.1.: Locations of Google public resolvers

1. Do queries to an authoritative name server originate in the region of the original query to Google Public DNS or are they local to the authoritative name server?
2. Is there a single shared cache for the whole service or do queries from different locations result in multiple queries to authoritative name servers? We subdivide this question into four subquestions:
   a) Is there any delay during the creation of the cache after flushing?
   b) Is it possible to figure out that all Level 1 cache are identical?
   c) Does Google Public DNS respect the TTL set by the authoritative name-server?
   d) Does Google Public DNS maintain a coherent Level 2 cache in a single location?

The research questions above differ somewhat from the original plan submitted at the start of the project, this is because we do not administer a popular domain and the only way to interact with Level 1 cache was to use Flush Cache Tool[1] which has crashed during this project (Google claims that any resource record can be flushed out of the whole service using this tool). Then we raised an issue ticket to Google Public DNS Team and they reported that there was bug on the tool. Therefore, we discarded the question 2a regarding cache flushing and the question 2b regarding Level 1 cache. Instead, we added a new question 2d regarding Level 2 cache.

By answering these questions, we would like to add valuable information regarding inner mechanism of Google Public DNS service.

[1]https://developers.google.com/speed/public-dns/cache

1.3. Related Work

There have been a large range of studies on the DNS performance, however, only few on exploring DNS caching mechanisms. The study of Huang et al. [5] helped us get the insight of the Public DNS working principle with examples including Google Public DNS. They present the ”DNS Beacon” technique that is used in order to uncover geographic presence of the Public DNS systems. Their technique is based on recording the unique IP addresses of the Public DNS servers by means of observing the authoritative name server logs. This idea led us to develop the methodology regarding the locality of Google Public DNS. In an early study by Jung et al. [6] studied how cache sharing can impact caching effectiveness and evaluation of DNS performance from client-side perspective. The methodology they presented as ”Trace-driven Simulation Algorithm” which examines the response TTL values to evaluate cache hit and miss rate gave us the idea of making use of decrement of the TTL values to analyze the cache coherency of Google Public DNS. In a study of Schomp et al. [7], the authors evaluate the caching behavior of recursive DNS servers and to what extent DNS servers are honest with the TTL values. They present a methodology comparing the TTL values set by authoritative name servers and the response TTL values to determine if the TTL values are modified by DNS servers.

Regardless, no research has been done regarding Google Public DNS cache coherency and locality up to this time.

1.4. Contribution

The expected end result is a proof of concept on how Google Public DNS maintains their cache coherency and locality around the world. The methodologies that we developed are also applicable to other public DNS providers, such as; OpenDNS and Level3. Also, our implications may contribute future studies in such a way that DNS cache coherency may present a possible DNS performance penalty.

2. Background Information

In this section, we give an overview of DNS infrastructure and RIPE Atlas probes.

2.1. DNS Overview

DNS (Domain Name System) is specified in RFC 1034 [8] and RFC 1035 [9]. We briefly summarize the basic design and the terminology used in this project.

DNS is a hierarchical globally distributed database that maps the human-readable domain names to the IP addresses of internet services (such as; www.example.org to 192.168.1.2). As a distributed service, the domain name space is cut into the portions (called zone, such as; example.org) and the administration responsibility of the zones is delegated to the authoritative name servers. Authoritative name servers maintain a zone file containing mapping information (called resource record, RR). The RRs have different types, however the most common type is "A" (address) which basically indicates the IP address of a certain domain name. Another important player of DNS is the resolver that query RRs from authoritative name servers in response to the recursive query initiated by the clients. A resolver may be administered by ISPs (local resolver) or by a public DNS provider (public resolver) such as; Google Public DNS and OpenDNS.

In order to achieve a low client latency, DNS makes use of caching. When a resolver conducts a DNS query on behalf of the clients, it stores the RR in its cache for further queries, thus the resolver is able to respond to the client immediately without any further search on the DNS tree [6].

Each RR has an expiration time (called Time-To-Live, TTL) which is set by the authoritative name servers. TTL is an integer value in seconds and defines how long a resource record should be kept in cache as described in RFC 1034 [8]. For instance, once a resolver caches an RR with a certain TTL value, say 300 seconds, it is responsible for decreasing the TTL value and after this period it must be discarded from the cache.

A DNS server is basically a software that implements DNS protocols. The most widely used name server software is BIND (Berkeley Internet Name Domain [10]. The authoritative name servers mentioned in this project are built by this software.

2.2. RIPE Atlas Probes

RIPE Atlas probe is a small hardware that can run network commands such as DNS query, ping, and traceroute. The result of these measurements are collected and reported to a database. The activity of the probe can be managed through a dashboard on RIPE

Atlas website. The aim is to build the largest internet measurement network initiated by RIPE NCC [11].

Everyone can register and create a user account on the site[1] to create a measurement with the probes. Each measurement deducts a number of credits from the user account. A user can earn credits by hosting a probe [12]. The credit system meant especially to distribute the usage of the probes evenly among the users. However, it is also used to prevent abuse. Thus, no credit, no measurement.

The probes are distributed across the world as shown in figure 2.1.

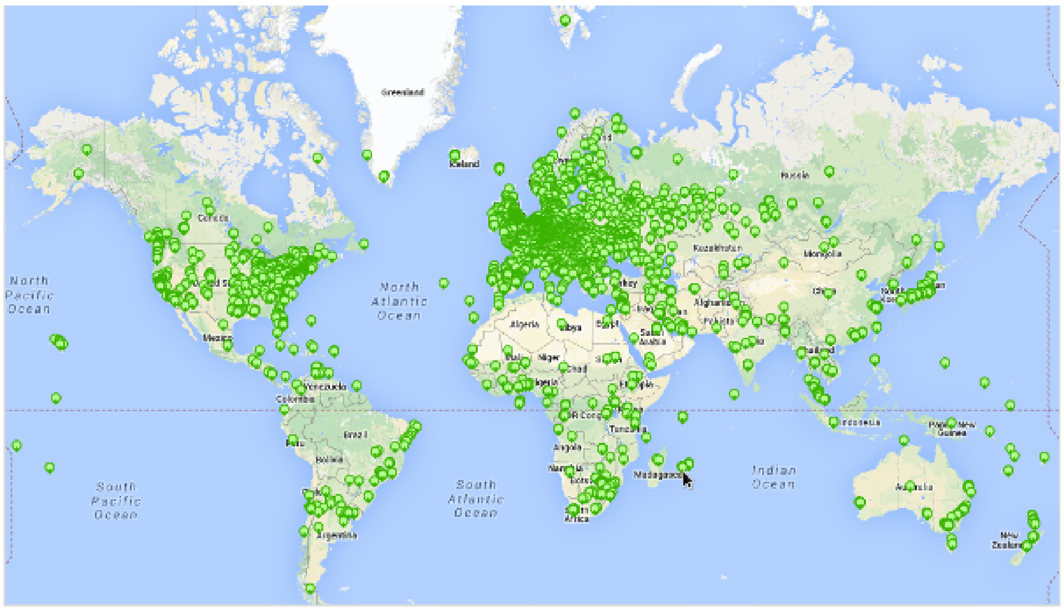


Figure 2.1.: RIPE Atlas probes distribution around the world [13]

We use these probes during the research to act as clients. The fact that they are spread around the globe is quite helpful for our research.

[1]https://atlas.ripe.net/

3. Methodology

3.1. General Topology

As the starting point of our research, we configured five authoritative name servers with BIND. We enabled the query log setting to be able to see the incoming DNS queries. We located these authoritative name servers in countries close to one of the Google Public DNS servers: The Netherlands, Chile, England, USA, and Singapore, as shown in the figure 3.1.

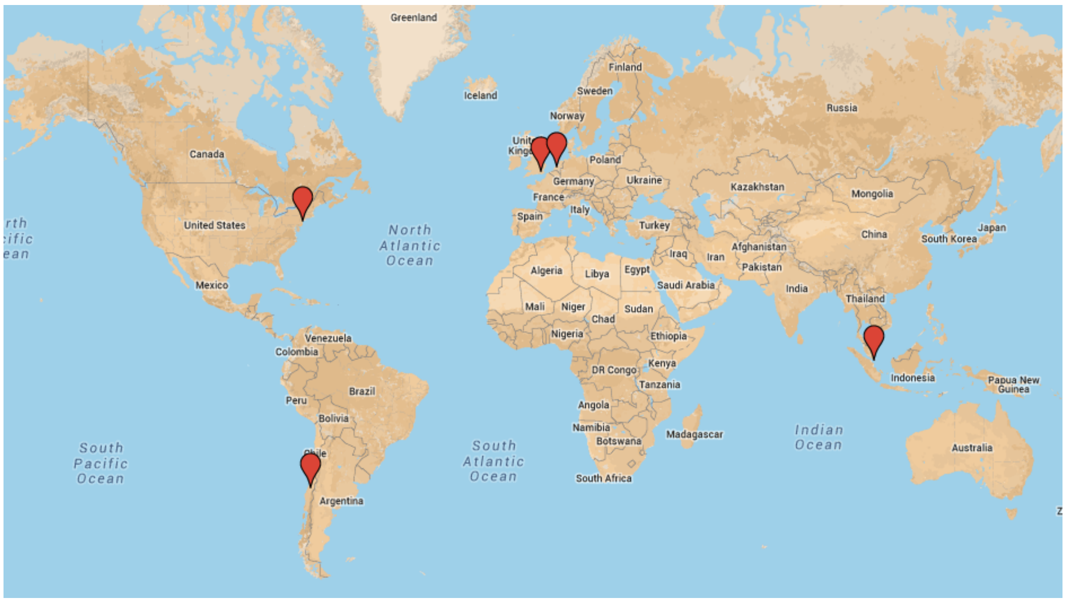


Figure 3.1.: The location of our authoritative name servers

With the help of SURFnet, we registered a domain name called inspectorgoogle.net. In table 3.1, five delegated subdomain names are shown with the associated authoritative name servers.

We chose to use gTLD .net is because the gTLD name server have a better globally distributed presence than .nl [14]. It minimizes any influence caused by unnecessary traffic from and to the name servers.

Subdomain name	Location of the authoritative name servers
nl.inspectorgoogle.net	The Netherlands
cl.inspectorgoogle.net	Chile
uk.inspectorgoogle.net	UK
us.inspectorgoogle.net	USA
sg.inspectorgoogle.net	Singapore

Table 3.1.: Delegated subdomains to the authoritative name servers

3.2. Origin of the DNS Query from Google Public DNS

To answer the research question mentioned in section 1 about locality, we applied the following methodology. We configured RIPE Atlas probes to send one DNS query to all of our authoritative name servers at a specific time stamp. The location of the probes were particularly picked to represent each continent in the world. By correlating the BIND query logs of the authoritative name servers and the time stamp of the DNS queries sent from the probes, we would be able to determine the origin of these queries.

Since Google Public DNS servers located around the world, it is interesting to know from which Google Public DNS server the query to the authoritative name servers originate. There are two possibilities: the Google Public DNS server close to the authoritative name servers or the one that close to the clients instead.

If the query originates from a Google Public DNS server close to the client, it would give us a small hint that there might not be a single globally shared cache. Had there been a single shared cache, we would expect that the query will be processed internally by Google Public DNS which will forward the query to the Google Public DNS server closest to the authoritative name servers.

3.3. Round Trip Time

We wanted to compare the round trip time (RTT) of traceroute to Google Public DNS from different location of the world between two or more countries.

The aim is to find out whether the network connection to Google Public DNS are more or less equal around the world or not. The higher round trip time the worse the overall performance would be. Despite the round trip time is not the only performance parameter of DNS, it affects the performance greatly.

We configured the probes in certain locations to traceroute to 8.8.8.8. The result of this measurement is RTT from the probe to 8.8.8.8 and we would also be able to see the edge router to the Google Public DNS autonomous system from the probes.

3.4. Correlation Between Edge Router to AS15169 and the Origin of the DNS Query

By doing traceroute, we will be able to find out the edge router of a certain autonomous system. This is also valid in the case of doing traceroute to 8.8.8.8. We will be able to know the edge router to AS15169 of Google Public DNS.

The reason of this experiment is to get more information on how Google Public DNS handles the incoming DNS query inside their own autonomous system. This is also to confirm the claim of Google that they are using anycast to route the packet to the closest Google Public DNS.

The probes are the source of traceroute and sending DNS queries to one of the authoritative name servers. The measurements is done at a certain short time interval during a certain period of time. Then the result of the traceroute and the query log of the name servers will be compared to see if there is any correlation between them.

Our aim here is to find out if a packet to 8.8.8.8 is routed through a certain edge router, whether or not the DNS query from the same source will be handled by or routed to the same Google Public DNS server.

3.5. Global Cache Analysis

To determine if Google Public DNS maintains a globally shared Level 2 cache or not, which addresses the second research question, we followed 4 steps:

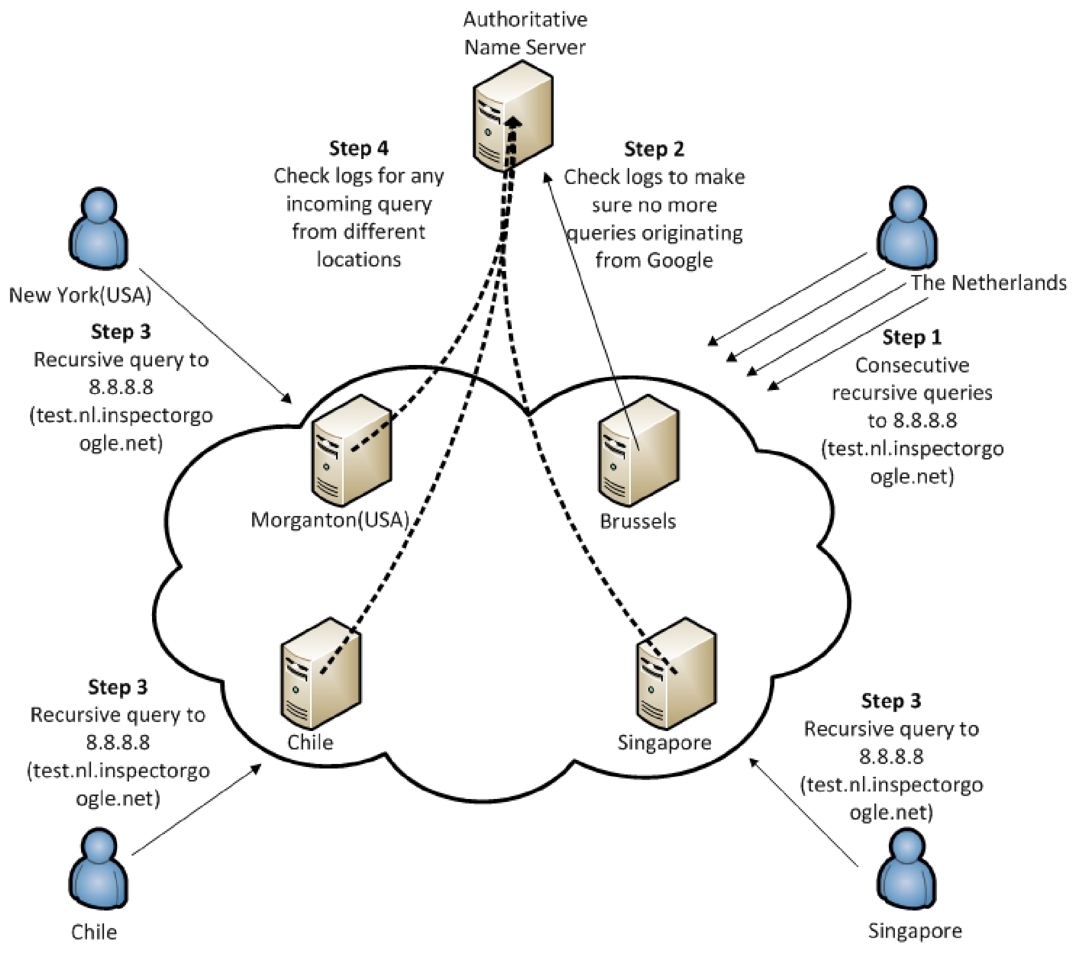
1. We let Google Public DNS service, say the public DNS server in Brussels, cache an A record that is administered by the authoritative name server located in the Netherlands, say test.nl.inspectorgoogle.net. To achieve this, the same queries originated from a client in London that is served by the public server in Brussels location. To make sure that the RR is certainly cached, consecutive queries are done per one second within the TTL of the A record until no more query logs are shown in BIND logs. By this means, we can assume that the A record is stored at least in Level 2 cache in Brussels.

2. Since Google Public DNS may maintain a globally distributed cache database, it may take a while to deliver the cache entry at different geographical areas. To overcome this uncertainty, the clients in different locations wait for different amount of time (the range is between 1 minute and default TTL value in different experiments) before doing queries from different continents.

3. After the waiting time, the client in USA (served by Morganton), the client in Singapore (served by Singapore) and the client in Chile (served by Chile) initiate queries for the same A record, test.nl.inspectorgoogle.net.

4. Meanwhile, we traced the BIND logs of the authoritative name server to check if any incoming query is logged originating from the Google resolvers in USA, Singapore and Chile.

The topology in figure 3.2 illustrates the 4 steps mentioned above.



Authoritative Name Server
Step 4
Check logs for any incoming query from different locations
Step 2
Check logs to make sure no more queries originating from Google
New York(USA)
Step 3
Recursive query to 8.8.8.8 (test.nl.inspectorgoogle.net)
The Netherlands
Step 1
Consecutive recursive queries to 8.8.8.8 (test.nl.inspectorgoogle.net)
Morganton(USA)
Brussels
Step 3
Recursive query to 8.8.8.8 (test.nl.inspectorgoogle.net)
Chile
Singapore
Step 3
Recursive query to 8.8.8.8 (test.nl.inspectorgoogle.net)
Chile
Singapore

Figure 3.2.: The topology of the steps

3.6. TTL and BIND Log Analysis

One concrete way to figure out how Google Public DNS maintains the Level 2 cache for unpopular domain names in a certain location is to observe BIND logs and TTL values in DNS responses received by the client, simultaneously. Our aim here is to present a technique addressing the research question 2d: Does Google Public DNS maintain a coherent Level 2 cache in a single location?

A process built by two python programs is used to fulfill this duty. Figure 3.3 shows the basic working principle of the flow. The first program originates DNS queries from the client (step 1) and parses the TTL value in the response real time (step 2). If the TTL is equal to default TTL of the RR, the program then parses the BIND logs to check whether it receives a DNS query from one of the Google resolvers and records the IP address of the resolver and the response TTL. If the TTL value is not equal to default

TTL, which also implies the query is responded by the cache, then it records only the TTL value (step 3). Second program analyzes the output of the first program and sets a cache ID to each response containing the default TTL value. Moreover, it relates the cache responses with a previously set cache ID based on the decrement of TTL (step 4). Hence, we are able to match the responses with the related cache ID. Finally, process waits for a certain period and be ready to originate new queries (step 5).

A sample output of the process with default TTL of 300 and query interval of 10 seconds is shown in Table 3.2. The first and second queries seem to be answered from different caches since both receives default TTL value, thus both queries have different cache IDs. The third query and fourth query can be associated with first and second queries, respectively, based on TTL decrement.

Query ID	Timestamp	Cache ID	Google Resolver IP	TTL
1	01:50:02	1	2a00:1450:400c:c05::153	300
2	01:50:12	2	74.125.181.83	300
3	01:50:22	1	Cache Response	280
4	01:50:32	2	Cache Response	280

Table 3.2.: A sample output of TTL and BIND Log analysis

A simple scenario can be described as follows: The authoritative name server in London is selected as the vantage point which is queried by Brussels location. The A record to be queried is set to test.uk.inspectorgoogle.net. The name server itself maintains the client role, thus the process runs in the same physical machine.

The concerns and measures regarding this analysis are as below:

1. Since Google Public DNS may carry out different policies in different locations, 4 authoritative name servers are selected as vantage points which are located in London, New York, Chile and Singapore, each gets queries from different Google Public DNS locations, Brussels, Morganton, Chile, Singapore, respectively.

2. By the reason of a potential change in the cache behavior at different hours of the day, time of day should be taken into consideration.

3. The RRs with higher and lower TTL values may change the behavior. Therefore, the experiments with different TTL values should be carried out.

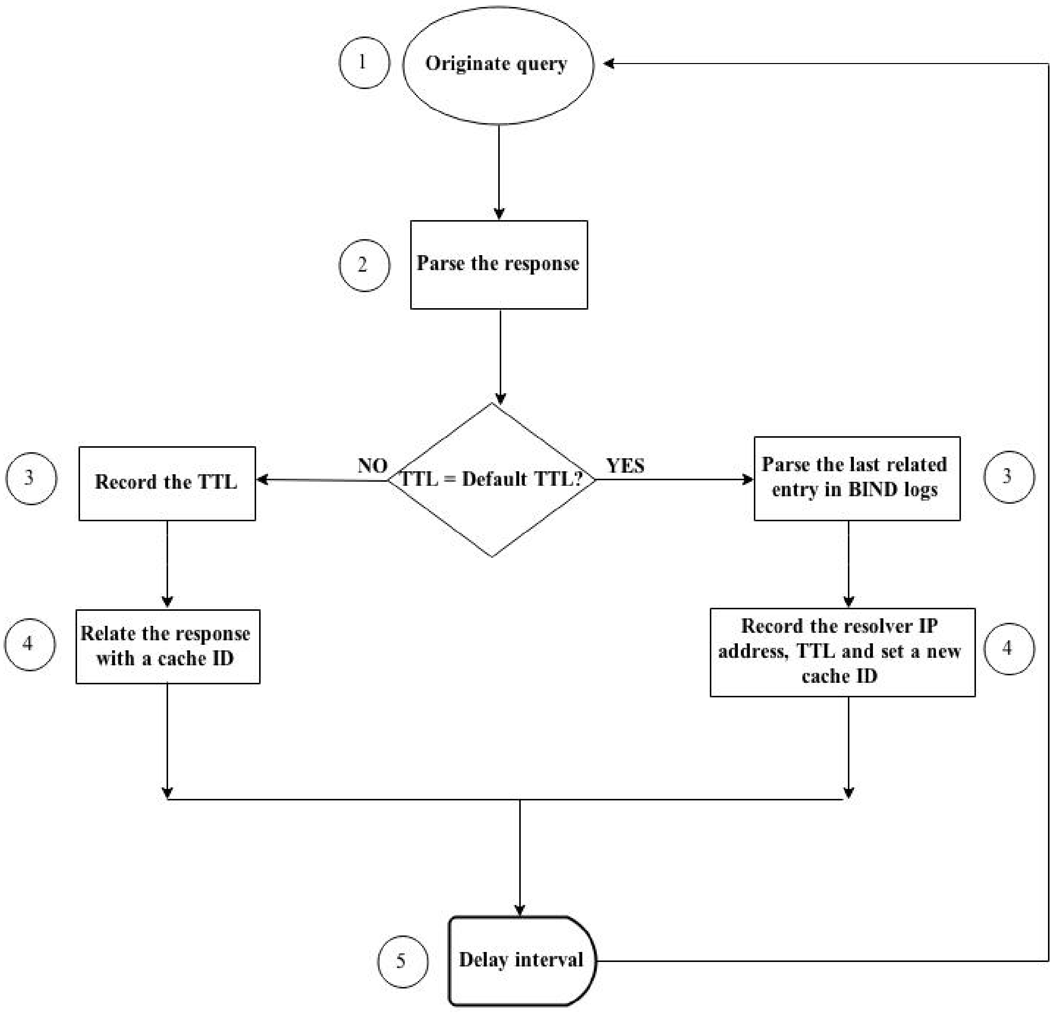

1
Originate query
2
Parse the response
TTL = Default TTL?
NO
YES
3
Record the TTL
Parse the last related entry in BIND logs
3
4
Relate the response with a cache ID
Record the resolver IP address, TTL and set a new cache ID
4
5
Delay interval

Figure 3.3.: Flow of TTL and BIND Log Analysis

4. Results and Implications

4.1. Origin of the Queries

We chose 50 countries distributed as even as possible. One probe is randomly picked from each country. A DNS query to one of our name servers is configured to be sent from these probes. The authoritative name server used here is the one in USA which is authoritative for us.inspectorgoogle.net. This authoritative name server is randomly picked and there is no any influence to the result of this research.

A part of the result can be seen in table 4.1. We can see that the origin of the query is coming from Google Public DNS closest to where the query was sent instead of the authoritative name server. The complete result is available in the appendix D.

Probe Location	Origin of The Query
Bangladesh	Singapore
Saudi Arabia	Belgium
Indonesia	Singapore
Algeria	Belgium
Russia	Finland

Table 4.1.: Probe location and the origin of the query

It gives us a hint that indeed there might be no globally single shared cache. As mentioned in the previous chapter, had there been a single globally shared cache around the world, it should have different result. We would expect that the query will originate from the Google Public DNS server close to the authoritative name server. It would indicate that the DNS query is routed through Google autonomous system and there is an internal process handling the query inside the Google network.

4.2. Round Trip Time

We chose two regions that relatively differs in terms of internet connection and also the number of Google Public DNS servers located in the region. Southeast Asia and Western Europe were the chosen regions. Because of the limitation of RIPE Atlas probes, we decided to only choose five countries from each region. In each country, we set five randomly picked probes to do traceroute to 8.8.8.8.

The result in table 4.2 shows that the average round trip time in Southeast Asia is higher than in Western Europe.

Country Name	Average RTT (in ms)
Indonesia	17
Philippines	45
Vietnam	40
Singapore	3
Malaysia	64
The Netherlands	5
France	3
Germany	2
Switzerland	2
Luxembourg	25

Table 4.2.: The comparison of RTT between Southeast Asia and Western Europe

The result is arguably really limited. We could not determine the quality of the internet connection in which the probe connected to. We also did not have a lot of samples because of the credit limitation. Nevertheless, it still gives an indication that the use of Google Public DNS in the country with less number of distribution of Google Public DNS servers has clearly some performance penalty. It could be better to use local DNS provided by ISP provider since it is closer to the client location and therefore resulted in a way less lower round trip time.

4.3. Correlation between Edge Router to AS15169 and the Origin of the DNS Query

The same setup as the previous measurement in section 4.2 was used. We chose the same five countries in Southeast Asia and five in Western Europe to examine. The same five probes each countries were also picked. In addition of a traceroute to 8.8.8.8, we also set up the probes to send a DNS query to our authoritative name server in USA (us.inspectorgoogle.net). In the end, we correlated the result of the traceroute with the BIND query logs to identify the edge router and eventually analyzed the origin of the incoming queries.

From all of the countries, we observed an interesting result from Malaysia. The DNS queries from other countries always went through the same edge router (based on the network where the probe is connected to) and also handled always by the same Google Public DNS server. However, the DNS query sent from the RIPE Atlas probe in Malaysia had a different behavior. The packet went always through the same edge router (also in Malaysia) but the DNS query was handled by a different Google Public DNS server in Singapore and Taiwan interchangeably in an undefined pattern.

It indicates that although each probe is always routed through the same edge router, the DNS query from the same probe could be handled by a different public resolver. This may imply that Google has its own mechanism to handle incoming queries. The

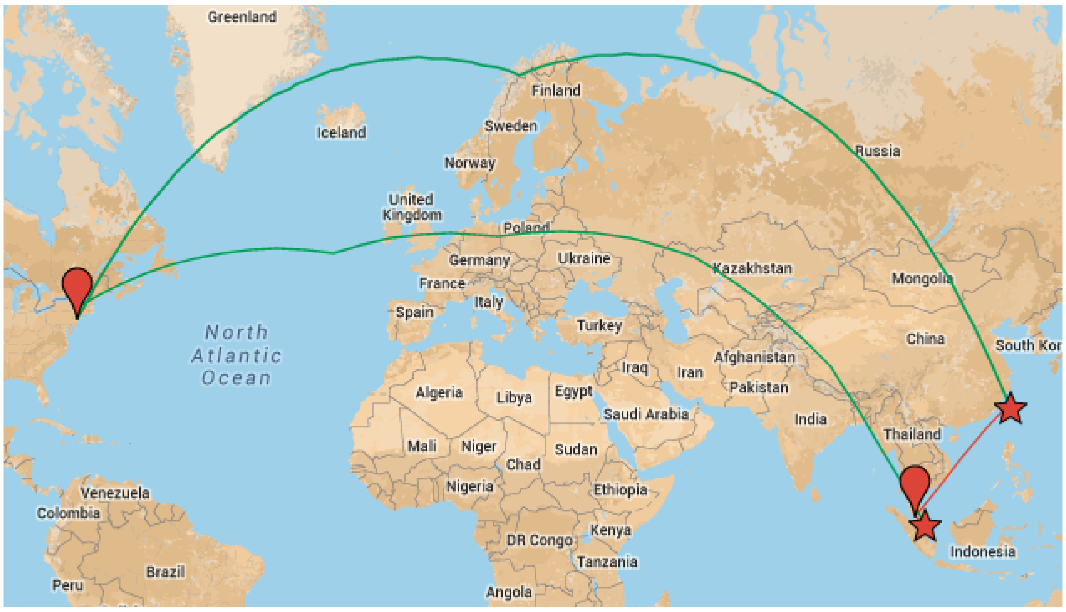


Figure 4.1.: Two Google Public DNS server handled DNS query from Malaysia

measurement result that shows DNS query from Malaysia is shifted to Google Public DNS server in Taiwan also indicates that this mechanism could lead to a performance penalty.

4.4. Globally Shared Cache

After following the steps indicated in the section 3.5, no sign is found regarding the existence of a single shared cache. Even if we let Google cache the RR sending queries to Brussels locations, we observed that authoritative name server receives queries from each Google data center independently. A possible delay of data synchronization across the locations are taken into consideration, however, even after hours no indication of a shared cache is detected. The same scenario is applied targeting different Google Public DNS server locations, however, the result did not differ.

4.5. Level 2 Cache Coherency in a Single Google Location

In order to investigate the Level 2 cache coherency in a single Google location, the method is applied described in the section 3.6. Our first finding was that Google does not manipulate the TTL values set by authoritative name servers unless it is more than 6 hours, addressing the research question 2c. Even if the TTL values were set to higher than 6 hours (eg. 12 hours), the maximum TTL value received in the DNS responses was 6 hours. For this reason, our experiments could have a maximum TTL of 6 hours. The results of an experiment with the default TTL set to 300 seconds and query interval to 10 seconds is shown in the appendix B. Four findings can be derived from this experiment:

1. Four queries were not responded by the cache since response TTL values are equal to default TTL and those queries were also shown in BIND logs.
2. The cache responses seemed to have responded by multiple caches as shown in "Cache ID" column.
3. The TTL values were decreasing gradually till zero.
4. The first and sixth queries were sent by the same resolver IP address.

The first two findings imply that Level 2 cache may be fragmented in a single location (Brussels in this experiment) as opposed to what Google claims [3]. The third finding shows that the RRs in Level 2 cache seem not evicted by Google, as we can observe the TTL values decreasing gradually till 10 seconds. This also may point out that Level 2 cache is big enough that can keep our records until TTL expires, in turn, strengthen our implications. At first sight, the fourth finding creates an impression of NAT usage, whereas those addresses are IPv6. As discussed in section A, Google stated that egress IP addresses are shared by multiple resolvers. This made a possible mapping of resolver IP to cache not applicable.

Another interesting observation during the experiments was that some cache responses had a TTL value that could not be related to any cache ID, labeled as UNKNOWN_SOURCE in appendix C. Despite this, those responses can be related to each other depending on the decrement of the TTL, such as; queries with id 7 and 24. We name such a case as "Ghost Cache". In addition, more than one ghost cache can be detected within a TTL interval. Two ghost caches can be seen in appendix C labeled as UNKNOWN_SOURCE1 and UNKNOWN_SOURCE2. However, we do not have a satisfactory explanation for the ghost cache.

Tests were performed in consideration of the concerns mentioned in section 3.6. The same behavior and similar results were observed in different locations of Google Public DNS (New York, Chile and Singapore), TTL values (300, 600, 1800, 3600, 7200, 21600 seconds) and time of day. During the tests, the DNS clients were configured in order to do recursive queries to 8.8.8.8 and 8.8.4.4. No queries were sent to the IPv6 addresses of Google Public DNS.

5. Conclusions and Future Work

We explored the locality and cache coherency of Google Public DNS service. Our experiments showed that recursive queries to an authoritative name server originate in the Google data center where the query is received. This means that DNS traffic is not routed within Google cloud, instead, each public resolver handles the DNS lookup by its own throughout the Internet. Actually, origin of the queries was a hint of that Google does not have a single globally shared Level 2 cache (Level 1 cache could not be analyzed due to technical limitations discussed in section 1.2). Our further analysis, using clients and authoritative name server in different continents, proved that Google stores different Level 2 cache in each location.

Tests and observations using the methodology described in section 3.6, posed a possible Level 2 cache fragmentation in contrast with the Google's aim[3]. As Google states cache fragmentation decreases the cache hit rate, in turn, increases the client latency. Our implications are hints to a possible performance penalty. Since there are currently privacy discussions on Google DNS, if such a study is done that comparing the performance of Google DNS and Local ISP resolvers and proving that local resolvers are performing better, then it would be another reason against using Google DNS service. Still very limited information is revealed by Google. Thus, future investigations would probably need more clues especially on the load-balancing strategy and the cache levels. The ghost cache issue discussed in section 4.5 is still open and gives an impression that there exists a complex inner mechanism, even more than two levels of cache.

Acknowledgements

We are grateful to our supervisor Roland M. van Rijswijk-Deij from SURFnet for his support and guidance throughout this project. We also thank Google DNS Team and Daniel Quinn from RIPE for their cooperative approach.

References

[1] Actual Number of Internet Users. http://www.internetlivestats.com/internet-users/. Accessed: 5-2-2015.

[2] G Huston. The Resolvers We Use. https://labs.ripe.net/Members/gih/the-resolvers-we-use. Accessed: 5-2-2015.

[3] What is Google Public DNS? https://developers.google.com/speed/public-dns/. Accessed: 8-1-2015.

[4] Performance Benefits. https://developers.google.com/speed/public-dns/docs/performance. Accessed: 5-2-2015.

[5] Huang C., Maltz D. A., Greenberg Aal., Jin Li. Public DNS System and Global Traffic Management, 2011.

[6] Jung J., Sit E., Balakrishnan H., Morris R. DNS Performance and the Effectiveness of Caching, 2002.

[7] Schomp K., Callahan T., Rabinovich M., Allman M. On Measuring the Client-Side DNS Infrastructure, 2013.

[8] P Mockapetris. DOMAIN NAMES - CONCEPTS AND FACILITIES. RFC 1034, Internet Engineering Task Force, November 1987.

[9] P Mockapetris. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. RFC 1035, Internet Engineering Task Force, November 1987.

[10] The most widely used Name Server Software. https://www.isc.org/downloads/bind/. Accessed: 5-2-2015.

[11] RIPE NCC. https://www.ripe.net/. Accessed: 5-2-2015.

[12] RIPE Atlas - The Credit System. https://atlas.ripe.net/docs/credits/. Accessed: 5-2-2015.

[13] RIPE Atlas Probes Location. https://atlas.ripe.net/about. Accessed: 5-2-2015.

[14] Geographic Implications of DNS Infrastructure Distribution. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_10-1/101_dns-infrastructure.html. Accessed: 5-2-2015.

A. Reflection

In this section we reflect on our experiences during the research project and what we have learned from this.

At the beginning of the project, since our main question was related to cache coherency we began with analyzing the BIND logs while originating queries to 8.8.8.8 from RIPE probes. Then we realized that, we first needed to find the answer of locality related question. Because we were unable to make sense of IP addresses shown in BIND logs. Before observing application layer (DNS), we should have started with network layer behavior. That costed us unnecessary workload.

After we figured out that there might not be a globally single shared cache, we contacted Google DNS team and verified our implication.

Since the Flush Cache tool has a bug during our project as mentioned in section 1.2, we changed our focus entirely to Level 2 cache. First we thought that resolver IP addresses might give a clue about the cache machines behind. However, we observed that our name servers receive queries from the same resolver IP address occasionally. We asked Google if they use NAT and they responded as multiple resolvers share the same egress IP address. Thus, our plan of mapping resolver IP to cache failed.

Even if Google did not tend to share information when it comes to technical details, they were friendly and willing to help during our correspondences.

We also had a credit limitation while using RIPE Atlas probes. The probes are manageable by using credits and each measurement needs certain credits. For example, one DNS query costs 20 credits and one traceroute measurement costs 30 credits. One of the reason of credits is to prevent any abuse to the system. Another limitation we have is regarding the maximum number of simultaneous measurements to a specific target. In our research we sent a DNS query through 8.8.8.8 and we could not do lots of measurement at once. After discussion with Daniel Quinn from RIPE, we got special arrangement for our project and therefore can use RIPE atlas probes without this limitation.

B. Results of the TTL analysis in a Single Location

Query ID	Timestamp	Cache ID	Google Resolver IP	TTL
1	01:50:02	1	2a00:1450:400c:c05::153	300
2	01:50:12	2	74.125.181.83	300
3	01:50:22	1	Cache Response	280
4	01:50:32	2	Cache Response	280
5	01:50:42	2	Cache Response	270
6	01:50:52	3	2a00:1450:400c:c05::153	300
7	01:51:02	2	Cache Response	250
8	01:51:12	2	Cache Response	240
9	01:51:22	1	Cache Response	220
10	01:51:32	3	Cache Response	260
11	01:51:42	4	74.125.17.209	300
12	01:51:52	2	Cache Response	200
13	01:52:02	2	Cache Response	190
14	01:52:12	1	Cache Response	170
15	01:52:22	2	Cache Response	170
16	01:52:32	1	Cache Response	150
17	01:52:42	1	Cache Response	140
18	01:52:52	2	Cache Response	140
19	01:53:02	2	Cache Response	130
20	01:53:12	1	Cache Response	110
21	01:53:22	1	Cache Response	100
22	01:53:33	2	Cache Response	100
23	01:53:43	4	Cache Response	180
24	01:53:53	1	Cache Response	70
25	01:54:03	2	Cache Response	70
26	01:54:13	1	Cache Response	50
27	01:54:23	2	Cache Response	50
28	01:54:33	2	Cache Response	40
29	01:54:43	2	Cache Response	30
30	01:54:53	1	Cache Response	10

C. Ghost Cache Sample

Query ID	Timestamp	Cache ID	Google Resolver IP	TTL
1	07:20:01	1	74.125.181.86	300
2	07:20:11	1	Cache Response	290
3	07:20:21	2	74.125.181.80	300
4	07:20:31	3	74.125.47.83	300
5	07:20:41	4	74.125.47.80	300
6	07:20:51	2	Cache Response	270
7	07:21:01	UNKNOWN_SOURCE1	Cache Response	250
8	07:21:11	4	Cache Response	270
9	07:21:21	2	Cache Response	240
10	07:21:31	4	Cache Response	250
11	07:21:41	2	Cache Response	220
12	07:21:51	2	Cache Response	210
13	07:22:01	1	Cache Response	180
14	07:22:11	4	Cache Response	210
15	07:22:21	1	Cache Response	160
16	07:22:31	3	Cache Response	180
17	07:22:41	2	Cache Response	160
18	07:22:51	1	Cache Response	130
19	07:23:01	1	Cache Response	120
20	07:23:11	3	Cache Response	140
21	07:23:21	1	Cache Response	100
22	07:23:31	4	Cache Response	130
23	07:23:41	1	Cache Response	80
24	07:23:52	UNKNOWN_SOURCE1	Cache Response	80
24	07:23:52	2	Cache Response	80
24	07:23:52	3	Cache Response	80
25	07:24:02	UNKNOWN_SOURCE2	Cache Response	90
26	07:24:12	3	Cache Response	60
27	07:24:22	2	Cache Response	40
28	07:24:32	UNKNOWN_SOURCE2	Cache Response	60
29	07:24:42	UNKNOWN_SOURCE2	Cache Response	50
30	07:24:52	UNKNOWN_SOURCE2	Cache Response	40

D. Probe Location and the Origin of the Query

Probe Location	Origin of The Query
Bahrain	Belgium
Bangladesh	Singapore
Bhutan	Belgium
China	Taiwan
India	Singapore
Indonesia	Singapore
Iran	Belgium
Israel	Belgium
Japan	Taiwan
Kazakhstan	Finland
South Korea	Taiwan
Malaysia	Singapore
Saudi Arabia	Belgium
Singapore	Singapore
Turkey	Belgium
Argentina	Chile
Brazil	Chile
Chile	Chile
Colombia	USA
Ecuador	USA
Paraguay	USA
Peru	Chile
Uruguay	USA
Venezuela	USA
Mexico	USA
USA	USA
Algeria	Belgium
Niger	Belgium
Tunisia	Belgium
Togo	Belgium
South Africa	Belgium
Madagascar	Belgium
Kenya	Belgium

Probe Location	Origin of The Query
Mauritius	Belgium
Cameroon	Belgium
Liberia	Belgium
Finland	Finland
Iceland	Belgium
Portugal	Belgium
Malta	Belgium
Bulgaria	Belgium
Russia	Finland
Ukraine	Finland
Estonia	Belgium
Germany	Belgium
Italy	Belgium
Tonga	Taiwan
Vanuatu	Taiwan
New Zealand	Taiwan
Samoa	USA