InHolland University of Applied Sciences

Bachelor Thesis

Geolocation of IP Resources

Author:
Emir Kaan Karadag

Supervisors:
Willem Toorop
(NLNet Labs)
Koos van Tubergen
(INHolland)

A thesis submitted in fulfillment of the requirements for the degree of Mathematics and Applications

in the

Department of Engineering, Design, and Computing

July 2015

"Pure mathematics is, in its way, the poetry of logical ideas."

Albert Einstein

Abstract

The ability to determine the geographical location of IP resources in the Internet is limited to a certain level of accuracy. Current techniques such as geodatabases and active-based measurements have limitations. While the problem of IP geolocation has been gaining the more research interest, there is still room for improvement. The aim of this project was therefore to improve the geolocation of IP resources by combining passive methods and active measurement-based techniques. The approach taken to answer the central research question is to make a systematic quantitative comparison between three datasets from different sources, cross- analyzing inconsistencies, and making corrections in each set by conducting an active measurements-based geolocation technique. There were in total 443 records corrected across the data sets. Although the research findings are not groundbreaking, through the derivation of methodologies, it was shown that how combining multiple data sets from different sources can help in correcting the falsely estimated locations.

Acknowledgements

I wish to express my sincere thanks to the NLNet Labs team who have provided me a place in their company where I had the opportunity to learn and grow professionally.

I would like to express my gratitude for my supervisor, Willem Toorop for his continuous support of my research, for his patience and immense knowledge. Without his guidance this project wouldn't have been possible.

Contents

List of Figures

List of Tables

Abbreviations

ICT	**I**nformation and **C**ommunication **T**echnology
CISPA	**C**yber **I**ntelligence **S**haring and **P**rotection **A**ct
PoP	**P**oint **o**f **P**resence
AS	**A**utonomous **S**ystem
RTT	**R**ound **T**rip **T**ime
NNDS	**N**earest **N**eighbor in **D**elay **S**pace
CBG	**C**onstraint **B**ased **G**eolocation
TBG	**T**opoogy **B**ased **G**eolocation
SOI	**S**peed **o**f **I**nternet
RIR	**R**egional **I**nternet **R**egistry
IANA	**I**nternet **A**ssigned **N**umber **A**thority
ICANN	**I**nternet **C**orporation for **A**ssigned **N**ames and **N**umbers
ISP	**I**nternet **S**ervice **P**rovider
RIPE	**R**éseaux **I**P **E**uropéens **C**oordination **C**entre
OIM	**O**pen **I**P **M**ap

To:

my family and loved ones...

Chapter 1

Introduction

1.1 Background

Today, Information and Communication Technology (ICT) has a key importance in our personal lives as much as in almost any economic sectors including but not limited to; health, education, production, logistic as well as many public administration. It is hard to understand whether the development of ICT sector has influenced other sectors to grow and rely on it or the exponential growth in other sectors and the need of communication has made ICT heavily dependable.(Rana & Panda, 2013)

Regardless of this, it can easily be said that our dependence on ICT has reached a level which was unthinkable less than a decade ago. Taking the Dutch ICT Sector as an example; in 2013 it had an annual turnover of €30 billion, providing 5% of the GDP. (Lundqvist, Apers, Smeulders, Huizer, & Mandersloot, 2012)

Consequently, the Internet is now considered to be one of the critical infrastructures and a failure or impairment of it could potentially lead to a shortage of supplies, disruptions to public order and other dramatic consequences. (Lewis, 2006)

The rise in the utilization of the computational and communication devices has been very drastic- especially in economically developed countries- for the last couple of decades. For example; for the Dutch Tax Administration; ten years ago for many of us, it was unimaginable to think that tax forms would be collected via Internet.

Although, its achievements are uncountable, it ise unwise to ignore its potential threats. The attack by the *Stuxnet worms* on a uranium enrichment facility in Iran in 2011 proves that these threats are getting more sophisticated and aren't only limited to PCs or laptops but rather they can affect the whole nation and jeopardize its critical infrastructures inevitability.(Lundqvist et al., 2012)

Thus, the countries are taking measures to protect their *cyber space*[1] . For example; *The Cyber Intelligence Sharing and Protection Act (CISPA)*[2] has been reintroduced by the US government in January 2015 after the recent *Sony hack*[3], the US government blames on North Korea. In 2013, a security company Mandiant has published a report (2013) claiming that a large portion of intrusions to the US critical infrastructures could be traced back to a Bureau located in Shanghai.

These examples show how *geolocation* is becoming an important factor in resilience and security of national critical infrastructures.(Koch, Golling, & Rodosek, 2013)

1.2 Research Motivation & Aim

NLnet Labs is a company that focuses on research and development of Internet standards, and initiates projects to deepen the understanding of the fundamental mechanisms and dynamics of the Internet.

The current state of IP geolocation, and in particular open-source IP geolocation information, is not sufficient for a number of use-cases and applications. By developing methods to improve the existing geolocation information, many users and/or use-cases can rely on correct and trustworthy geographical information related to IP addresses.

The aim of this thesis is to improve the location accuracy of existing approaches and try and overcome their shortcomings. The end product is expected to yield useful results which will be available to community for further researches.These results later can be used in a wider variety of applications and use-cases.

1.3 Research Questions

1.3.1 Central Research Question

How can the geolocation of IP resources be improved by combining passive (geodatabases) and active measurement-based geolocation techniques?

[1] As defined in the Oxford dictionary: The notional environment in which communication over computer networks occurs.

[2] CISPA is a proposed bill which would allow data sharing between US government and third-party technology and communication vendors. (HR 3523 as reported to the House Rules Committee)

[3] Sony hack was a release of confidential data belonging to Sony Pictures on November 24th, 2014.

1.3.2 Sub-Research Questions

1. What can be said of the current state-of-the-art in IP geolocation?
2. What existing methods/techniques are used for geolocation of IP resources?
3. What are the shortcomings of the existing methods/techniques of IP geolocation?

1.4 Research Approach

The approach taken to answer the central research question in section 1.3 is to make a systematic quantitative comparison between three datasets from different sources, cross-analyzing inconsistencies,and making corrections in each set by conducting an active measurements-based geolocation technique.

Sub-questions are answered by an extensive literature study that is presented in Chapter 2.

1.5 Document Structure

An in-depth literature review of geolocation and related topics is covered in Chapter 2. In Chapter 3 the preparation process before the measurements is explained. Chapter 4 describes how the passive measurement is carried out. Chapter 5 is devoted to the active measurement and the results obtained from the active measurement. A short summary of the whole project is given in Chapter 6. The findings of this research are evaluated in Chapter 7. Finally, future research and improvements are proposed in Chapter 8. This is followed by the conclusion in Chapter 9.

Chapter 2

Literature Review

2.1 Introduction

This chapter provides the necessary background information regarding the current state of art in IP geolocation. First, in section 2.2 an overview of Internet Topology is given. The basic knowledge and concepts of geolocation are introduced in section 2.3, followed by the application and use cases in section 2.4. Active and passive measurements are discussed in sections 2.5 and 2.6, respectively. The chapter is finalized by introducing the method for IP addressing along with organizations who are responsible for it in section 2.7.

2.2 Overview of Internet Topology

The network topology can be described as "The representation of the interconnection between directly connected peers in the network. There are three different levels at which to describe the network topology: the *link layer* topology, the *network layer* topology, sometimes referred to generically as the *internet topology*, and the *overlay topology*." (Donnet & Friedman, 2007, p.57)

Below each layer is described briefly:

2.2.1 Link Layer

The link layer topology represents the physical connections between different mediums e.g. switches, routers, bridges and etc. in the communication network. It is quite crucial

to have an extensive knowledge of the link layer topology of a network as it is the key to many critical network management tasks. (Breitbart et al., 2004)

2.2.2 Overlay Layer

The Overlay Layer is the layer where simultaneous sharing between dedicated peers is done. The torrent clients are built upon this so-called peer-to-peer system to maintain a file sharing network. Overlay Layer is beyond the scope of this project therefore wont be discussed in further details.

2.2.3 Network Layer

The network layer topology can be analyzed at four different levels. These levels are described by Donnet & Friedman as follows: (2007, p.58)

The IP interface level deals with the interfaces of routers. The second level, the router level topology can be obtained by indirectly (such as traceroute measurements) and each router considered to be a one-hop. The Point-of-Presence level is the hardest topology to be identified geographically. Lastly, AS level provides information on connectivity Autonomous Systems.

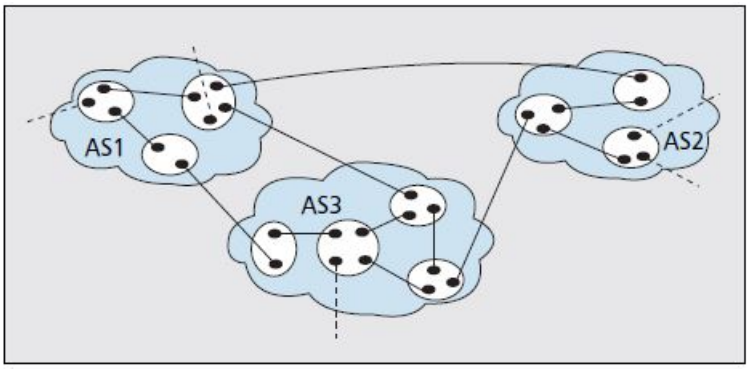

FIGURE 2.1: The different levels of Internet Topology

Source: (Donnet & Friedman, 2007)

As seen in Figure 2.1 depicts, three levels of Internet topology are present. AS level I represented as the blue clouds, the blank shapes are the routers and the black dots are the IP interface level. These are linked through plain or dotted lines. PoP level links, which are aggregated from their corresponding IP level edges, are not visible in this figure.

2.3 Overview of Geolocation

Geolocation is a method that uses a variety of resources and services to determine the geographic- physical position of an object (i.e. computer, mobile, person). In contrast to positioning systems that indicate the graphic coordinates, the result of geolocation services are mostly physical addresses – state, city, postal code or other specific data.(Parekh, Friedman, Tibrewala, & Lutch, 2004)

As it is stated by Anna Sainsbury - "It is important thing to remember about geolocation is that it doesn't refer to any one, particular type of technology. The reality is that there are number of different geolocation methods, each with its own particular strengths and weaknesses." (2013, p.33) There is no single correct method, rather it depends on what type of application that will benefit from it.

There are multiple methods that use different properties of computer networks that the IP geolocation can be derived from such as searching through domains measuring delay (latency) or via Wi-Fi. Geolocation methods can be divided into two;

- Passive Methods - IP, DNS, Wi-Fi
- Active Methods - Active probe measurements

To determine the position of the stations in Internet, one can use the following information: (Parekh et al., 2004)

- IP Address
  - IP database records (Whois, GeoIP, etc.)
  - Domain knowledge from DNS
- Information provided by web tools
  - Web browser language settings i.e. webmail, Facebook, etc.
  - Set time and time zone on the browser
  - Details of user's system derived from JavaScript, Adobe Flash and alike.
- Metadata stored in photos
- Payment information on debit/credit cards

Basic concepts of geolocation are explained as: (Katz-Bassett et al., 2006)

Target: A target is an Internet host with an unknown location.

Landmark: A host with known location.

Probe: A probe is a station whose main objective is to measure the desired information.

The main difference between the probe and landmark is their purpose i.e. the probe is intended to find a geographical location by conducting measurements meanwhile landmark is solely interested in geographical location. In other words, probes are used for measuring networking information (e.g. ping, traceroute, etc.) meanwhile landmarks are reference hosts with a well-known geographical locations. In this project, probes are a part of the networking tool, namely RIPE Atlas. Ping measurements will conducted using the probes. Landmarks are IP prefixes whose geographical locations are entered manually.

The state-of-the-art in IP geolocation is quite open to be advanced considering most of the time it is still impossible to pin point a host's location due to the complex structure of Internet. The continent granularity is at 100% as IP blocks are given out and recorded by RIRs in each region. The country level accuracy obtained by geolocation techniques is claimed to be around 96% to 98%. (Koch, Golling, Stiemert, & Rodosek, 2015). This number is more than 99% for the commercial geolocation databases. (Shavitt & Zilberman, 2011)

However, city-level and PoP-level (longitude and latitude) are too low to be completely trusted. The percentage is less than 60% (at the city level) and 20% (at postal) for the countries where Internet is densely present. The Appendix A and B respectively represent the countries and their corresponding accuracy percentage.

2.4 Geolocation Applications & Use-cases

Common uses of geolocation information include targeted local advertisements, real time content i.e. weather, local news or traffic information.(Shavitt & Zilberman, 2011) Companies, corporations and organizations globally want to make use of these resources to gain a better market share in their business.

For example; Google with its search engine will offer search results based on your location. *FourSquare*[1] will suggest restaurants that are within proximity of your area. Facebook

[1]Local search and discovery service

will display advertisements from their customers' surroundings rather than the other side of the globe. In contrast, real time applications such as *Songza*[2] , *Vudu*[3] or *BBC iPlayer*[4] will limit their content to a specific region due to security issues.(Bendale & Kumar, 2014)

Perhaps one one of the most important application of geolocation is the fraud detection and other criminal investigations. Many criminal activities e.g. phishing attempts can be potentially prevented with use of geolocation. For example; when a credit card of a person who is a resident of Amsterdam, is being used in the other side of the world, the banks might suspect that the credit card is stolen and used by someone else than the owner. Then as a security measure, many banks contact the owner asking if the transaction is done by him/her. The banks use geolocation information to detect where exactly the credit card is used.

Using geolocation information might help to put restrictions to specific locations from where threats origin. Many sectors including, banking, political organizations can benefit. Moreover people can be protected from individuals with bad intentions.(Shavitt & Zilberman, 2011, p.2044)

There are also incidents caused by wrong geolocation information. An example is what happened to the Dutch entrepreneur in early 2015 when he was visiting U.S. for a business trip. He was interrogated by customs by U.S. authorities because he was thought to be traveling from Jordan, although he was never there. Apparently, the incident occurred because when he was completing his admission papers online using his phone, Vodafone - his SIM carrier - used an IP address from Jordan. Vodafone supposedly buys blocks of IP addresses from other countries and assigns them to their customers. The full article can be found here.[5]

As the geolocation information is used in more and more applications, the dangers and threats are becoming more common. This results in more strict policies by companies and regulations by the governments.

[2] a music streaming service (for North America)
[3] a movie delivery service (via Internet for USA)
[4] Radio and TV streaming service (for UK)
[5] http://www.at5.nl/artikelen/140265/ondernemer-verdacht-op-schiphol-door-ip-adres-van-vodafone

2.5 Active Methods

In this paper, geolocation methods are divided into two classifications: active and passive. Although there are approaches that have both active and passive components called hybrid, for clarity, these are also classified as active methods.

This section contains a number of selected active geolocation methods. The emphasis was placed on the methods that use Round-trip delay (RTT) measurements.

2.5.1 GeoPing

GeoPing method, is an active method of geolocation introduced by Padmanabhan, V. N., and Subramanian, L. in 2001. The position of the target is acquired based on the relationship between latency measurements i.e. RTT and the geographical distance.(Mandiant, 2013)

"GeoPing measures the delay to the target host from multiple sources (e.g., probe machines) at known locations and combines these delay measurements to estimate the coordinates of the target host." (Padmanabhan & Subramanian, 2001, p.177-178)

It uses *Nearest Neighbor in Delay Space (NNDS)* approach which can simply be explained as the delay between a certain landmark and the target is the same as the delay between another landmark and the target, then it must be that the landmarks must have the same (at least very similar) geographic positions.(Katz-Bassett et al., 2006)

The first step in this method is to build a map, so-called *delay map*. This can be seen below in Figure 2.2, each record contains the coordinates of the landmarks and a *delay vector*, $D_V = (d_1, \ldots, d_N)$ containing the measured (minimum) delay to the host.

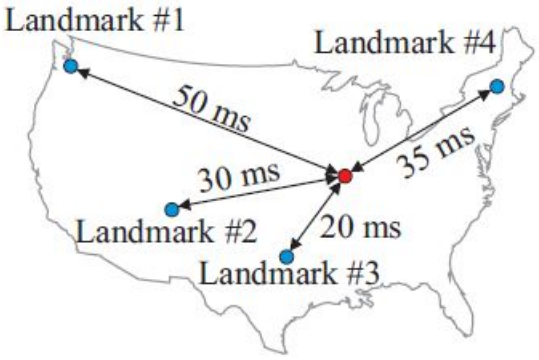

FIGURE 2.2: Functionality of GeoPing

Source: (Koch et al., 2013)

Two delay vectors created. First the previously mentioned delay vector which obtained by carrying out delay measurements to the landmarks from N probes at known location. Upon given a new target, the second one, $D'_V = (d'_1, \ldots, d'_N)$, is created by carrying out measurements from N probes to the target.

To have a better understanding, the visualization of the process can be seen in Figure 2.3. The second and third windows show the creation of the first vector and in the 4th window the process is completed by creating the second vector.

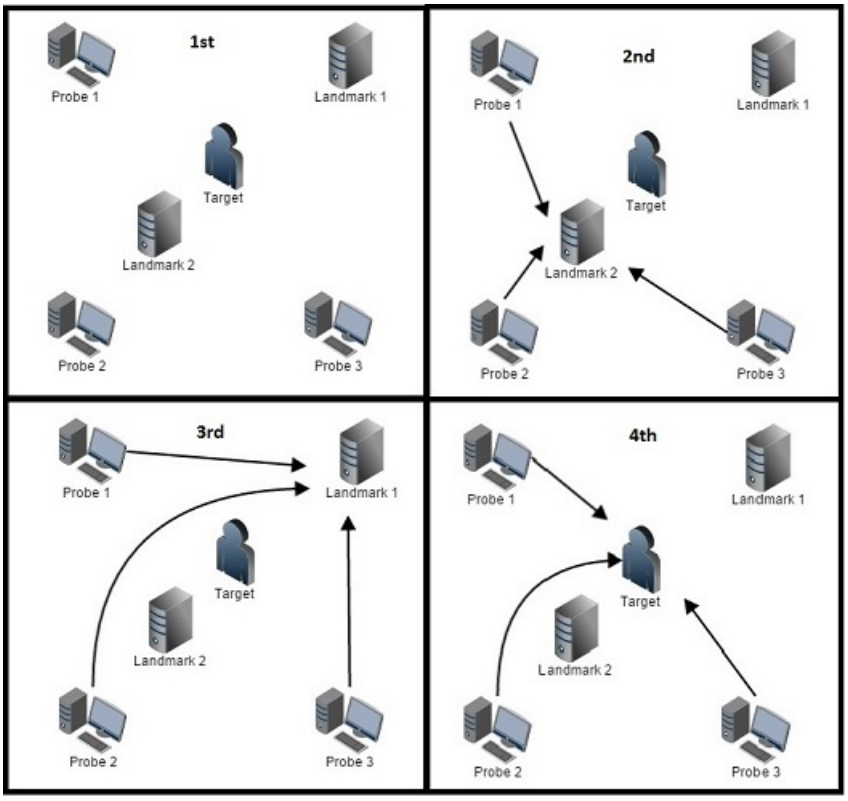

FIGURE 2.3: The Process of Creating Delay Vectors

Finally, searching through the delay map the best match of D_V and D'_V is made and the Euclidean distance formula, shown below in Equation 2.1, is used to estimate the location of the target.

$$E = \sqrt{(d_1 - d'_1)^2 + \ldots + (d_N - d'_N)^2} \tag{2.1}$$

As stated by Koch et al., "GeoPing is limited to a discrete solution space, which in this context means a concrete landmark and not a region." (2013, p.506) The disadvantage is in the fact that it is unable to localize the direct target. The results yield only the position of one of the reference points. To obtain more accurate results, a larger number of landmarks and the suitable deployment of probes are needed.

2.5.2 Shortest Ping

Shortest Ping is considered to be one of the simplest active geolocation methods.(Katz-Bassett et al., 2006) It is based on the principle of using the smallest RTT. The delay to the target from the reference points with the known locations (landmarks) is measured and the smallest value is set to be the position of the target.

Although, this method is much simpler than the previously mentioned GeoPing, Katz-Bassett, et al. argue that it performs considerably better. As stated in their research:

"The median error for the 53 targets with a latency less than 4 ms is 15 km; for the 75 with no latencies less than 4 ms, the median is 266 km. This observation also suggests why Shortest Ping is competitive with more complicated techniques."

(Katz-Bassett et al., 2006, p.78)

The disadvantage of Shortest Ping is similar to GeoPing the result is *discreet* – corresponds to the geographic position of one of the probes. The method works best if the landmarks are spread evenly. Otherwise; there is a large probability that the determined position of the target will likely to be incorrect.The problem is explained in Figure 2.4 visually:

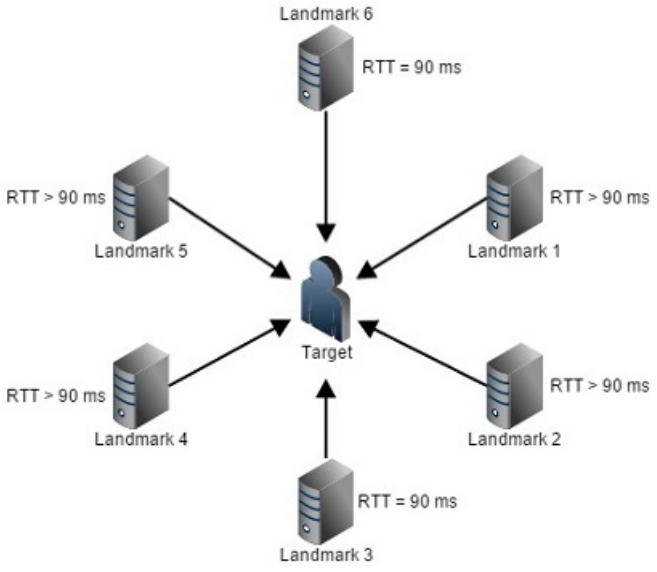

FIGURE 2.4: A bad scenario in Shortest Ping

There are six landmarks (Landmark 1 to Landmark 6) in Figure 2.4. Both *Landmark 3* and *Landmark 6* have the lowest RTT values to the target T (=90 ms). As it is

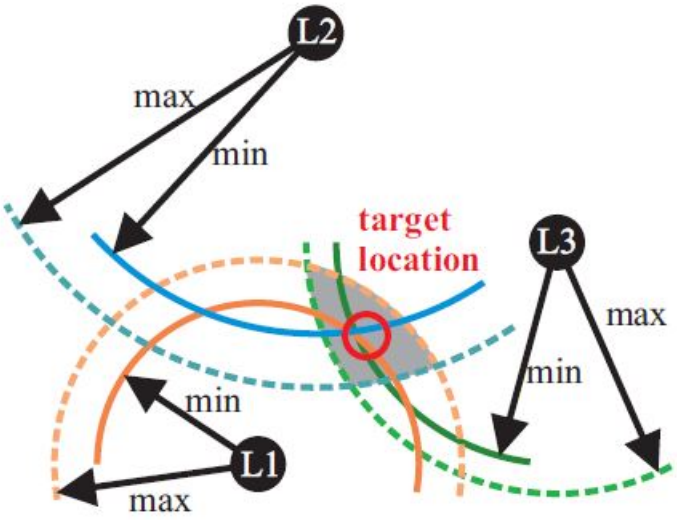

FIGURE 2.5: Multilateration used in Constraint Based Geolocation

Source: (Koch et al., 2013)

seen, these landmarks are distant from each other. Therefore; the position of the target will not be determined correctly instead it will most likely be around Landmark 3 and Landmark 6.

2.5.3 Constraint Based Geolocation (CBG)

This method overcomes of the limitations of two previously mentioned methods – GeoPing and Shortest Ping – which yield a discrete solution. Instead, it returns the position of the target in the form of a continuous area where the target may reside. This makes it possible to estimate the geographic location even for the targets that are relatively distant from the probes.

CBG is based on the principle of a triangulation-like approach (see Figure 2.5) where the physical location of the target can be estimated by calculating a sufficient number of distances or angles from certain hosts with known locations. If the distances are to be used, it is called *multilateration* and for angles, *multiangulation.*(Gueye, Ziviani, Crovella, & Fdida, 2006)

In practice, a signal that propagates at a constant speed from three landmarks to the target is transmitted and the time is recorded in order to find the minimum and maximum distance from each of the landmarks as seen in Figure 2.5.

> CBG then combines the distance estimates from all landmarks by intersecting all the circles. This intersection produces a feasible region in which the target is assumed to lie. The target is arbitrarily estimated to be located at the centroids of the region, and the size of the region is taken as a measure of the uncertainty (or confidence) in the estimate.
>
> (Katz-Bassett et al., 2006, p.78)

The accuracy of this method depends on the number of available landmarks. The biggest disadvantage of CBG is the fact that it uses ping-based measurements which give faulty results with firewalls, proxies and Intrusion Detection Systems.(Koch et al., 2013)

2.5.4 Other Active Geolocation Methods

Topology Based Geolocation (TBG), Octant and Speed of Internet (SOI) are the other geolocation methods which are variations of CBG bringing different aspects into equation in order to increase the accuracy rate of the results.

TBG takes topological aspects into account, meanwhile Octant uses a variety of geometric curves, known as Bézier curves. Speed of Light (SOI), on the other hand, deals with a different multilateration method to eliminate some complexity of CBG.

2.6 Passive Methods

Passive geolocation methods, in contrast to active methods, rely on the manually maintained static databases with records of IP addresses and their geographical positions. The location stored in the databases can be in the form of coordinates, as well as other type of information such as city, state or country where the IP addresses are estimated to be located.

The passive methods achieve different precision levels depending on which continent/-country where in the target is positioned. In countries where the Internet is considerably expanded, the geographic location of the host can be determined more precisely than where the Internet is scarcer i.e. less developed countries.

The biggest disadvantage of these methods is the dependence on the continuous updates. Also, since these databases are updated manually, they are likely to suffer having invalid/outdated records. Nevertheless, these databases are often used for their relative speed and reliability.

Passive geolocation methods can be divided into three different categories according to the information used to create the entries in the databases:

- Geolocation based on IP addresses
- Geolocation based on domains
- Geolocation based on Wi-Fi

2.6.1 Geolocation via IP Address

2.6.1.1 Databases of RIRs (WHOIS)

WHOIS is the best known public (free) database containing details of IP address registrations maintained by each Regional Internet Registries(RIRs)[6] in their regions.

WHOIS databases can be queried through the command line interface or many web-based tools can be found on Internet to perform lookups. Below in Figure 2.6 an example of a WHOIS output is shown:

```
[emir@dicht ~]$ whois 192.16.197.229
                    :

inetnum:        192.16.197.0 - 192.16.197.255
netname:        CWI-NET-197
descr:          CWI
                Amsterdam
                The Netherlands
country:        NL
admin-c:        CWI1024-RIPE
status:         LEGACY
remarks:        For information on "status:" attribute r
-status-values-legacy-resources
tech-c:         CWI1024-RIPE
mnt-by:         AS1888-MNT
remarks:        rev-srv:      ns1.cwi.nl
remarks:        rev-srv:      ns2.cwi.nl
created:        1970-01-01T00:00:00Z
last-modified:  2015-05-05T01:50:05Z
source:         RIPE # Filtered
remarks:        rev-srv attribute deprecated by RIPE NCC

role:           CWI CST
address:        Centrum Wiskunde & Informatica
                Science Park 123
                NL-1098 XG  Amsterdam
                The Netherlands
phone:          +31 (0)20 592 9333
fax-no:         +31 (0)20 592 4199
```

FIGURE 2.6: An example of WHOIS output

[6]Discussed in further details in the next chapter

Figure 2.6 shows the output of a query to "192.16.197.229" which translates into "nlnet-labs.nl".It can be seen the company address is shown along with many other information[7].

The biggest drawback of WHOIS is the inconsistency in its output which isn't standardized throughout the whole system. The formatting of the output within RIR is the same but differs across different RIRs. In addition, all information is presented as a text statement, making it relatively difficult for one to select a certain attribute e.g. only the address.

2.6.1.2 Geodatabases

These databases are logically divided according to geographical locations with corresponding IP blocks. Depending on the company they vary in prices and much like WHOIS service, these databases are dependent on a regular manual maintenance.

There are several companies in the market that offer this type of geolocation service. *Maxmind*[8],being one of them, is an American company founded in 2002 that has a product, namely GeoIP2 (successor of GeoIP) which is offered at different levels (country level & city level). It is said to be in the mid-price range. (Shavitt & Zilberman, 2011)

The company also offers a free version which is called GeoIP2 Lite. According to the company the Lite version "is comparable to, but less accurate than, MaxMind's GeoIP2 databases". In this paper, GeoIP2 Lite City will be used as one of the geodatabases.

IP2Location[9] is another company that offers geolocation services to their clients based on their needs. Much like Maxmind, they offer open-source databases for developers free of charge. IP2Location DB5 Lite is the other geodatabase that is used in this thesis. The motivation for these choices is given in Chapter 3 when the data sets are formally introduced.

Also characteristics of both of these databases will be discussed in further details in Chapter 3

[7] Other irrelevant information has been excluded
[8] www.maxmind.com
[9] www.ip2location.com

2.6.2 Geolocation via DNS-LOC

Another way to identify the position of a target is an estimation on DNS-LOC records. A LOC record as it is described in experimental protocol RFC 1876 (Davis, Dickinson, Goodwin, & Vixie, 1996) contains longitude, latitude, and attitude information about a host along with three optional values for the size of the host and precision information. A simple DNS-LOC record can be seen below in Figure 2.7:

```
LOC record statdns.net.   IN LOC   52 22 23.000 N 4 53 32.000 E -2.00m 0.00m 10000m 10m
```

FIGURE 2.7: An example of a DNS-LOC Record

Source: (Wikipedia, 2015)

2.6.3 Geolocation via Wi-Fi

Another geolocation technique used by companies such as Google to create Google Maps' geodatabase is via Wi-Fi. Much like GPS system, it is possible to determine the position of the device that is connected to Internet via Wi-Fi by triangulation, which uses the delay time and electromagnetic field which generates an access point.

All the relevant information about access points around (signal strength, SSID, MAC address) is retrieved and sent to Google Maps' database as soon as the client is connected to Wi-Fi.

2.7 Method for IP Addressing

An ordering scheme to assign IP addresses in a logical fashion is essential for a properly functioning network, including the Internet. Thus, several international organizations were founded to keep the network operating smoothly and efficiently. These organizations are not only responsible for the organization of IP addresses but also for managing and coordinating Domain Names, AS Numbers and Protocol Assignments in their regions.(Carpenter, Roberts, & Baker, 2000; IANA, 2015)

2.7.1 Distributions of organizations

The supreme body of assigning these Internet parameters is The Internet Assigned Number Authority (IANA), which is currently governed by a non-profit organization Internet Corporation for Assigned Names and Numbers (ICANN) based in California, USA.

ICANN oversees the operation of the IANA and subordinate organizations and as it is stated in their bylaws one of their main core values is "preserving and enhancing the operational stability, reliability, security, and global interoperability of the Internet".(ICANN, 2008)

IANA thus creates the range of numbers of IP addresses, domain names, etc. and assigns them its subordinate organizations, which are generally referred to as a regional RIRs (Regional Internet Registries). The territory of each RIR covers a large area, mostly a continent.

The world is divided in such a way that each region is managed by regional registries. In each particular region larger groups of local registers, which are primarily Internet Service Providers (ISPs), telecommunication companies and large companies who are interested in telecommunications are involved financially in the activities of regional RIRs. The distribution of regions is shown in Figure 2.8

FIGURE 2.8: Worldwide Division of regions

Source: (Wikimedia, 2015)

As seen in Figure 2.8, the world is divided into five RIRs, as such:

- African Network Information Centre (AfriNIC)
- American Registry for Internet Numbers (ARIN
- Asia-Pacific Network Information Centre (APNIC)
- Latin America and Caribbean Network Information Centre (LACNIC)
- Réseaux IP Européens Network Coordination Centre (RIPE NCC)

2.7.2 RIPE NCC

Réseaux IP Européens Network Coordination Centre (RIPE NCC) is an international non-profit organization, functioning as a regional Internet registry for Europe, the Middle East and some parts of Asia (see Figure 2.8). More detailed list of countries can be found on the website of RIPE NCC.[10].

In addition to the distribution of IP addresses, domains, etc. it also offers publicly accessible RIPE database. It is stated by RIPE NCC that it "contains registration details of IP addresses and AS numbers originally allocated by the RIPE NCC. It shows the organizations that hold the resources, where the allocations were made, and contact details for the networks." (RIPE-NCC, 2015c)

2.7.2.1 RIPE Atlas

RIPE Atlas is an active Internet measurement network developed by RIPE NCC in late 2010. It helps users to measure Internet connectivity and reachability. There are thousands of probes in the RIPE Atlas network distributed around the globe that could be used to conduct active measurements (e.g. Ping, Traceroute,DNS, SSL, and NTP)

"RIPE Atlas probes are small, USB-powered hardware devices that hosts attach to an Ethernet port on their router via a network (UTP) cable. The RIPE NCC provides probes free of charge." (RIPE-NCC, 2015d, p.p. 9)

Anybody interested can volunteer to host individual probes by plugging a registered probe into their router.

In Figure 2.9, the locations of probes can be seen:

It can be noticed that most probes are located in Europe. The reason is that RIPE NCC has most of their members in that specific region. (RIPE-NCC, 2015d)

Built-in measurements that can be conducted by the Atlas probes are :

"

- Current uptime, total uptime and uptime history
- RTT (round trip time) measurements (on IPv4) to the first and second hops (think about the first two lines in your outgoing traceroutes)
- Ping measurements to a number of predetermined destinations

[10]The list of countries covered by the RIPE NCC: https://www.ripe.net/membership/indices/

FIGURE 2.9: The locations of Atlas Probes

Source: (RIPE-NCC, 2015d)

- Traceroute measurements to a number of predetermined destinations
- DNS queries to root DNS servers (others to come)
- SSL queries to a number of predetermined destinations more measurement types may be added in the future.

”

(RIPE-NCC, 2015a, p.p. 18)

One of the data sets that will be used in this thesis is the data set containing the locations of the Atlas probes along with their IP addresses. These locations are entered by the probe-owners and are therefore error prone. That's why a significant amount of time was invested in picking out the most-likely errors off this data set. This process will be explained along with the motivation for choice in further details in Chapter 3.

2.7.2.2 OpenIPMap

OpenIPMap(OIM) is a open-source prototype of a network discovery and mapping tool that maps IP addresses and hostnames to geographical locations by combining multiple sources of data. It uses RIPE Atlas probes to create traceroute measurements and builds on underlying API to access crowd-sourced information.

The project was initiated by Emilie Aben (RIPE NCC Amsterdam) and is being developed by volunteers in the Internet community. It aims to advance the state-of-the-art in router geolocation by improving the confidence level at the city and PoP level.

One of its objectives is to help users to visualize traceroutes so that they can compare routes or visually identify outliers. It can also potentially help with answering strategic network questions such as where is the most feasible to build out a network or how to reduce latency issues.

The following Figure 2.10 illustrates how OIM works in principle:

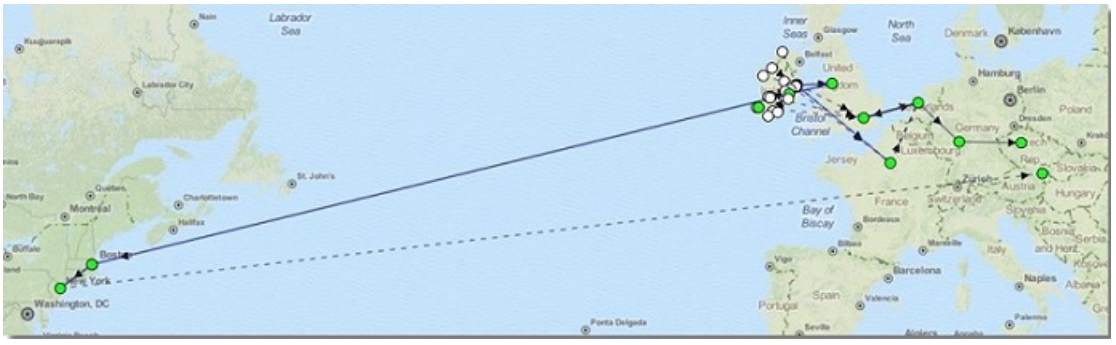

FIGURE 2.10: The functionality of OpenIPMap

Source: (Aben, 2015)

As mentioned before, OIM uses Atlas probes and visualizes traceroutes. The green dots seen in Figure 2.10 are the hosts that have been visited along the route from the probe to the target. It then creates a database containing IP addresses of these hosts matched with the estimated geographical locations along with its weight which indicates the likelihood.

> By combining information from all of those sources and appropriately weighing each data source we can come up with a list of likely locations for a given IP address, together with a score on how likely this is correct. For instance for a given IP address the answer could be:
>
> - 95% Bakel, The Netherlands
> - 4% Bakel, Senegal
>
> (RIPE-NCC, 2015b, p.p.6)

Chapter 3

Preparation

In the previous chapters, relevant and necessary information was given to have a better insight of this project. From this chapter on, the practical part of this project will be explained in further details.

The goal is to find a method to geographically locate IP addresses more accurately by combining passive and active methods discussed in the previous chapters. This will be attempted to achieve in two parts:

First part - The passive measurement:

Two well-known geodatabases, namely GeoIP2(by Maxmind) and IP2Location will be compared to the Atlas probes which are also refereed as landmarks. The reason why these specific databases were chosen because Maxmind and IP2Location are the two leading companies in the field of IP geolocation. The Lite version of these databases will be used because of the economical reasons as there was no certain budget set aside for this project. The reason why Atlas probes is used is because as mentioned in the previous chapter, Atlas probes are maintained by RIPE NCC users which form the biggest community in Europe and Middle region therefore it is considered to be the biggest source. Moreover, the reason why only two geodatabases (GeoIP2 and IP2Location) were chosen because all the other free-source databases wouldn't contribute to this project since they don't contain much different geolocation information than GeoIP2 and IP2Location.

The comparison between these data sets is the passive measurement that is mentioned in the research question.This comparison is made to find the inconsistent entries across all the data sets. For example; if for a specific IP address, there are three different location estimates scattered all around the world, then the location of this IP address is falsely estimated and should be evaluated further. If the location estimates for a specific IP address are geographically close to each other, then the location of this IP

address is estimated correctly and is not going to be taken into consideration for active measurements.

The end result of this comparison will be two tables (v4 and v6) filled with IP ranges and their estimated locations from the data sets. The details of this comparison will be explained in further details in the next chapter.

Second part - The active measurement:

The comparison made in the first part will yield results (inconsistent entries) that are worth for conducting active measurements. RTT measurements will be conducted on these entries to find which of the location estimates (coming from each data set) is the most accurate one. These RTT measurements are the active measurements that are mentioned in the research question. Measurements are done on the RIPE Atlas platform which will be introduced later in this chapter.

RTT measurements are done so that these inaccurate entries in the data sets can be corrected by calculating the round trip time which takes from the known sources to the target.For example; if there are three different location estimates for a specific IP address, each location estimate will be taken, geographically near probes to the each estimation will be found , RTT measurements will be done on these probes and after mathematical calculations, the most likely to be true estimate will be chosen as the correct estimation. The details of the RTT measurements will be discussed in further in Chapter 5.3.

The reason why Atlas is chosen as the measurement platform is because of two reasons. Firstly, it is the biggest network measurement platform in Europe and MIddle East region. Secondly, NLNet Labs could provide credits which are used to conduct these measurements on the Atlas platform.

3.1 Introduction

This chapters starts off with presenting the setup before starting to conduct measurements. In this setup section, data sets are introduced, the choice of programming language, network measurement tool and the communication devices are presented. This is followed by the section where the explanation is given how the data processing was done on each set.

The terms "probe" and "landmark" are used interchangeably throughout this document. Mostly, when in general context the term "landmark" will be used, "probe" will be seen when discussing matters in active measurements.It also important to clarify that when the term "database" in use, it is referred to GeoIP2 Lite and IP2Location Lite.

Sometimes, the full names of the might be omitted and used GeIP2 and IP2Location, they are still refereed to the Lite versions of the databases.

3.2 Setup

In this section of the chapter, the setup process will be explained in details. This process includes:

- Structure and folders
- Data sets
- The choice of programming language
- The network measurement tool

3.2.1 Structure and Folders

All the scripts that will be presented in the upcoming sections are written in Linux platform. Due to the complexity of operations, immense knowledge of Python language was crucial hence help of the company supervisor, Willem Toorop, was sought at times where needed. That's why a flawless communication was important therefore secure shell connection was created between two machines.

The scripts were contained the in root folder, /data folder had the data and /lib folder had two Python scripts, namely "utils.py" and "atlas.py", whose functions were shared across other scripts.

3.2.2 Data sets

The data sets that the passive measurements will be conducted on are GeoIP2 Lite City (by Maxmind), IP2Location Lite and Atlas probes.

GeoIP2 Lite City

This geodatabase is freely available to download on Maxmind's website. It offers a different versions depending on the information the user wants. For this project, City version was downloaded as a zip file which contained latitude and longitude information (unlike the Country version). The database is updated monthly.

A short script was written to download this file and can be found in the Appendix C.1. This script basically downloads the zip file and unzips to the destination folder for later use. The zip file contains two separate csv files; one for IPv4 and other for IPv6.

To have an overview a very small part of the database is also present in the Appendix I.1 and I.2. Also a website link is given for downloading the database directly.

Since the database contained more information than necessary for this project, a script was written in order to acquire relevant information (only IP ranges and its associated locations (latitude and longitude)) from the csv files. This script can be found in the Appendix D.1.

This script takes the unzipped csv files, creates a list, appends four columns to the list (ip_from, ip_to, latitude, longitude). At the end, it also creates a cPickle file for faster access for later use.

As it can be seen, the script calls some other function, namely "prefix2range" and imports "utils". These will be explained later in the data processing section of this chapter.

IP2Location Lite

The version of IP2Location geodatabase that is used in this project is called "IP2Location Lite DB5" and is free to download. The database is updated monthly.

A short script to download and unzip this geodatabase can be found in the Appendix C.2. Contrary to GeoIP2, this zip file contains a single csv file which contains both IPv4 and IPv6 addresses.

To have an overview a very small part of the database is also present in the Appendix I.3. Also a website link is given for downloading the database directly.

Similiar to GeoIP2, there is a script written to acquire only relevant information from the database for the comparison. This script can be found in the Appendix D.2. However, as it can be seen, unlike GeoIP2, this script doesn't call "prefix2range" function and doesn't imports "utils". This is because IP2Location already comes in a format that doesn't need as much processing as GeoIP2.

Atlas Probes

The last data set has a few differences than the the first databases introduced previously. Unlike the first two, it is not considered to be a geodatabase and is not maintained/updated as a whole. This data set contains information about the probes whose main purpose is to act as a part of a network measurement tool. Each probes information is

manually entered and updated by its owner. That's why the information is more reliable than the first two databases.

A script was written to download the relevant information for this project. It can be found in the Appendix C.3. As it can be seen, it imports the "atlas" file from the /lib folder. This file contains login information to the Atlas platform and will be explained under the section 3.2.4.

The main function of this script is to acquire the latitude, longitude, ipv4 and ipv6 addresses of probes which are connected or have been connected less than a week ago. It then creates a cPickle file for fast access.

3.2.3 Programming Language

Python was chosen as the programming language to write the scripts. The decision results from the benefits of using a language that is simple and offers a rapid development. Moreover, a native library called Pickle was used for object serialization. Object serialization is needed because of the following reasons:

1. Storing Python objects in datasets.
2. Persistence in the state of the data.
3. Faster development; caching and memorization.

3.2.4 Network Measurement Tool

To conduct the active measurements, Atlas by RIPE will be used. This network measurement tool, Atlas can be accessed through a web interface via atlas.ripe.com and requires to have a RIPE NCC account. Measurements such as ping, traceroute, and etc. can be made with credits purchased from RIPE NCC. The credits used for this project is provided by NLNet Labs.

The script to connect and conduct active measurements on this Atlas measurement tool is presented in Appendix E.1.

3.3 Data Processing

In this section, data processing is explained. This includes changing the format of IP addressing from prefix to range. This was done to have consistency to be able to compare

the data sets against each other. This is followed by the cleaning process done. Finally, the process how the landmarks were selected explained in the last section.

3.3.1 IP Prefixes to IP Range

The formatting of IP addresses in GeoIP2 Lite and IP2Location Lite differ. To compare the two, it was necessary to put them in same format. Therefore, IP prefixes were converted into a range (ip_from, ip_to) for the sake of efficiency.

The process as follows:

1. Strip off the mask
2. Convert it into an integer
3. Convert into a range (ip_from, ip_to)

A script was written to achieve this and it can be found in the Appendix E.2. As it can be seen there are two functions "ip2int" and "prefix2range". "ip2int" is called within the "prefix2range" function and it converts the IP addresses to integer value. "prefix2range"starts with stripping off the mask of the IP address, then calls the "ip2int" and function and lastly returns an IP range.

To have the process below applied to both IPv4 and IPv6 addresses, IPv4 addresses were mapped to have a IPv6 style address. These addresses are called IPv4-mapped IPv6 addresses. (Hinden & Deering, 1998). This process is done within the "ip2int" function.

The conversion was applied only to GeoLite2 and landmarks because IP2Location data set was already in the desired format.

3.3.2 Cleaning

There are different downloadable versions of both GeoIP2 and IP2Location. The difference lies in the variety of geolocation information that each database contains.

There are two versions of GeIP2 Lite that can be downloaded from their website, GeoIP2 Lite City and GeoIP2Lite Country. The one used in this project is the "City" version, the reason being is that the "Country" version doesn't contain longitude and latitude information.

IP-from	IP-to	Latitude	Longitude

TABLE 3.1: The Format

IP2Location offers a wider range of variety than Maxmind. The one that is used in this project is called DB5.Lite.

Since both of these databases contain information beyond the scope of this project, we have cleaned out the excessive information. As mentioned previously, this was done by writing two scripts which can be found in the Appendix D.1 and Appendix D.2. The attributes of the databases after processing and cleaning can be seen in Table 3.1:

3.3.3 Landmark Selection

The last step of data processing is the landmark selection. As mentioned before, the raw data set from the Atlas probes was used as landmarks to compare against the geolocation databases. The constraints for the landmark selection are as follows:

1. Only the active probes or, those that have been active within last week were selected as the intention was to have the most up-to-date information from the probes.
2. This IPv6 block ('2001:470::/32') belonging to Hurricane electrics was taken out of the data set. Because it is used for IPv6 tunneling. This means that even though the actual location of the IP addresses belong to this block, these show up scattered across globe. This might potentially alter the results of the comparison.
3. This IPv6 block ('2002::/16') that is a transition mechanism for migrating from IPv4 to IPv6. This system allows IPV6 packets to be transmitted over IPv4 addresses without tunneling. It was also picked out. (Carpenter & Moore, 2001)
4. The IP blocks that are associated with more than 2 probes were taken into closer consideration. If the average distance of these associated locations (latitude and longitude) were bigger than 500 km, these blocks were discarded. 500 km was chosen as the limit value because the locations within 500 km are considered to be in the same region.

There might be multiple probes scattered all around the world that have the same IP range. These probes are therefore not reliable. That's why a significant amount of time was invested in picking out the probes that might give incorrect results.

The operations mentioned above were accomplished by writing a script which can be found in the Appendix F.1. The script at the end creates a cPickle file filled with selected landmarks for the comparison.

Chapter 4

Passive Measurement

4.1 Introduction

In this chapter, the approach taken to answer the passive measurement part of the central research question defined in the Chapter 1, is explained in more details.

Firstly, the data sets will be discussed regarding the number of IP blocks that GeoIP and IP2Location databases don't have in common with the landmarks data set. This is important to know because then we know the number of IP blocks that wont be taken into consideration for comparison of data sets.Secondly, the process of how the IP range was made will be explained. This is followed by the section where an explanation is given for how the calculation of the distances between the estimated locations for the same IP block is explained. Lastly, the results of the passive measurement and analysis of the results will be discussed.

4.2 Data Sets To Be Compared

In this section, data sets are introduced again after they have gone through the data processing procedure explained in the previous chapter. As it can be seen three tables, Table 4.1, Table 4.2, and Table 4.3 are presented.

Table 4.1 contains the number of IP blocks in the landmark data set (both v4 and v6). As it is important to remember, landmarks in this project are the locations of IP blocks that are coming from the Atlas probes. So the values in the Table 4.1 show that how many IP blocks in the landmark data set are available for comparison.

Total no. of IP blocks	
6988	v4
2161	v6

TABLE 4.1: Total No. of Landmarks

No. of Missing IP Blocks	From
6	IP2Location
21	GeoIP2
1	Both

TABLE 4.2: Overview of the IPv4 Set

No. of Missing IP Blocks	From
212	IP2Location
4	GeoIP2
1	Both

TABLE 4.3: Overview of the IPv6 Set

The values represented in Table 4.2 and Table 4.3, are the number of IP blocks that are present in the landmark dataset but missing from the geodatabases (GeoIP and IP2Location).

To clarify, the numbers in these three tables represent the number of IP blocks.

It is safe to say that most of the landmarks are present in both databases with the exception being IP2Location data set is missing 212 IP blocks in IPv6 set of the landmarks.This means nearly 10% of the landmarks is missing from the IP2Location database.

The values in the tables were found because they give an overview of how many IP blocks will be compared on the basis of their locations estimated by different data sets.

4.3 Range Selection

Before advancing any further, it is important to explain how the IP ranges were selected. This selection was done by a script that is shown in Appendix F.2. This process was necessary because of the difference in how IP blocks were arranged in each data set. Consider the following Figure 4.1:

The Figure 4.1 is a visual representation of how IP blocks are arranged in each data set. Each rectangle represents a single IP block. As it is seen, the sizes of the rectangles are different as the size of the IP blocks may vary in each data set. This difference makes the distance calculation harder because the intention is to compare the same IP block across data sets that have different geographical locations. To overcome this difficulty, it

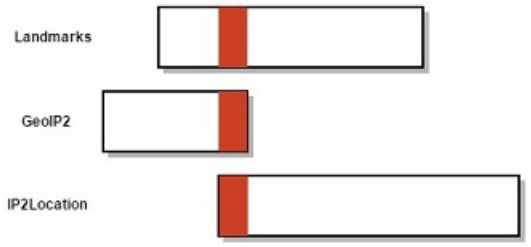

FIGURE 4.1: The process of IP range selection

was decided that the smallest intersection range was compared.This decision was made because the intention was to have the most specific results possible. Otherwise, the comparison couldn't be made. The red area in the rectangles represent the same IP range found in IP blocks in each data set.

To understand better, a possible scenario is given. For example:

- The range in the landmark data set is: 2 - 10
- The range in the GeoIP data set is: 0 - 5
- The range in the IP2Location data set is: 3 - 12

The range that will be selected is 3-5 as it is the intersection of all the three data sets. The numbers that are given in this example are much simplified to understand better how the IP range selection works.

4.4 Distance Calculation

In this section, the distance calculation between the geographical locations (estimated by GeoIP, IP2Location and landmarks for the same IP block) will be explained. There were two scripts written to accomplish this goal.

The script that the mathematical calculations was made can be found in the Appendix E.3.There can be seen the algorithm which was taken from a blog by John D. Cook to calculate the distance between two locations based on their coordinates. The explanation of the algorithm as follows:

To start with , the code returns the distance between two locations based on their longitude and latitude. The distance returned is relative to Earth's radius which is 6373

kilometers and 3960 miles. Since we want the distance in kilometers, it is multiplied by 6373. Therefore $p = 6373$

The θ (theta) coordinate is the longitude value which is the degrees east of the prime meridian. The ϕ is the latitude value which is the angle from the north pole down to the geographical location. These angles are converted to radians by multiplying by $\frac{2\pi}{360}$.

The arc is created by connecting any given two points in coordinates. The formula to calculate the arc ψ as it is seen in the code is :

$$\psi = arccos(sin\phi 1 sin\phi 2 cos(\theta 1 - \theta 2) + cos\phi 1 cos\phi 2)$$

The length of the arc ψ is $p\psi$. This is the great circle distance between two locations.

Lastly, the vectors of the locations and their respective coordinates relative to the origin at the center of the earth found. This is done by calculating the the angle between the two vectors and the arc by the formula as found in the script:

$$= sin\phi 1 sin\phi 2 cos(\theta 1 - \theta 2) + cos\phi 1 cos\phi 2$$

As it is mentioned previously, this algorithm gives the distance between two locations in spherical coordinates.

The second script to conduct this distance calculation can be found in the Appendix G.1. The main responsibility of this script is to create lists and fill them with values that are calculated with the first script mentioned previously. As it can be seen, another responsibility of this script is to find the missing IP addresses in each dataset. These numbers were presented in the Tables 4.1, 4.2 and 4.3.

4.5 Results & Analysis

This section is dedicated for discussing and analyzing the results obtained from the passive measurement. After successfully conducting the operations mentioned in the previous chapters, we have calculated the distances between geographical locations (estimated by GeoIP2, IP2Location and Landmarks for the same IP range). The results were put in two different tables (one for IPv4 and IPv6) which can be found in the Appendix J.1 and J.2. To explain the columns of these tables , a small example (10 entries) of the IPv4 table is shown in the Table 4.4.

Confidence	L > I	L > G	I <> G	ProbeID	IP Range
34965.2	17482.6	17482.6	0.0046	2819	('98.158.108.0','98.158.108.255')
31069.7	15534.8	15534.1	0.75	3978	('63.218.228.0', '63.218.228.255')
28623.7	14311.8	14311.8	0.0041	4403	('95.93.128.0', '95.93.131.255')
28341.9	9677.1	12749.9	5914.8	3976	('63.218.188.0', '63.218.188.255')
28260.3	14130.1	14129.4	0.75	3958	('63.218.170.0', '63.218.170.255')
28068.9	9067.8	13086.2	5914.8	3975	('63.218.150.0', '63.218.150.255')
27076.8	12170.1	12994.9	1911.7	3960	('63.223.1.0', '63.223.1.255')
26899.6	13449.6	13449.3	0.75	3963	('63.223.4.0', '63.223.7.255')
26894	13446.7	13446.5	0.75	3973	('63.223.4.0', '63.223.7.255')
26200.5	4.3	13099.2	13097	4980	('63.218.205.0', '63.218.205.255')

TABLE 4.4: Calculated Distances

FIGURE 4.2: A Low Confidence Example

For the sake of clarity, the first column will be explained later. The number in the second column in Figure 4.4, is the calculated distance between the location estimated by the landmark dataset and location estimated by IP2Location. The third column contains the distance between the location estimated by the landmark dataset and the location estimated by GeoIP2. The distance between the location estimated by the IP2Location dataset and the location estimated by the GeoIP2 dataset is present in the fourth column.

The number in the first column is the sum of all the distances mentioned. The sum was calculated because it provides a certainty level for a specific IP range. For example; if the confidence for a specific IP range is high, it means that the number in the first column for that IP range will be low and vice verse. In other words, the lower the number, the better the location of that specific IP range is estimated. To understand the confidence value better, consider the Figure 4.2 and Figure 4.3:

Figure 4.2 contains the locations estimated for the IP range '159.118.119.0'-'159.118.119.255'. The estimations for each specific data set are as follows;

- **Amstedam, Netherlands:** GeoIP2 dataset
- **California, United States:** IP2Location dataset

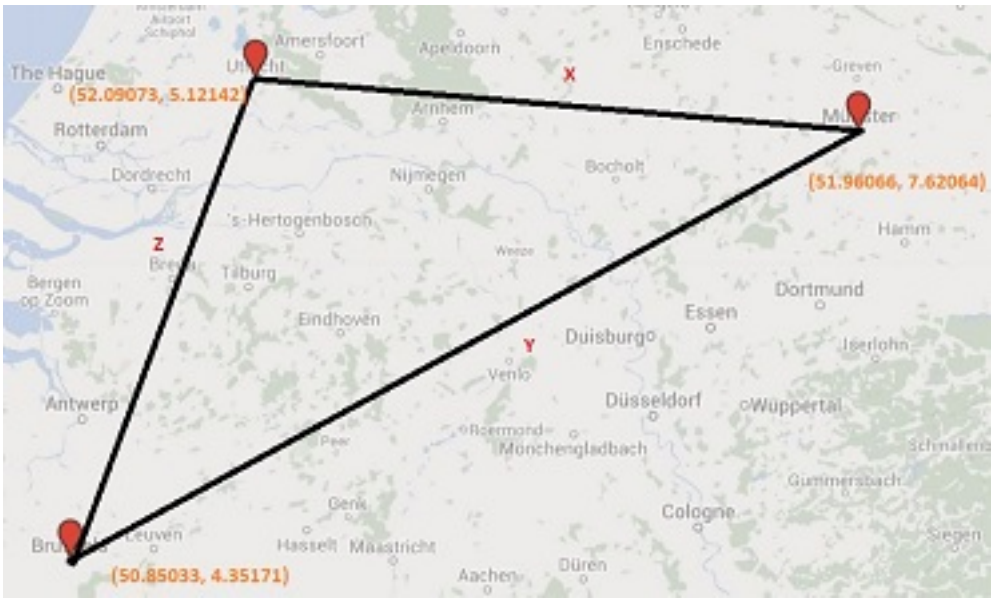

FIGURE 4.3: A High Confidence Example

- **Izmir, Turkey:** Landmarks dataset

As it can be seen, the locations that are estimated by each dataset are geographically far away from each other. if we were to calculate the distances between these locations are;

- The distance between Landmark and IP2Location (Y) : 10994 km
- The distance between Landmark and GeoIP2 (Z) : 2334 km
- The distance between GeoIP2 and IP2Location (X) : 8767 km
- The sum of all the distances: 22095 km

Figure 4.3 contains the locations estimated for the IP range '108.26.128.0'-'108.26.160.255'. The estimations for each specific data set are as follows;

- **Utrecht, Netherlands:** GeoIP2 dataset
- **Munich, Germany:** IP2Location dataset
- **Brussels, Belgium:** Landmarks dataset

The calculated distances between the estimations are;

- The distance between Landmark and IP2Location (Y) : 258 km

- The distance between Landmark and GeoIP2 (Z) : 148 km
- The distance between GeoIP2 and IP2Location (X) : 174 km
- The sum of all the distances: 580 km

As a result, it can be said that the IP range '108.26.128.0'-'108.26.160.255' has a higher confidence than the IP range '159.118.119.0'-'159.118.119.255'. This is because the sum of all the distances for the IP range '108.26.128.0'-'108.26.160.255' is lower than '159.118.119.0'-'159.118.119.255'. The idea behind this comparison is that if the sum of differences for a specific IP range is small then we are confident that this IP range is estimated more correctly than the IP range with a higher sum value.

The table 4.4 is filtered in a descending order for the confidence value. Because as mentioned before, the less confidence (the bigger sum) indicates that IP range is not consistent across the data sets. We are more interested in the IP ranges with little confidence as we will be trying to correctly estimate the location for these ranges.

The next two columns are the Probe ID; this complemented with the IP address range in the last column. It is important to remember that the Table 4.1 is a small example. The full tables can be found in the Appendix J.1 and J.2.

To show the distribution of the distances per IP range, two graphs 4.4 and 4.5 were created out of the Tables in Appendix K.1 and K.2.

As the legends in Figure 4.4 and Figure 4.5 read, the red line corresponds to the distance from landmark to GeoLite2, green is from landmark to IP2Location and blue is the distance between GeoIP2 and IP2Location.

The X-axis represents the number of probes that are compared against the databases and the Y-axis the distances in km. Y-axis is in logarithmic scale.

It can be seen that in Figure 4.4, the two databases follow the same trend for the IPv4 distribution of distances. Meanwhile for IPv6 in Figure 4.5, they tend to have greater variance locations for the same IP range. In other words, blue line in Figures 4.4 and 4.5 shows how much the databases GeoIP and IP2Location disagree.

There are 586 records that have distance of 500 km or more in both databases in IPV4. 2671 records with a distance of 50 km or more and around 6000 within a distance of 1 km for IPV4.

As excepted after seeing the Table 4.3, that IP2Location falls short in X-axis compared to GeoIP2. There are around 200 less records in IP2Location database that were compared against landmarks.

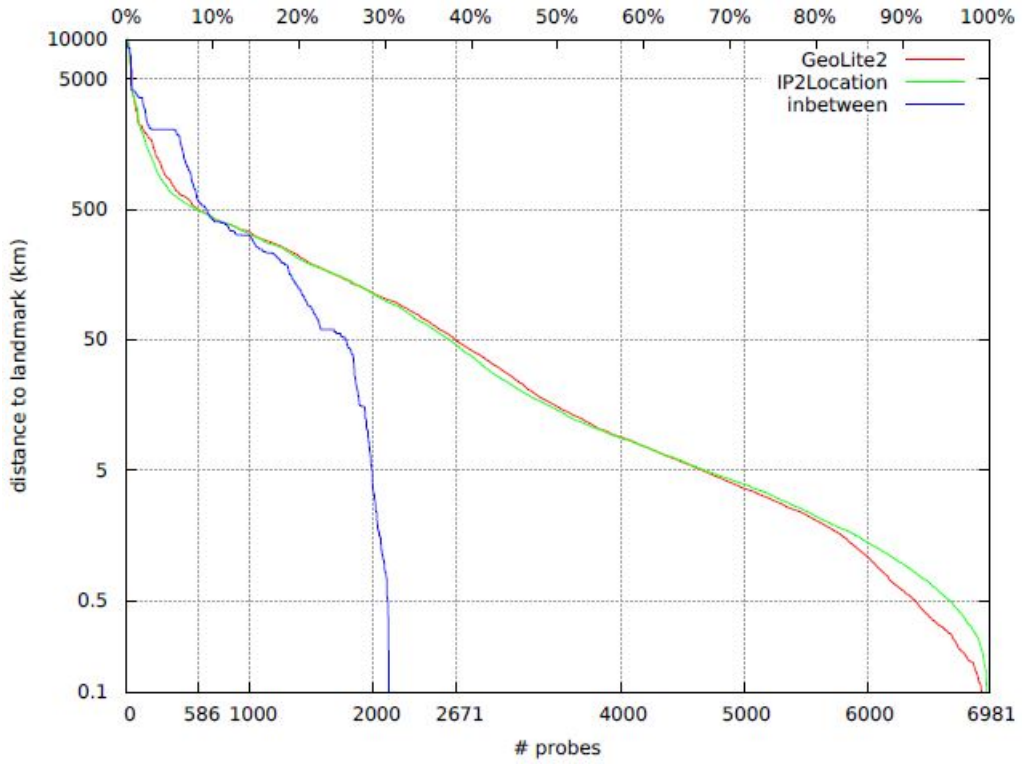

FIGURE 4.4: Distribution of distances IPv4

What is also interesting to observe that, the disagreement between two databases in IPv6 is far greater than IPv4. There are around 2000 out of 7000 in IPv4 and 2100 out of 2300 in IPv6. The explanation of this difference can be that IPv6 standard is a much newer standard that results in the resolutions of commercial databases differs more in IPv6.

At this point, the passive measurement has been completed. The next section, the active measurements will be conducted on the results that are derived from the passive measurement.

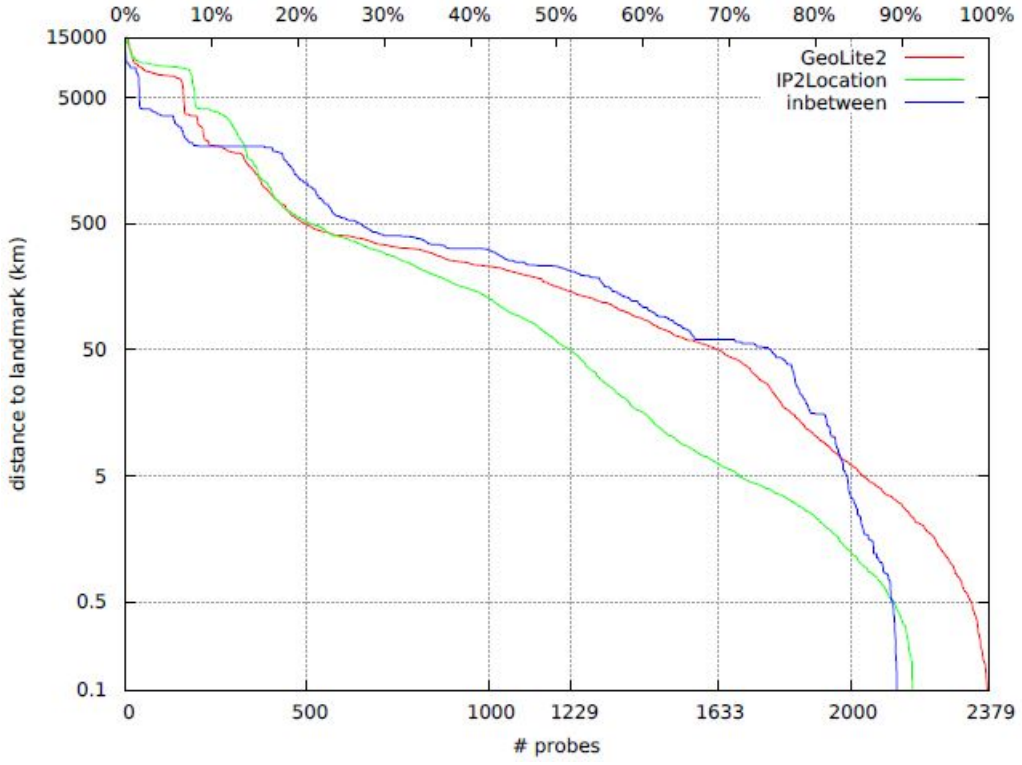

FIGURE 4.5: Distribution of distances IPv6

Chapter 5

Active Measurements

5.1 Introduction

In this chapter, the approach taken to answer the active measurement part of the research question will be explained in details.

As the active measurements, ping utility was used to measure the latency - how long it takes for one packet to get from A to B. A round trip time (RTT) is taken from each ping reply. The measurement is done by the local clock in the in the computer which is sending the ping request. It is calculated as measuring the time taken from when the request left to when the reply arrived.

Ping utility is chosen as the active measurement in this project, because the ping times give a good estimation of the distance between the source and destination machines. Moreover, ping is easy to apply and its outputs are easy to read and analyze.

Atlas by RIPE NCC was used in this project as the networking platform to conduct the ping measurements. The measurements were conducted on the selection of IP ranges that were obtained from the passive measurements. All the measurements are public and can be found on Atlas's web interface. The instructions on how to check these public measurements along with a small example of these measurements are given in the Appendix O.1 and O.2 . The ping measurements can also be referred to as RTT measurements from this point on, as RTT values of the ping measurements are the sole interest of this project.

5.2 Preparation

In this section, the preparation necessary before conducting the RTT measurements will be explained. This preparation consists of two parts. After these parts are explained, the results of the preparation is presented in the subsection.

The first part is the IP range selection. It is the selection process of IP ranges (obtained from the passive measurements) to conduct the RTT measurements on. This selection is necessary as we are only interested in the IP ranges that are inconsistent in their estimated geographical locations across the data sets.

Second part of the preparation consists of the probe selections. This part is the process of selecting the nearest probes to the estimated locations for each IP range that the measurements will be conducted on. This is an important step of the preparation because to achieve the best results for the RTT measurements, it is crucial to select the probes that are nearest the estimated locations.

There was a script written for the preparation before the RTT measurements. This script can be found in the Appendix G.2. The operations that are accomplished with this script will be explained in two parts in the subsections 5.2.1 and 5.2.2 respectively.

5.2.1 IP Range Selection

This process is done in order to make a selection of IP ranges to conduct RTT measurements on. The selection will be made from the lists that are obtained from the passive measurement. These lists can be found in the Appendix J.1 and J.2. This selection of IP ranges is necessary because not every IP range in these lists are considered to be worth looking further into. For example; it is not necessary to look into a specific IP range with high confidence (low sum value) since this IP range is considered to be located accurately since the estimated locations for it are consistent across the data sets.

The operations to accomplish this process are explained step-by-step as follows:

1. The lists,that are found in the Appendix J.1 and J.2, are iterated through from the top of the lists
2. The IP ranges that have more than 50 kilometers for their confidence (the sum value - first column in the list-) are selected.
3. The selected IP ranges are then placed into two new lists to schedule the RTT measurements on the Atlas network platform. As usual, one of these lists are for

IPv4 ranges and the other for IPv6. These lists can be found in the Appendix L.1 and L.2.

IP ranges were selected starting from the top of the lists because the IP ranges with a lower confidence level (higher sum value) are the ones that are most likely to be incorrect.

The reason for choosing 50 km as the threshold value is that research by Shavitt and Zilberman indicates that the locations fall in the range of 50 km within each other, are considered to be in the same city.

5.2.2 Probe Selection

This process is done in order to make a selection of probes to conduct RTT measurements. The selection will be made from the lists that are obtained from the passive measurement. These lists can be found n the Appendix J.1 and J.2. A certain number of probes that are closest to the estimated locations (by GeoIP2, IP2Location and Landmarks dataset) will be selected. Only for the IP ranges that were selected in the previous section 5.2.1, the nearest probes will be selected. The selection of probes is necessary for the preparation phase of the active measurement because by selecting the nearest probes, it is guaranteed to have the best results.

The operations to accomplish this process is explained step-by-step as follows;

1. The lists,that are found in the Appendix J.1 and J.2, are iterated through from the bottom of the lists. In other words, the lists are reversed (seen in the code, as "rdists" parameter)
2. The probe selection process starts by calling the "find_closest" function (as seen in the code in Appendix G.2. This function takes four parameters. The "threshold" and "amount" parameters along with latitude and longitude.
3. The "threshold" value is the maximum distance from the selected probes to the locations (estimated by GeoIP2, IP2Location and Landmarks for the same IP range). It is taken as 50 km.
4. The "amount" value is the maximum number of probes to be selected for a specific set of locations (estimated by GeoIP2, IP2Location and Landmarks for the same IP range). It is decided to be 50.
5. Another constraint is that the minimum number of probes to be selected for a specific set of locations (estimated by GeoIP2, IP2Location and Landmarks for the

IP Range	I	G	P	Confidence
63.218.228.166	16	16	46	31069.695224
63.218.188.58	50,	16	24	28341.890104
63.218.150.110	50,	16	33	28068.864704
63.218.205.106	11,	16	11	26200.544580
63.218.204.138	16,	16	11	26199.690087
63.223.16.22	16,	16	37	23354.991654
67.215.82.69	11,	31	31	22203.463553
195.10.43.85	42,	46	43	21396.265389
199.19.52.230	21,	21	37	20696.328868
195.10.43.93	42,	11	16	19230.297843

TABLE 5.1: Address & Probe Selection For RTT measurements

same IP range) is 10. It means that, there has to be at least 10 probes within the threshold value (50 km) for a specific set of locations in order for these locations to qualify for conducting RTT measurements on.

6. The number of probes which are qualified for RTT measurements is calculated.

7. The results are then appended into the two lists mentioned in the 5.2.1 (Appendix L.1 and L.2).

Probes were selected starting from the bottom of the list because the intention was to choose the closest probes to the IP ranges.

The maximum number of probes is chosen as 50 because it is the default value RIPE Atlas gives when conducting measurements on their network.

The minimum number of probes is 10 because it is estimated that any number of probes fall under will yield inaccurate results.

5.2.3 Preparation Results

In this section, the results obtained from the preparation before the RTT measurements are presented. These results can be found in the Appendix L.1 and L.2. To explain the results better, a small example of these tables are given in 5.1.

The list shown in Table 5.1, contains 10 entries of IPv4 ranges that are fit for RTT measurements. In the list, the IP range is shown as a single IP address, this is because ping request scheduled for a single IP address in RIPE Atlas. But it can noticed that the IP addresses in Table 5.1 are within the IP ranges of Table 4.4. As mentioned previously, the measurements were conducted on the lists which are found in the Appendix L.1 and L.2.

The first column is the IP ranges that the each data set has an estimated location for. The second column (I) represents the number of probes that are qualified for RTT measurements for the location estimated by IP2Location dataset. The third column (G) contains the number of probes that are qualified for RTT measurements for the location estimated by the GeoIP2 dataset. The number of probes that are qualified for the location that is estimated by the landmarks dataset is shown in the fourth column. The last column of the list contains the confidence (sum of differences) for the IP range. The list starts with the least confidence (highest sum of differences) because it indicates that that specific IP range is the most inconsistent across the three data sets.

410 IPv4 and 111 IPv6 ranges are determined to be fit for RTT measurements. The RTT measurements are done in order to find out which of the data sets have the best location estimate for the IP ranges. This process is explained in details in the next section 5.3.

5.3 RTT measurements

In this section, the process of RTT measurements will be explained in details. As mentioned previously, the networking platform to conduct the ping measurements is the Atlas by RIPE NCC.

There was a script written to schedule the ping measurements and do necessary calculations with RTT values to find out which of the data sets has estimated the most correct location for a given IP range. The script to schedule the measurement and calculating the average RTT can be found in the Appendix H.1. Also another script was written to find out which of the data sets have the best location estimate for a specific IP range and to calculate the improvement factor of the best dataset on the other datasets. In other words, how much the best dataset has improved the other data set's location estimates. This script can be found in the Appendix H.2. The operations that are accomplished by this script will be explained step-by-step, as follows;

1. An IP range is taken and set as the target for the ping measurement. (The IP ranges are taken from the first column of the lists in Appendix L.1 and L.2.
2. A new measurement on Atlas RIPE is created and given a "msm_id" which represents the unique number on the Atlas system to identify the each scheduled measurement.
3. First ping requests are scheduled to be sent to the location estimated by the IP2Location for that specific IP range with the selected probes. (IP2 Location because it is taken from the second column of the lists).

4. 3 ping requests are sent for each probe. For example; if the number of selected probes are 50 for a specific location, then the number of ping requests sent are 150.

5. The average RTT values are calculated for IP2Location.

6. The steps 3 to 5 are repeated for the other two data sets whose probes are in the third and fourth column. (Up to this point, these operations are accomplished by the script found in Appendix H.1).

7. Best RTT value is chosen and corresponding location estimate by its dataset set as the best source for the specific IP range. The process of choosing the best RTT value is explained in further details with a help of a figure because of its difficultness. Refer to the Figure 5.1 along with its explanation.

8. Improvement factor for the other two data sets is calculated separately as shown in the Formula 5.1. The corresponding RTT in the formula refers to the one of the other datasets (not the best estimate). The reason why the improvement factor was calculated was to show how much the other location estimates (not the best estimate) were bettered. As seen in the code, it can only take a value higher than 1 because if the factor were to be 1, it would mean there were no improvement made.

9. Improvements for each dataset are then put into separate lists that can be found in the Appendix M.1, M.2 and M.3.

10. It is also seen in the code that the results are put into two other lists, namely improvements and improvements_source. These lists are created in order to create graphs to show how much each dataset is influenced by others and how influential each data set was. This will be discussed in further details in the next section 5.4.

$$= \frac{Best_{RTT}}{Corresponding_{RTT}} \tag{5.1}$$

The Figure 5.1 describes the process of RTT measurements. The big black dots represent the locations estimated by the datasets. The small white dots that around the black dots represent the selected probes. By using the networking platform RIPE Atlas, the ping measurements were scheduled. The ping requests was sent from the source - Amsterdam - to the destinations (the probes around the estimated location by each dataset) on RIPE Atlas. The solid black arrows indicate that there are ping requests are sent from the source to the targets. The dotted lines are the expected RTT that should take in normal circumstances. These numbers are given for an example and might not reflect precise times.

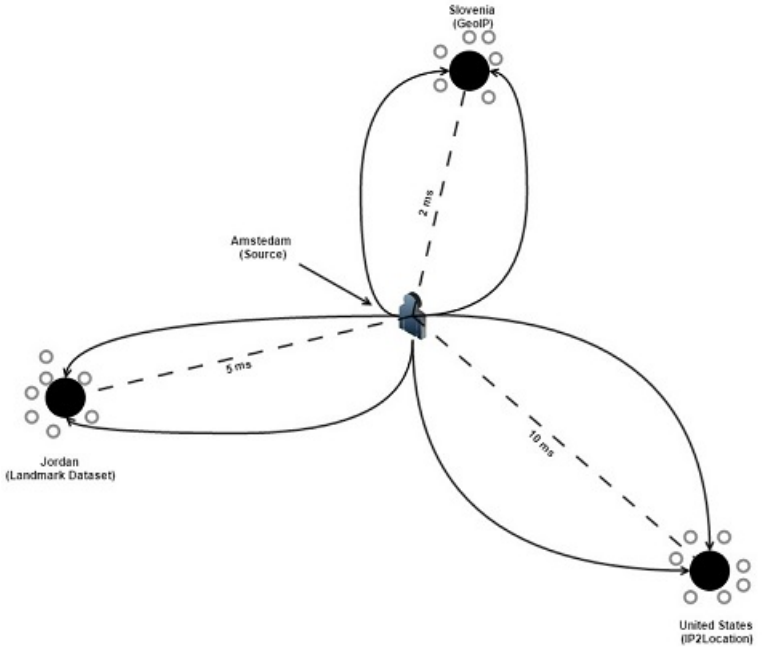

FIGURE 5.1: Process of RTT measurements

Since the Round Trip Time that would normally take between the source to the target is known, the best source is the time that is obtained from the measurements that is closest to the actual time. This is how the best RTT value is found that is mentioned in the 7th step. For example; as seen in the Figure 5.1, GeoIP2 dataset estimates that a specific IP range is located in Slovenia. If the measured RTT is much different that 2 ms then we know that the location estimated by the GeoIP2 will not be the best source.

The instructions how to see the measurements conducted for this project are given in the Appendix O.1

5.4 Results & Analysis

In this section, the results obtained from the active measurements will be presented and analyzed. As mentioned in the previous chapter, the results of the RTT measurements were placed in three different lists.O One for each data set; IP2Location, GeoIP2 and Landmarks Dataset. These lists can be found in the Appendix M.1, M.2 and M.3. Each of these lists contains the improved IP ranges along with information on how much improvement was made for each dataset.

IP Range	Coordinates	Best-Source	Factor	Distance (km)
281...6880 - 281...8159	(45.7675 , 4.8695)	P	1.67	392.8
281...4544 - 281...7311	(45.7775 , 4.8085)	P	2.26	389.6
281...9088 - 281...9343	(51.50949, -0.59541)	I	1.20	388.5
281...0144 - 281...1167	(51.8695,4.4475)	P	2.44	63.3
281...9296 - 281...9551	(51.60928,-1.24214)	I	1.3	77.8
281...5648 - 281...5903	(51.9225,4.4792)	I	1.81	18.3
425...4144 - 425...0319	(52.25,5.75)	P	1.46	59.3
425...3728 - 425...9903	(52.3740 , 4.88969)	I	1.38	60.08

TABLE 5.2: Improvements on GeoIP2 Data Set

Data set	No. of corrections
IP2Location Lite	169
GeoIP2 Lite	167
Landmarks	107
Total	443

TABLE 5.3: Number of Corrections

Table 5.2 contains a small example of the derived results from the RTT measurements.(The full lists are in the Appendix M.1, M.2 and M.3. This table is given to explain the each column of the in details.

The first column in Table 5.2 contains the IP range that the improvement was made. This was shortened for better clarity. The coordinates of the best estimated location for that specific IP range is given in the second column; this is followed by the source of these coordinates. "I" represents IP2Location, "G" is GeoIP2 and "P" is for Landmarks Dataset. Improvement factor is shown in the fourth column. The last column contains the distance between the best estimated location (by the best source) and the location estimated by the dataset that the improvement was made on (In this particular case GeoIP2 because this sample is taken from the Appendix M.1. To see the full results, refer to Appendix M.1, M.2 and M.3.

For example; refer to the first row in Table 5.2. For the IP range "281470768986880 - 281470768988159", it was found that landmarks dataset had the best location estimation. The coordinates of this best location is "(45.7675,4.8695)". The estimated location by GeoIP2 was bettered by 1.67. The distance between the estimated location by GeoIP2 and the best source location (In this case, landmarks dataset) was 392.8 kilometers.

Table 5.3 shows the number of corrections made in each database.

To see which data set was the most influential and which one was influenced the most by others, two graphs were drawn Figures 5.3 and 5.2. These figures were drawn from the lists that are found in Appendix N.1 and N.2.

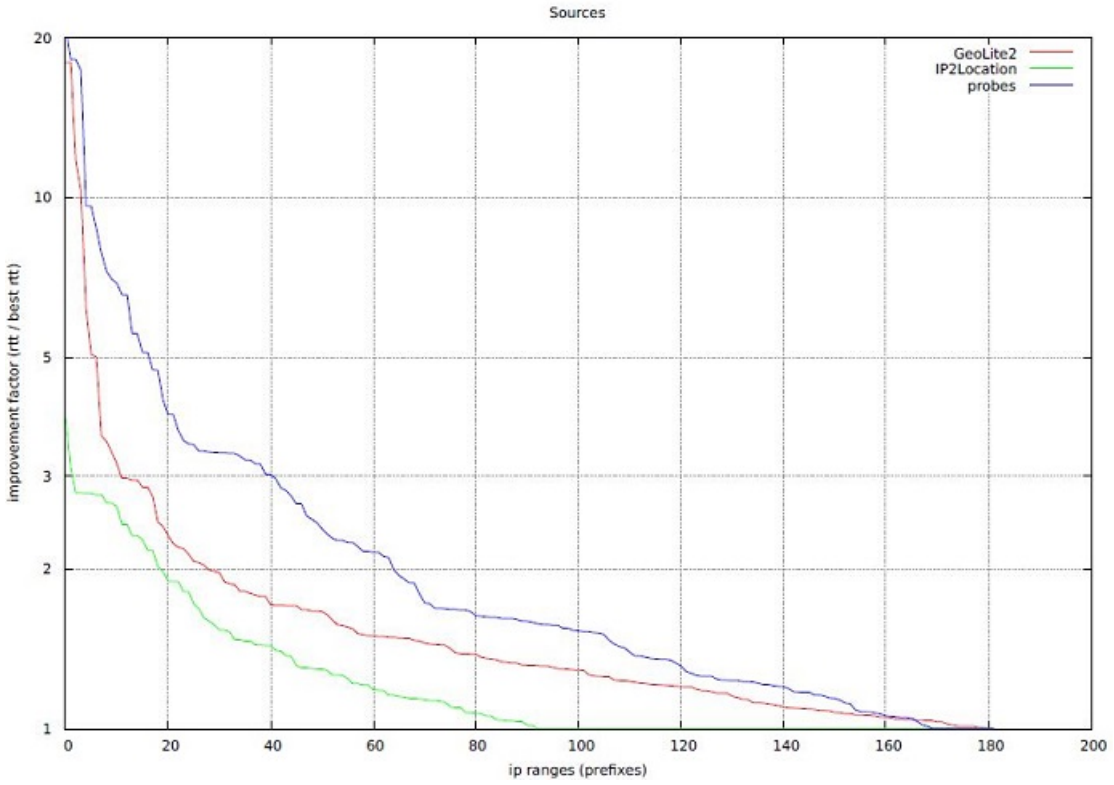

FIGURE 5.2: Influence on data sets

The first noticeable aspect of the Figure 5.3 is the corrected number of IP ranges. As it is mentioned previously in Chapter 5.3, there were 411 IPv4 and 111 IPv6 addresses. It makes a total of 522 addresses that RTT measurements were conducted on. Taking a closer look at the lists that the figures were based in order to calculate the precise number IP ranges that were corrected, it is observed that there are around 80 RTT measurements in total, that weren't effective. This is due to the fact that their improvement factor being was 1 and not shown on the figures.

The most influential data set, as expected, is landmarks. It can been seen in Figure 5.2 that it has the biggest improvement factor over the other two data sets. Although, it is worth to mention GeoIP2 has influenced more ranges with a little improvement factor. The least influential data set is IP2Location with the smallest factor and it fell short with the number of ranges it has improved compared to other data sets.

The graph 5.3 shows that the most influenced data set is IP2Location with only a slight difference compared to GeoIP2. The trend of both datasets is quite close but it can be seen the green line is mostly over the red line which indicates the degree of improvement factor.

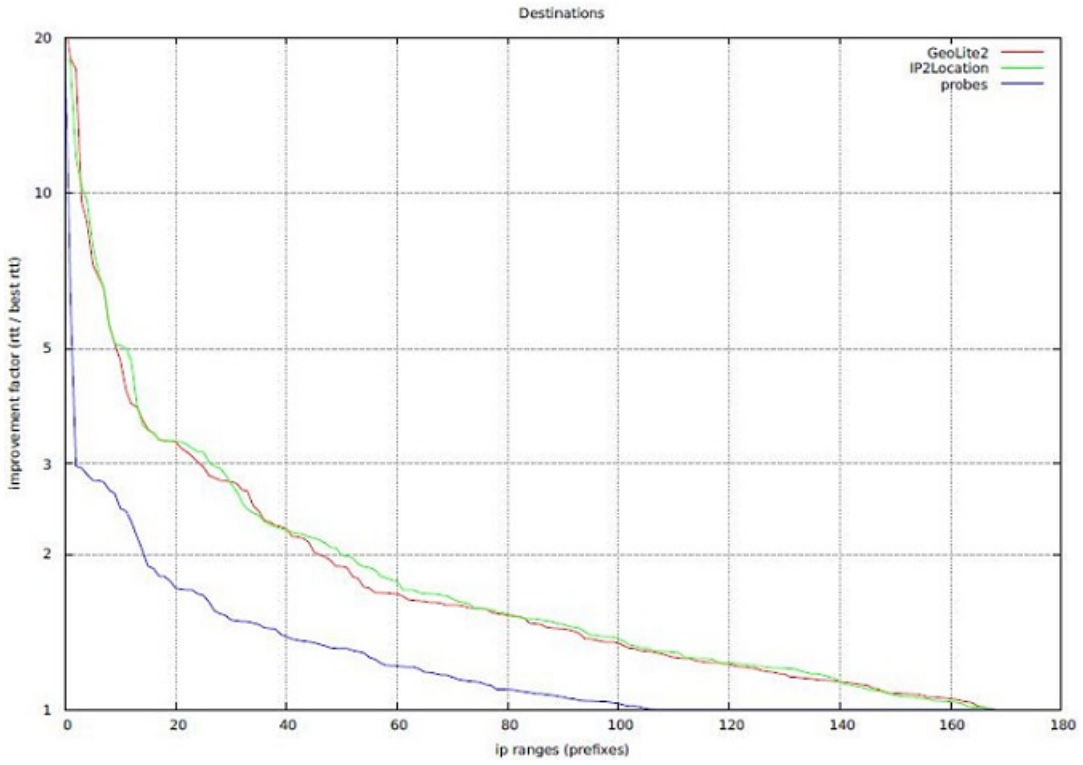

FIGURE 5.3: Influenced by data sets

As a result of the active measurement, there were in total 443 IP ranges were corrected. Even though, it is not many corrections, the method that was followed accomplished the goal.

Chapter 6

Summary

This chapter is devoted to summarizing what has been done in this project. This project contains two parts. The first part is the theoretical part where theory that is relevant to this project is covered with an extensive literature study about geolocation of IP addresses. In the literature study, an overview of geolocation has been given along with applications of use cases. A number of active and passive measurements that are in line with this research has been introduced. The literature study was concluded with presenting the international organizations working in IP addressing and the tools that are used for network measurements. This literature study aims to answer the sub research questions that are presented in the Chapter 1.

The second part of this project is the practical part where scripts in Python are written in order to find a method to improve the location accuracy by combining a passive measurement with active measurements. the answer the main research question. This practical part is explained below step-by-step, as follows;

1. Choose data sets to conduct passive measurement on.
2. Process the data sets to prepare them for the passive measurement.
3. Find the matching IP ranges across these data sets
4. Select the most specific (smallest) range
5. Calculate and output the distances between the locations estimated by landmark dataset to IP2Location dataset, landmark dataset to GeoIP2 dataset, and IP2Location dataset to GeoIP2 dataset for specific IP ranges into a list
6. Calculate the sum of all these distances and indicate them as the confidence value (the higher the value, the less confidence)

7. Iterate through the list starting from the top to choose the IP ranges that have the most inconsistent results (>50 km) across data sets. Output into a list.

8. Iterate through the list from the bottom to choose the probes within 50 km range that will be used to conduct the RTT measurements.(max = 50, min = 10). Output into the list

9. Find the number of probes that are located within 50 km range to the estimated locations taken from each data set. Output into the list.

10. Conduct RTT measurements with selected IP ranges along with their probes.

11. Analyze the results

The enumerated list above is a very short summary of the practical part of this research. With this, the theoretical and practical part of this project is completed.

Chapter 7

Evaluation

This chapter is devoted to evaluation of the results obtained in this project. The research findings will be critically evaluated on three aspects; validity, reliability and completeness.

The best way to assess geolocation information is to compare it against the ground truth data. However, there is no source that contains all the correct locations of IP addresses. Therefore, in the lack of ground truth data, the reliability of the results can be assured by the RTT measurements.

Bearing in mind the risk of probes not being spread evenly, ping measurements were created using minimum 10 probes and taken the average RTT collected. There were average of 37 probes used for each ping measurement.This reduces the possibility of this research being dependent on chance however doesn't fully eliminate it.

In this project, measurements were conducted with the intention of improving the estimate of the locations. To be able to calculate this improvement, the improvement factor was introduced in Chapter 5. This helped to to asses the validity for the results

Finally, being fully aware of the fact that the results of this project is not complete. This is because of the lack of reliable geolocation data.

The solutions to the issues discussed above are presented in the following chapter, 7.

Chapter 8

Discussion & Future Work

This chapter discusses the effectiveness of the solutions found and elaborates on the potential of placing the research in a broader context and critically reflects upon the approach and execution of the task with relation to the research. Moreover, suggestions are offered for improvements and possibilities are discussed for future research.

To start with, let's consider the number of corrections made in each database is given in the Table 5.3

The significance of this project is that the IP geolocation can be improved by combining passive and active measurements. However looking at the table, it can be seen that there were only 443 corrections made in total for three data sets. It is a very low amount considering the number of IP blocks, GeoIP2 and IP2Location have.

Effectiveness of the developed solutions can be increased by a bigger data set of landmarks. Center for Applied Internet Data Analysis (CAIDA) has an infrastructure of monitors, providing researchers a platform called Archipelago (Ark) tailored specifically for active measurements since 2007. They have been collecting data and building an extensive topology. To increase the amount of landmarks, data from the Ark project could have been used. However due to the inconsistency in the output and the variety of information this Ark database contained, using this database was beyond the scope of this project. Another data source could used was the data collected from OpenIpMap. However, because it is still in early development stage and the collected data is mainly crow-source-based, it would cause inaccurate results. Given a longer time frame, these two data sets could be added into the set of landmark.

To have better RTT measurements, instead of taking a certain of number probes, measurements could done across all the probes in the system. Moreover, traceroute measurements could have been done between probes in order to find the routers along the way. This also results in finding better RTTs.

Aside from these issues, the methodology applied in this project, has shown potential to be developed in future. One of the interesting ideas we came across was to investigate in the IPv6 deployment. Because of all the tunneling activity directed to United States, meanwhile the actual location being elsewhere, it could result in false numbers in IPv6 deployment by country.

Chapter 9

Conclusions

The ability to determine the geographical location of IP resources in the Internet is limited to a certain level of accuracy. Current techniques such as geodatabases and active-based measurements have limitations. While the problem of IP geolocation has been gaining the more research interest, there is still room for improvement.

The aim of this project was therefore to improve the geolocation of IP resources by combining passive methods and active measurement-based techniques.

The approach taken to achieve this goal was to make a systematic quantitative comparison of three datasets. Two of these data sets were the Lite versions of commercial geodatabases, namely GeoIP2 and IP2Location. Those were compared to the landmarks with known locations acquired from Atlas probes.

Through the derivation of methodologies, it was shown that how combining multiple data sets from different sources can help in correcting the falsely estimated locations.

Having examined the results collected, there were in total 443 records corrected across three data sets. The research findings were not significant however with the improvements proposed in the previous chapter 7, the results are likely to improve.

The data used to answer the research question are to be drawn was appropriate in terms of its relevance and was evaluated critically on their validity, reliability and completeness by assessing on its relevant aspects.

In conclusion, the geolocation of IP resources was improved by a systematic comparison across data sets (passive measurement), conducting ping measurements (active measurement-based geolocation technique) on the results and correcting records in each data set by calculating RTT values of the measurements.

Appendix A

City-Granularity - GeoIP2

Country	Accuracy Rate
Australia	15%
Austria	39%
Belgium	28%
Brazil	59%
Canada	58%
Denmark	46%
Finland	30%
France	28%
Germany	30%
India	44%
Indonesia	32%
Italy	26%
Netherlands	46%
New Zealand	49%
Norway	52%
Poland	38%
South Africa	45%
Spain	45%
Sweden	42%
Switzerland	24%
Turkey	66%
United Kingdom	36%
United States	53%

Appendix B

Postal-Granularity - GeoIP2

Country	Accuracy Rate
Australia	12%
Austria	18%
Belgium	22%
Brazil	10%
Canada	33%
Denmark	43%
Finland	10%
France	27%
Germany	11%
India	8%
Indonesia	N/A
Italy	9%
Netherlands	21%
New Zealand	13%
Norway	21%
Poland	12%
South Africa	18%
Spain	15%
Sweden	10%
Switzerland	18%
Turkey	4%
United Kingdom	14%
United States	40%

Appendix C

Get Scripts

C.1 GeoLite2 Get

```
cd data
rm -f GeoLite2-City-CSV.zip
ftp "http://geolite.maxmind.com/download/geoip/database/GeoLite2-City-CSV.zip"
unzip -o GeoLite2-City-CSV.zip
```

C.2 IP2Location Get

```
cd data
cat << EOT
# Register and download IP2LOCATION-LITE-DB5.IPV6.CSV.ZIP from
# http://www.ip2location.com/download?code=DB5LITEIPV6 and put
# it in the data directory.
#
# Then:
cd ~/data
unzip -o IP2LOCATION-LITE-DB5.IPV6.CSV.ZIP

# An overview of all downloadable databases is here:
# http://www.ip2location.com/file-download
EOT
```

C.3 Landmarks dataset Get

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
```

```
from atlas import *
import cPickle, time

# Get probes that are "connected", or "disconnected" only recently
# (i.e. less then one week ago).
#
last_week = time.time() - 604800
probes = [p for p in atlas.probe_archive() if p['latitude']  is not None
                                          and p['longitude'] is not None
                                          and ( p['address_v4']
                                             or p['address_v6'] )
                                          and ( p['status'] == 1
                                             or p['status'] == 2
                                            and p['status_since'] > last_week)]
print('%d probes' % len(probes))
cPickle.dump(probes, file('data/probes.cPickle', 'w'))
```

Appendix D

Process Scripts

D.1 Geo2Lite Process

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from utils import *

import glob, csv, cPickle

geolite2 = list()

base = sorted(glob.glob('data/GeoLite2-City-CSV_*'))[-1]

#with open(base + '/GeoLite2-City-Locations-en.csv', 'rb') as locations_file:
#       locations_csv = csv.reader(locations_file)

for fn in ('GeoLite2-City-Blocks-IPv4.csv', 'GeoLite2-City-Blocks-IPv6.csv'):
        print('Processing ' + fn + ' ...')
        with open(base + '/' + fn, 'rb') as blocks_file:
                blocks_csv = csv.reader(blocks_file)
                blocks_csv.next()
                for row in blocks_csv:
                        ip_from, ip_to = prefix2range(row[0])
                        try:
                                lat, lon = float(row[-2]), float(row[-1])
                        except ValueError:
                                continue
                        geolite2.append((ip_from, ip_to, lat, lon))

print('Writing cPickle file...')
with open('data/GeoLite2.cPickle', 'wb') as geolite2_file:
        cPickle.dump(geolite2, geolite2_file)
```

D.2 IP2Location Process

```
import glob, csv, cPickle, sys

ip2location = list()

base = 'data'

#with open(base + '/IP2Locationg-City-Locations-en.csv', 'rb') as locations_file:
#       locations_csv = csv.reader(locations_file)

for fn in ('IP2LOCATION-LITE-DB5.IPV6.CSV',):
        print('Processing ' + fn + ' ...')
        with open(base + '/' + fn, 'rb') as blocks_file:
                blocks_csv = csv.reader(blocks_file)
                blocks_csv.next()
                blocks_csv.next()
                for row in blocks_csv:
                        ip_from, ip_to, CC, country, state, city, lat,lon = row
                        lat, lon = float(lat), float(lon)
                        if (lat,lon,CC,country,state,city) ==
                           (0.0,0.0,'-','-','-','-'):
                                continue
                        ip2location.append((int(ip_from), int(ip_to), lat, lon))

print('Writing cPickle file...')
with open('data/IP2Location.cPickle', 'wb') as ip2location_file:
        cPickle.dump(ip2location, ip2location_file)
```

Appendix E

Shared Scripts

E.1 Atlas Script

```
import urllib2, urllib, json, os, sys
from pprint import pprint
from datetime import datetime, timedelta
from time import time

API_URL = 'https://atlas.ripe.net'

def api_path(*path, **args):
        return '/api/v1/%s?%s' % ( '/'.join(map(str, path)).replace('_', '-')
                                , urllib.urlencode(args))

def update_defaults(d, **defaults):
        d.update((k, v) for k, v in defaults.items() if k not in d)

class Atlas:
        def __init__(self, create_key = None, result_key = None):
                if not create_key:
                        with file('%s/.atlas/auth' % os.path.expanduser('~')) \
                        as f:
                                keys_l = f.read().strip().split()
                                create_key = keys_l[0]
                                if not result_key and len(keys_l) > 1:
                                        result_key = keys_l[1]
                self.create_key  = create_key

                if not result_key:
                        result_key = create_key
                self.result_key  = result_key

                redirect_handler = urllib2.HTTPRedirectHandler()
                cookie_handler   = urllib2.HTTPCookieProcessor()
                self.opener      = urllib2.build_opener( redirect_handler
                                                       , cookie_handler )
```

```
        def __getattr__(self, name):
                def get(*path, **args):
                        update_defaults(args, key = self.result_key, limit = 0)
                        url = api_path(name, *path, **args)
                        while url:
                                try:
                                        r = self.opener.open(API_URL + url)
                                except urllib2.HTTPError, e:
                                        print 'get "' + API_URL + url + '"'
                                        print e
                                        print e.read()
                                        r = self.opener.open(API_URL + url)

                                assert r.getcode() / 100 == 2
                                s = r.read()
"atlas.py" 196L, 5481C                                                          1
#!/usr/bin/env python

import urllib2, urllib, json, os, sys
from pprint import pprint
from datetime import datetime, timedelta
from time import time

API_URL = 'https://atlas.ripe.net'

def api_path(*path, **args):
        return '/api/v1/%s?%s' % ( '/'.join(map(str, path)).replace('_', '-')
                                 , urllib.urlencode(args))

def update_defaults(d, **defaults):
        d.update((k, v) for k, v in defaults.items() if k not in d)

class Atlas:
        def __init__(self, create_key = None, result_key = None):
                if not create_key:
                        with file('%s/.atlas/auth' % os.path.expanduser('~')) \
                        as f:
                                keys_l = f.read().strip().split()
                                create_key = keys_l[0]
                                if not result_key and len(keys_l) > 1:
                                        result_key = keys_l[1]
                self.create_key  = create_key

                if not result_key:
                        result_key = create_key
                self.result_key  = result_key

                redirect_handler = urllib2.HTTPRedirectHandler()
                cookie_handler   = urllib2.HTTPCookieProcessor()
                self.opener      = urllib2.build_opener( redirect_handler
                                                       , cookie_handler )

        def __getattr__(self, name):
                def get(*path, **args):
```

```
                        update_defaults(args, key = self.result_key, limit = 0)
                        url = api_path(name, *path, **args)
                        while url:
                                try:
                                        r = self.opener.open(API_URL + url)
                                except urllib2.HTTPError, e:
                                        print 'get "' + API_URL + url + '"'
                                        print e
                                        print e.read()
                                        r = self.opener.open(API_URL + url)

                                assert r.getcode() / 100 == 2
                                s = r.read()
                                try:
                                        j = json.loads(s)
                                except ValueError:
                                        j = eval(s)
                                if 'objects' not in j or 'meta'    not in j:
                                        yield j
                                        return
                                for obj in j['objects']:
                                        yield obj
                                url = j['meta'].get('next', None)
                return get

        def msm(self, *path, **args):
                return self.measurement(*path, **args)

        def result(self, msm_id):
                return self.measurement(msm_id, 'result')

        def create(self, definitions, *probes):

                probes = list(probes)
                req = { 'definitions': definitions if type(definitions) is list
                                                   else [definitions]
                      , 'probes'     : list()
                      }

                for key in ('stop_time', 'start_time'):
                        if not probes: break
                        if   type(probes[-1]) in (int, float):
                                req[key] = int(probes[-1])
                        elif type(probes[-1]) is datetime:
                                req[key] = int(probes[-1].strftime("%s"))
                        else:
                                break
                        probes.pop()

                for probe in probes:
                        if type(probe) is list:
                                req['probes'].append(
                                        { 'requested': len(probe)
                                        , 'type'     : 'probes'
                                        , 'value'    : ','.join(map(str, probe))
```

```
                                        })
                        elif type(probe) is dict:
                                req['probes'].append(probe)
                        else:
                                raise Exception( "Unknown probe type: %s"
                                        % repr(probe))

                url = API_URL + api_path('measurement/', key = self.create_key)
                try:
                        r = self.opener.open( urllib2.Request( url, json.dumps(req)
                                        , {'Content-Type': 'application/json'}))
                except urllib2.HTTPError, e:
                        print 'post "' + url + '"'
                        print e
                        print e.read()
                        r = self.opener.open( urllib2.Request( url, json.dumps(req)
                                        , {'Content-Type': 'application/json'}))

                assert r.getcode() / 100 == 2
                return json.loads(r.read())

def msm_defaults(kwargs, **defaults):
        update_defaults(defaults, description = kwargs.get('description', '')
                                , is_oneoff = 'interval' not in kwargs
                                , af = 4)
        update_defaults(kwargs  , **defaults)
        return dict([(k, v) for k, v in kwargs.items() if v is not None])

def dns(query_argument, query_type = 'A', target = None, **kwargs):
        return msm_defaults( kwargs, type = 'dns', query_class = 'IN'
                           , query_argument = query_argument
                           , query_type = query_type, target = target
                           , use_probe_resolver = target is None
                           , recursion_desired  = target is None
                           )

def dns6(query_argument, qtype = 'TXT', target = None, **kwargs):
        return dns(query_argument, qtype, target, af = 6, **kwargs)

def msm_constructor(msm_type, **params):
        def constructor(target, **kwargs):
                return msm_defaults( kwargs, type = msm_type
                                   , target = target, **params)
        return constructor

ping        = msm_constructor('ping')
ping6       = msm_constructor('ping', af = 6)
traceroute  = msm_constructor('traceroute', protocol = 'ICMP')
traceroute6 = msm_constructor('traceroute', protocol = 'ICMP', af = 6)
sslcert     = msm_constructor('sslcert')
sslcert6    = msm_constructor('sslcert', af = 6)

def probes(amount, p_type, value):
        return { 'requested': amount, 'type': p_type, 'value': value }
```

```
def probes_WW(amount):
        return probes(amount, 'area', 'WW')

try:
        atlas = Atlas()
except:
        atlas = None

### Examples
#
### Import atlas
#
# from atlas import *
#
#
### Get a list of all probes
#
# probes = atlas.probe(prefix_v6 = '::/0', limit = 0)
#
#
### Filter probes that are up
#
# probes = filter(lamba p: p['status'] == 1, probes)
#
#
### Create a one off dns6 measurement
#
# definition = dns6('ripe67.nlnetlabs.nl', 'AAAA', '2001:7b8:40:1:d0e1::1')
# r = atlas.create(definition, probes_ww(500))
#
#
### Create periodic dns6 measurement (each 20 minutes, ends after 100 minutes)
#
# definition = dns6('ripe67.nlnetlabs.nl', 'AAAA', '2001:7b8:40:1:d0e1::1'
#                   , interval = 20 * 60)
# r = atlas.create(definition, probes_ww(500), time() + 20 * 60 * 5.5)
#
#
### Check on status of measurement
#
# atlas.measurement(r['measurements'][0])
#
#
### Check result of measurement
#
# atlas.result(r['measurements'][0])
#
#
```

E.2 Utils Script

```
from socket import inet_pton, inet_ntop, AF_INET, AF_INET6
```

```
def ip2int(ip):
        b = inet_pton(AF_INET if '.' in ip else AF_INET6, ip)
        if not b:
                return 0
        i = int(b.encode('hex'), 16)
        if '.' in ip:
                i |= 0xffff00000000
        return i

def prefix2range(prefix):
        if prefix is None:
                return None
        ip, mask = prefix.split('/')
        ip_from = ip2int(ip)
        mask = int(mask) + (96 if '.' in ip else 0)
        return (ip_from, ip_from + 2 ** (128 - mask) - 1)

import sys

def print_range(prange):
        def packed(i):
                h = hex(long(i))[2:-1]
                return ('0'+h if len(h) % 2 else h).decode('hex')

        if prange[0] < 0xffff00000000 or prange[0] > 0xffffffffffff:
                return ( inet_ntop(AF_INET6, packed(prange[0]))
                       , inet_ntop(AF_INET6, packed(prange[1])) )
        return ( inet_ntop(AF_INET, packed(prange[0] & 0xffffffff))
               , inet_ntop(AF_INET, packed(prange[1] & 0xffffffff)) )

import cPickle
from bisect import bisect_left

class rangeMap:
        def __init__(self, filename):
                self.filename = filename
                with file(filename, 'r') as f:
                        self.data = cPickle.load(f)
                self.keys = [x[0] for x in self.data]

        def lookup(self, ip):
                i = ip2int(ip)
                l = bisect_left(self.keys, i)
                if l <= 0 or i < self.data[l-1][0] or i > self.data[l-1][1]:
                        return None
                return self.data[l-1]
```

E.3 Distance Calculation Script (utils)

```
import math
```

```
def distance_on_unit_sphere(lat1, long1, lat2, long2):
        # Convert latitude and longitude to
        # spherical coordinates in radians.
        degrees_to_radians = math.pi/180.0

        # phi = 90 - latitude
        phi1 = (90.0 - lat1)*degrees_to_radians
        phi2 = (90.0 - lat2)*degrees_to_radians

        # theta = longitude
        theta1 = long1*degrees_to_radians
        theta2 = long2*degrees_to_radians

        # Compute spherical distance from spherical coordinates.

        # For two locations in spherical coordinates
        # (1, theta, phi) and (1, theta, phi)
        # cosine( arc length ) =
        #    sin phi sin phi' cos(theta-theta') + cos phi cos phi'
        # distance = rho * arc length

        cos = (math.sin(phi1)*math.sin(phi2)*math.cos(theta1 - theta2) +
               math.cos(phi1)*math.cos(phi2))
        arc = math.acos( cos )

        # Remember to multiply arc by the radius of the earth
        # in your favorite set of units to get length.
        #
        # Earth radius is the distance from the Earth's center to its surface,
        # about 6,371 kilometers (see: https://en.wikipedia.org/wiki/Earth_radius)
        return arc * 6371
```

Appendix F

100-Script

F.1 Landmark Selection

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from utils import *

import cPickle
from itertools import combinations
from pprint import pprint as pp

with open('data/probes.cPickle') as probes_file:
        probes = cPickle.load(probes_file)

exclude = rangeMap([
        prefix2range('2001:470::/32'),  # Hurricane electric IPv6 tunnels
        prefix2range('2002::/16')       # 6to4 (RFC3056)
])

ranges = dict()
for p in probes:
        for family in ('v4', 'v6'):
                prange = p['range_' + family]
                if prange[0] is None:
                        continue
                if exclude.lookup(p['address_' + family]):
                        p['range_' + family] = (None, prange[1] + 'E')
                        continue
                if prange[0] not in ranges:
                        ranges[prange[0]] = list()
                ranges[prange[0]].append((p['latitude'], p['longitude']))


def mean(l):
        if l:
                return sum(l) / len(l)
```

```
        return 0.0

ranges = sorted([ ( mean([distance_on_unit_sphere(loc1[0],loc1[1],loc2[0],loc2[1])
                          for loc1, loc2 in combinations(locs, 2)]
                         ), len(locs), prange )
                  for prange, locs in ranges.iteritems() if len(locs) > 1
                ] , reverse=True)

for avg_d, n, prange in ranges:
        if avg_d > 0:
                print(avg_d, n, print_range(prange))


exclude.data = sorted(exclude.data + [prange for avg_d, n, prange
                                            in ranges if avg_d > 500])
with file('data/exclude.cPickle', 'w') as f:
        cPickle.dump(exclude.data, f)

for p in probes:
        for family in ('v4', 'v6'):
                prange = p['range_' + family]
                if prange[0] is None:
                        continue
                if exclude.lookup(p['address_' + family]):
                        p['range_' + family] = (None, prange[1] + 'E')

with file('data/washed_probes.cPickle', 'w') as f:
        cPickle.dump(probes, f)
```

F.2 Smallest Prefix

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from utils import *

import cPickle
from pprint import pprint as pp

probes  = cPickle.load(open('data/probes.cPickle'))
ip2loc  = rangeMap('data/IP2Location.cPickle')
geolite = rangeMap('data/GeoLite2.cPickle')

def smallest_range(p, rmap, family, source):
        if p['address_' + family] is None:
                return
        l = rmap.lookup(p['address_' + family])
        if l is None:
                return
        prange = p['range_' + family]
        if prange[0] is None:
                prange = ((l[0], l[1]), prange[1] + source)
```

```
        else:
                if l[0] > prange[0][0]:
                        prange = ((l[0], prange[0][1]), prange[1] + '>' + source)
                if l[1] < prange[0][1]:
                        prange = ((prange[0][0], l[1]), prange[1] + '<' + source)
        p['range_' + family] = prange
        p['dists_' + family].append((source,
                distance_on_unit_sphere(p['latitude'], p['longitude'], l[2], l[3]),
                l[2], l[3]))

for p in probes:
        for family in ('v4', 'v6'):
                p['dists_' + family] = list()
                p['range_' + family] = (prefix2range(p['prefix_' + family]), 'p')
                smallest_range(p, ip2loc, family, 'i')
                smallest_range(p, geolite, family, 'g')
                if len(p['dists_' + family]) == 2:
                        p['dists_' + family].append(('p',
                                distance_on_unit_sphere( p['dists_' + family][0][2]
                                                       , p['dists_' + family][0][3]
                                                       , p['dists_' + family][1][2]
                                                       , p['dists_' + family][1][3]
                                ), p['latitude'], p['longitude']))

cPickle.dump(probes, open('data/probes.cPickle', 'w'))
```

Appendix G

200-Scripts

G.1 Measure Distances

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from utils import *

import cPickle
from itertools import combinations
from pprint import pprint as pp

with open('data/washed_probes.cPickle') as probes_file:
        probes = cPickle.load(probes_file)

n_probes   = {'v4': 0, 'v6': 0}
in_ip2loc  = {'v4': 0, 'v6': 0}
in_geolite = {'v4': 0, 'v6': 0}
in_probes  = {'v4': 0, 'v6': 0}
missing    = {'v4': 0, 'v6': 0}

dists      = {'v4': {'i': [], 'g': [], 'p': []}, 'v6': {'i': [], 'g': [], 'p': []}}

dist_lines = {'v4': [], 'v6': []}

i = -1
for p in probes:
        i += 1
        for family in ('v4', 'v6'):
                prange = p['range_' + family]
                if prange[0] is None:
                        continue
                n_probes[family] += 1
                dists_line = [0, None, None, None, i]
                if p['dists_' + family]:
                        for d in p['dists_' + family]:
                                dists[family][d[0]].append((d[1], i))
```

```
                                dists_line[0] += d[1]
                                if d[0] == 'i':
                                        in_ip2loc[family] += 1
                                        dists_line[1] = d[1]
                                elif d[0] == 'g':
                                        in_geolite[family] += 1
                                        dists_line[2] = d[1]
                                elif d[0] == 'p':
                                        in_probes[family] += 1
                                        dists_line[3] = d[1]
                else:
                        missing[family] += 1

                dist_lines[family].append(tuple(dists_line))

for family in ('v4', 'v6'):
        print('total probes with %s address: %d' % (family, n_probes[family]))
        print('\taddress missing from IP2Location db: %d' %
                (n_probes[family] - in_ip2loc[family]))
        print('\taddress missing from GeoLite2    db: %d' %
                (n_probes[family] - in_geolite[family]))
        print('\taddress missing from either one  db: %d' %
                (n_probes[family] - in_probes[family]))
        print('\taddress missing from both        db: %d' % (missing[family]))
        print

def print_lists(l1, l2, l3, f):
        l1.sort(reverse=True)
        l2.sort(reverse=True)
        l3.sort(reverse=True)
        if len(l1) > len(l2):
                l2.extend([(0, None)] * (len(l1) - len(l2)))
        elif len(l2) > len(l1):
                l1.extend([(0, None)] * (len(l2) - len(l1)))

        if len(l1) > len(l3):
                l3.extend([(0, None)] * (len(l1) - len(l3)))
        elif len(l3) > len(l1):
                l1.extend([(0, None)] * (len(l3) - len(l1)))
                l2.extend([(0, None)] * (len(l3) - len(l1)))

        i = 0.0
        for (d1, i1), (d2, i2), (d3, i3) in zip(l1, l2, l3):
                f.write('%f\t%f\t%f\t%f\n' % ((i / len(l1) * 100), d1, d2, d3))
                i += 1

with file('data/v4.csv', 'w') as f:
        print_lists(dists['v4']['g'], dists['v4']['i'], dists['v6']['p'], f)

with file('data/v6.csv', 'w') as f:
        print_lists(dists['v6']['g'], dists['v6']['i'], dists['v6']['p'], f)

for fam in ('v4', 'v6'):
        dist_lines[fam].sort(reverse=True)
        with file('data/dists_'+fam+'.csv', 'w') as f:
```

```
                for dists_line in dist_lines[fam]:
                        f.write( '\t'.join(['-' if d is None
                        else str(d) for d in dists_line])
                                + '\t'
                                + str(probes[dists_line[4]]['id'])
                                + '\t'
                                + str(print_range(probes[dists_line[4]]
                                ['range_' + fam][0]))
                                + '\n'
                                )

        with file('data/dists_'+fam+'.cPickle', 'w') as f:
                cPickle.dump(dist_lines[fam], f)
```

G.2 Find Near Probes

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from utils import *

import cPickle
from itertools import combinations
from pprint import pprint as pp

with open('data/washed_probes.cPickle') as probes_file:
        probes = cPickle.load(probes_file)

def find_closest(lat, lon, threshold, amount):
        ps = list()
        for dist_line in rdists:
                if dist_line[0] > threshold:
                        break
                p = probes[dist_line[4]]
                d = distance_on_unit_sphere(lat, lon, p['latitude'], p['longitude'])
                if d < threshold:
                        ps.append((p['id'], d, dist_line[0], dist_line[4]))
                if len(ps) >= amount:
                        break
        return ps

for fam in ('v4', 'v6'):
        with open('data/dists_%s.cPickle' % fam) as dists_file:
                dists = list(cPickle.load(dists_file))
                rdists = list(reversed(dists))

        to_schedule = list()
        j = -1
        for dist_line in dists:
                j += 1
                sys.stderr.write('%d (%f)  ' % (j, dist_line[0]))
                if dist_line[0] < 50:
```

```
                        break
                p = probes[dist_line[4]]
                near_probes = list()
                for source, distance, lat, lon in p['dists_' + fam]:
                        do_continue = False
                        for s in near_probes:
                                if distance_on_unit_sphere(lat, lon, s[1], s[2]) < 50:
                                        near_probes.append((source, lat, lon, s[0]))
                                        do_continue = True
                                        break
                        if do_continue:
                                continue
                        closest = find_closest(lat, lon, 50, 50)
                        if len(closest) > 10:
                                near_probes.append((source, lat, lon, closest))

                if len([1 for s in near_probes if type(s[3]) is list and s[3]]) > 1:
                        to_schedule.append((p['id'],
                        p['address_'+fam], dist_line[4], dist_line[0], near_probes))

        with file('data/schedule_%s.cPickle' % fam, 'w') as f:
                cPickle.dump(to_schedule, f)

        with file('data/schedule_%s.csv' % fam, 'w') as f:
                for prb_id, address, i, d, near_probes in to_schedule:
                        f.write('%6d\t%s\t%s\t%f\n' %
                        (prb_id, address, ', '.join(['%s: %s' %
                        (s, cl if type(cl) is str else str(len(cl)))
                        for s, lat, lon, cl in near_probes]), d))
```

Appendix H

300-Scripts

H.1 Schedule Measurements

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from atlas import *

import cPickle
from itertools import combinations
from pprint import pprint as pp

with open('data/washed_probes.cPickle') as probes_file:
        probes = cPickle.load(probes_file)

for fam in ('v4', 'v6'):
        with file('data/schedule_%s.cPickle' % fam) as f:
                to_schedule = cPickle.load(f)

        j = -1
        for prb_id, address, index, distance, near_probes in to_schedule:
                j += 1
                if j >= 450:
                        break
                new_near_probes = []
                for sched_line in near_probes:
                        if len(sched_line) > 4:
                                # already scheduled
                                new_near_probes.append(sched_line)
                                #pp(sched_line)
                                r = sched_line[4]
                                if 'measurements' in r and 'result' not in r:
                                        msm_id = r['measurements'][0]
                                        m = atlas.msm(msm_id).next()
                                        if m['status']['id'] == 4: #Stopped
                                                r['result'] =
                                                atlas.result(msm_id).next()
```

```
                                        else:
                                                print prb_id, address,
                                                sched_line[0], 'running'
                                if 'result' in r and 'avg_rtt' not in r:
                                        rtts = [t['rtt'] for x in r['result']
                                                        for t in x['result']
                                                        if 'rtt' in t]
                                        if len(rtts) == 0:
                                                avg_rtt = '-'
                                        else:
                                                avg_rtt = sum(rtts) / len(rtts)
                                        r['avg_rtt'] = avg_rtt
                                if 'avg_rtt' in r:
                                        print prb_id, address, sched_line[0],
                                        r['avg_rtt']
                                continue
                        source, lat, lon, probes = sched_line
                        if type(probes) is list:
                                if fam == 'v4':
                                        ping_def = ping(address, description =
                                        'source: %s, near: %f, %f' %
                                        (source, lat, lon))
                                else:
                                        ping_def = ping6(address, description =
                                        'source: %s, near: %f, %f' %
                                        (source, lat, lon))
                                r = atlas.create(ping_def, [p[0] for p in probes])
                                print r
                                sched_line = (source, lat, lon, probes, r)
                        new_near_probes.append(sched_line)
                #print j, len(near_probes), len(new_near_probes),
                #near_probes == new_near_probes
                to_schedule[j] = (prb_id, address, index, distance, new_near_probes)



        with file('data/schedule_%s.cPickle' % fam, 'w') as f:
                cPickle.dump(to_schedule, f)
```

H.2 Count Corrections

```
import sys, os.path
sys.path.append(os.path.dirname(sys.argv[0]) + '/lib')
from atlas import *

import cPickle
from itertools import combinations
from pprint import pprint as pp

with open('data/washed_probes.cPickle') as probes_file:
        probes = cPickle.load(probes_file)
```

```
distmap = { 'pi': 'i'
          , 'ip': 'i'
          , 'pg': 'g'
          , 'gp': 'g'
          , 'ig': 'p'
          , 'gi': 'p'
          }

improvements = {'p': [], 'i': [], 'g': []}
impdist = {'p': [], 'i': [], 'g': []}
impdistsrc = {'p': [], 'i': [], 'g': []}
for fam in ('v4', 'v6'):

        with file('data/schedule_%s.cPickle' % fam) as f:
                to_schedule = cPickle.load(f)

        j = -1
        for prb_id, address, index, distance, near_probes in to_schedule:
                j += 1

                p = probes[index]
                range_from, range_to = p['range_' + fam][0]

                r = dict()
                for sched_line in near_probes:
                        if len(sched_line) <= 4:
                                continue
                        if 'avg_rtt' not in sched_line[4]:
                                continue
                        avg_rtt = sched_line[4]['avg_rtt']
                        r[sched_line[0]] = avg_rtt

                for sched_line in near_probes:
                        if type(sched_line[3]) is str:
                                r[sched_line[0]] = r[sched_line[3]]

                r = dict([(key, val) for key, val in r.iteritems() if val != '-'])
                if not r:
                        continue

                rating = sorted(r.iteritems(), key = lambda x: x[1])
                best_src, best_val = rating[0]
                lat, lon = [x[2:] for x in p['dists_' + fam]
                if x[0] == best_src][0]
                dists = dict([x[:2] for x in p['dists_' + fam]])
                for rest_src, rest_val in rating[1:]:
                        factor = rest_val / best_val
                        if factor == 1:
                                continue
                        impdist[rest_src].append(factor)
                        impdistsrc[best_src].append(factor)
                        improvements[rest_src].append(
                        ( range_from, range_to
                        , lat, lon
```

```
                        , best_src
                        , factor
                        , dists[distmap[best_src+rest_src]]
                        ))

for src, name in (('p', 'probes'), ('i', 'ip2location'), ('g', 'geolite2')):
        improvements[src].sort()
        impdist[src].sort(reverse=True)
        impdistsrc[src].sort(reverse=True)
        with file('data/improved_%s.csv' % name, 'w') as f:
                for line in improvements[src]:
                        f.write('\t'.join(map(str,line)) + '\n')
        print '%d improved %s' % (len(improvements[src]), name)

def lzip(*lists):
        def lsz(l):
                cnt = sz
                for i in l:
                        yield str(i)
                        cnt -= 1
                while cnt:
                        yield '1'

        sz = max(map(len, lists))
        iters = map(lsz, lists)
        while True:
                yield map(next, iters)

with file('data/improvements.csv', 'w') as f:
        for line in lzip(impdist['i'], impdist['g'], impdist['p']):
                f.write('\t'.join(line) + '\n')

with file('data/improvements_source.csv', 'w') as f:
        for line in lzip(impdistsrc['i'], impdistsrc['g'], impdistsrc['p']):
                f.write('\t'.join(line) + '\n')
```

Appendix I

Geodatabases

I.1 GeoIP2-City IPv4

Table I.1: GeoIP2 Lite - City IPv4

network	geoname_id	country_id	latitude	longitude
1.0.0.0/24	2077456	2077456	-27.0000	133.0000
1.0.1.0/24	1814991	1814991	35.0000	105.0000
1.0.2.0/23	1814991	1814991	35.0000	105.0000
1.0.4.0/22	2077456	2077456	-27.0000	133.0000
1.0.8.0/21	1809858	1814991	23.1167	113.2500
1.0.16.0/20	1850147	1861060	35.6850	139.7514
1.0.32.0/19	1809858	1814991	23.1167	113.2500
1.0.64.0/18	1862415	1861060	34.3963	132.4594
1.0.128.0/22	1605651	1605651	13.7500	100.4667
1.0.132.0/24	1158432	1605651	13.4167	99.9500
1.0.133.0/24	1605651	1605651	13.7500	100.4667
1.0.134.0/24	1609350	1605651	13.7500	100.5167
1.0.135.0/24	1605651	1605651	13.7500	100.4667
1.0.136.0/21	1605651	1605651	13.7500	100.4667
1.0.144.0/20	1605651	1605651	13.7500	100.4667
1.0.160.0/21	1605651	1605651	13.7500	100.4667
1.0.168.0/24	1605651	1605651	13.7500	100.4667
1.0.169.0/24	1609350	1605651	13.7500	100.5167
1.0.170.0/23	1609350	1605651	13.7500	100.5167
1.0.172.0/22	1605651	1605651	13.7500	100.4667
1.0.176.0/22	1605651	1605651	13.7500	100.4667
1.0.180.0/23	1605651	1605651	13.7500	100.4667
1.0.182.0/24	1150452	1605651	8.8333	98.3667
1.0.183.0/24	1605651	1605651	13.7500	100.4667

To download the full database: http://geolite.maxmind.com/download/geoip/database/GeoLite2-City-CSV.zip

I.2 GeoIP2Lite-City IPv6

Table I.2: GeoIP2 Lite - City IPv6

network	geoname_id	country_id	latitude	longitude
2001:200::/49	2110681		36.0833	140.1167
2001:200:120::/49	1850147		35.6600	139.8067
2001:200:167::/49	1850147		35.6850	139.7514
2001:208:5::/49	1880252		1.2931	103.8558
2001:218::/32	1861060	1861060	35.68536	139.75309
2001:220::/32	1835841	1835841	36.5	127.75
2001:230::/32	1835841	1835841	36.5	127.75
2001:238::/32	1668284	1668284	24	121
2001:240::/40	1861060	1861060	35.68536	139.75309
2001:240:100::/43	1861060	1861060	35.68536	139.75309
2001:240:120::/44	1861060	1861060	35.68536	139.75309
2001:240:130::/45	1861060	1861060	35.68536	139.75309
2001:240:138::/48	1861060	1861060	35.68536	139.75309
2001:240:139::/49	1848373	1861060	34.9667	136.6167
2001:240:139:8000::/49	1861060	1861060	35.68536	139.75309
2001:240:13a::/47	1861060	1861060	35.68536	139.75309
2001:240:13c::/46	1861060	1861060	35.68536	139.75309
2001:240:140::/42	1861060	1861060	35.68536	139.75309
2001:240:180::/41	1861060	1861060	35.68536	139.75309
2001:240:200::/39	1861060	1861060	35.68536	139.75309
2001:240:400::/39	1861060	1861060	35.68536	139.75309
2001:240:600::/41	1861060	1861060	35.68536	139.75309
2001:240:680::/43	1861060	1861060	35.68536	139.75309
2001:240:6a0::/44	1861060	1861060	35.68536	139.75309

To download the full database: http://geolite.maxmind.com/download/geoip/database/GeoLite2-City-CSV.zip

I.3 IP2Location Dataset

Table I.3: IP2Location

IP	Ranges	Co	Country	Province	City	Latitude	Longitude
2.81471E+14	2.81471E+14	AU	Australia	Queensland	Brisbane	-27.46794	153.02809
2.81471E+14	2.81471E+14	CN	China	Fujian	Fuzhou	26.06139	119.30611
2.81471E+14	2.81471E+14	AU	Australia	Victoria	Melbourne	-37.814	144.96332
2.81471E+14	2.81471E+14	CN	China	Guangdong	Guangzhou	23.11667	113.25
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	CN	China	Guangdong	Guangzhou	23.11667	113.25
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528
2.81471E+14	2.81471E+14	JP	Japan	Tokyo	Tokyo	35.689506	139.6917
2.81471E+14	2.81471E+14	JP	Japan	Hiroshima	Hiroshima	34.38528	132.45528

To download the full database : https://lite.ip2location.com/database-ip-country-region-city-latitude-longitude

Appendix J

Distance Table

J.1 Calculated Distances IPv4

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/18bbukIQjX69IqAQR86vjuCjcPtdWIbHxpVbpg9m5e88/edit?usp=sharing`

J.2 Calculated Distances IPv6

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/1aq_TU9OqWX9xLT6WM2n9JmhBE-DqNPLg92Ki7YJ4OKo/edit?usp=sharing`

Appendix K

Distance Graph

K.1 Distance Graph v4

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/1adUILmPSuewo4OQ6WzoUNuaFa6FYWT56a0iYMZenl5c/edit?usp=sharing`

K.2 Distance Graph v6

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/14_NEsCqGYKWyuqvnNF2Oc4RCKISk1o-KtnVbzYJdW5s/edit?usp=sharing`

Appendix L

Schedule Lists

L.1 Schedule Measurements v4

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

https://docs.google.com/spreadsheets/d/1sWr7tQek2zVuHhnn6QDUb45JQMQAbMsuJNrHeGqzZnw/edit?usp=sharing

L.2 Schedule Measurements v6

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

https://docs.google.com/spreadsheets/d/1kkVc1WqE_ejGlT90WjaLc3mBt3Z1EgkR5j3PLhq-E5U/edit?usp=sharing

Appendix M

Improved Data sets

M.1 Improved Ranges - GeoIP2

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/164BJYVw0xMey3Fih13NzY2Z2wpTHTs4AmtGzBQr03vo/edit?usp=sharing`

M.2 Improved Ranges - IP2Location

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

`https://docs.google.com/spreadsheets/d/1-YUJnpSByRIIuv7trpy9tPjMf9o7JZcLuWPZmlKPaf8/edit?usp=sharing`

M.3 Improved Ranges - Landmarks Dataset

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

https://docs.google.com/spreadsheets/d/1KFx6w1oR0qb0iJbqc4YGnmGaTAcKPwfJfas8NzT-6FA/edit?usp=sharing

Appendix N

Improvement Graphs

N.1 Improvements Graph - Destination

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

https://docs.google.com/spreadsheets/d/14xtOVCZMXrPk9U1uYQgZtZdA77Y2MfXZ2ajjCExH0oo/edit?usp=sharing

N.2 Improvements Graph - Source

Due to the table being too big to fit properly in this paper, the list was uploaded to Google Drive in spread sheet format. Follow the link below to see the list. It is a publicly shared document, anyone (no-sign in required) can see.

https://docs.google.com/spreadsheets/d/1gBZUEAJe7ef6IqRWmrStoPi8izX0To7A2bAcZSYa_dI/edit?usp=sharing

Appendix O

Public Ping Measurements

O.1 Instructions

1. Go to atlas.ripe.net
2. Create an account (Free of charge)
3. Click on "My Atlas" on the left side bar and proceed with clicking "Measurements".
4. Choose the "Public" tab.
5. Copy an IP address from one of the lists found in Appendix L.1 or L.2.
6. Paste it into the search bar and filter it to only see ping measurements.
7. Click on the ID of the measurements to see more details.
8. From there on, a lot of information can be seen such as which probes are used for the measurement, specific RTT values, probes seen on a map and so on.

O.2 Example

Here is the link to the results of the RTT measurements for the IP address "63.218.228.166". It is a public measurement and can be seen in a better format with JSON.

`https://atlas.ripe.net/api/v1/measurement/2055884/result/?start=1435017600&stop=1435103999&format=json`

Probe	ASN (v4)	ASN (v6)	Time	RTT
19627	55430		2015-06-23 13:16	4.107
14171	56300		2015-06-23 13:16	4.234
19961	4773		2015-06-23 13:16	4.549
18385	55430		2015-06-23 13:16	4.652
19930	56300		2015-06-23 13:16	4.829
14776	56300		2015-06-23 13:16	4.899
14350	4773	4773	2015-06-23 13:16	5.092
13807	132047		2015-06-23 13:16	5.307
86	55430		2015-06-23 13:16	5.677
14772	56300		2015-06-23 13:16	5.787
4430	37989		2015-06-23 13:16	5.875
10676	36351	36351	2015-06-23 13:16	35.433
14672	10091	10091	2015-06-23 13:16	42.265
22724	10091		2015-06-23 13:16	45.574
17912	36351	6939	2015-06-23 13:16	46.161
20621	10091		2015-06-23 13:16	48.556

FIGURE O.1: RTT Measurement Example

References

Aben, E. (2015). *Ripe 69 - london.* `https://ripe69.ripe.net/presentations/83-2014-11.emileaben.ripe69.openipmap.pdf`. (Accessed 18th of April, 2015)

Bendale, J., & Kumar, J. R. (2014). Review of different ip geolocation methods and concepts. *International Journal of Computer Science & Information Technologies*, *5*(1), 436 – 440.

Breitbart, Y., Garofalakis, M., Jai, B., Martin, C., Rastogi, R., & Silberschatz, A. (2004). Topology discovery in heterogeneous ip networks: the netinventory system. *IEEE/ACM Transactions on Networking (TON)*, *12*(3), 401–414.

Carpenter, B., & Moore, K. (2001). Rfc 3056. *Connection of IPv6 Domains via IPv4 Clouds*.

Carpenter, B., Roberts, M., & Baker, F. (2000). Memorandum of understanding concerning the technical work of the internet assigned numbers authority.

Davis, C., Dickinson, I., Goodwin, T., & Vixie, P. (1996). Rfc 1876: A means for expressing location information in the domain name system. *IETF*.

Donnet, B., & Friedman, T. (2007). Internet topology discovery: a survey. *Communications Surveys & Tutorials, IEEE*, *9*(4), 56–69.

Gueye, B., Ziviani, A., Crovella, M., & Fdida, S. (2006). Constraint-based geolocation of internet hosts. *Networking, IEEE/ACM Transactions on*, *14*(6), 1219–1232.

Hinden, R., & Deering, S. (1998). Rfc 2373. *IP version*, *6*, 6.

IANA. (2015). *Memorandum of understanding concerning the technical work of the internet assigned numbers authority.* `http://www.iana.org/about`. (Accessed 5th of May, 2015)

ICANN. (2008). *Board of directors' code of conduct, internet corporation for assigned names and numbers.* `https://www.icann.org/en/system/files/files/bod-code-of-conduct-01oct08-en.pdf`. (Accessed 7th of May, 2015)

Katz-Bassett, E., John, J. P., Krishnamurthy, A., Wetherall, D., Anderson, T., & Chawathe, Y. (2006). Towards ip geolocation using delay and topology measurements. In *Proceedings of the 6th acm sigcomm conference on internet measurement* (pp. 71–84). Rio de Janeiro, Brazil.

Koch, R., Golling, M., & Rodosek, G. D. (2013). Advanced geolocation of ip addresses. In *International conference on communication and network security (iccns)* (p. 505-514).

Koch, R., Golling, M., Stiemert, L., & Rodosek, G. D. (2015). Using geolocation for the strategic preincident preparation of an it forensics analysis. *IEEE SYSTEMS JOURNAL*, 1 – 12.

Lewis, T. G. (2006). Critical infrastructure protection in homeland security: defending a networked nation. In *Index* (p. 463-474). Monterey: John Wiley & Sons.

Lundqvist, A., Apers, P., Smeulders, A., Huizer, E., & Mandersloot, P. (2012). *Roadmap ict for the top sectors.* Amsterdam. (This booklet is published by Dutch Ministry)

Mandiant, A. (2013). *Exposing one of china's cyber espionage units* (Tech. Rep.). Mandiant. Retrieved from `http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf`

Padmanabhan, V. N., & Subramanian, L. (2001). An investigation of geographic mapping techniques for internet hosts. In *Acm sigcomm computer communication review* (Vol. 31, pp. 173–185).

Parekh, S. M., Friedman, R. B., Tibrewala, N. K., & Lutch, B. (2004, June 29). *Systems and methods for determining collecting and using geographic locations of internet users.* Google Patents. (US Patent 6,757,740)

Rana, N. S., & Panda, S. (2013). Dependency analysis of other service sectors on ict. *International Journal Of Computational Engineering Research*, *3*, 62-66.

RIPE-NCC. (2015a). *Frequently asked questios - ripe atlas - technical details.* `https://atlas.ripe.net/about/faq/`. (Accessed 8th of May, 2015)

RIPE-NCC. (2015b). *Infrastructure geolocation - plan of action - ripe labs.* `https://labs.ripe.net/Members/emileaben/infrastructure-geolocation-plan-of-action`. (Accessed 15 of May, 2015)

RIPE-NCC. (2015c). *Ripe database fast facts.* `https://www.ripe.net/manage-ips-and-asns/db/support/documentation/ripe-database-fast-facts`. (Accessed 5th of May, 2015)

RIPE-NCC. (2015d). *What is ripe atlas?* `https://atlas.ripe.net/about`. (Accessed 16 of May, 2015)

Sainsbury, A. (2013). Geolocation basics. *Gaming Law Review and Economics*, *17*(1), 33–35.

Shavitt, Y., & Zilberman, N. (2011). A geolocation databases study. *Selected Areas in Communications, IEEE Journal on*, *29*(10), 2044–2056.

Wikimedia. (2015). *File: Rir.gif.* `http://commons.wikimedia.org/wiki/File:Rir.gif`. (Accessed 3rd of April, 2015)

Wikipedia. (2015). *Loc record.* `http://en.wikipedia.org/wiki/LOC_record`. (Accessed 5th of April, 2015)