NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SOURCES OF CHANGE IN INTERNET PROTOCOL GEOLOCATION DATABASES

by

Bryan J. Kauffman

September 2021

Thesis Advisor:	Robert Beverly
Co-Advisor:	Justin P. Rohrer
Second Reader:	Thomas J. Krenc

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2021	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE
SOURCES OF CHANGE IN INTERNET PROTOCOL GEOLOCATION DATABASES

5. FUNDING NUMBERS

6. AUTHOR(S) Bryan J. Kauffman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Commercial IP geolocation databases provide a capability to associate IP addresses or prefixes to a physical location. These services provide critical information for emergency services, commercial entities, and government agencies. The accuracy of these databases can vary to a degree that degrades their utility to an unacceptable level, and the algorithms that are making location determinations are typically proprietary. This study seeks to identify patterns in, or otherwise characterize, the set of network prefixes that exhibit geolocation change between weekly snapshots of a particular commercial geolocation database: MaxMind.

We employ ground-truth correlations using active Internet measurements to characterize discernable patterns of prefix movements in the database. By measuring round-trip times from known-location vantage points to prefixes with location changes, and identifying the closest vantage point to the likely actual prefix location, we categorize and correlate possible causes of location instability in MaxMind. We find that approximately 7.5% of MaxMind prefix-location variance possibly results from geolocation granularity changes. Our methodology demonstrates a scalable technique to use Internet measurements to characterize movement shown by geolocation databases. Finally, we propose methodology enhancements for future employment.

This study illuminates the efficacy of IP geolocation databases for intelligence community, DOD, academic, and commercial use.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Internet, geolocation, Internet protocol geolocation, IP geolocation, Internet measurement, RIPE	115
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SOURCES OF CHANGE IN INTERNET PROTOCOL GEOLOCATION DATABASES

Bryan J. Kauffman
Lieutenant Commander, United States Navy
BS, The George Washington University, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Robert Beverly
Advisor

Justin P. Rohrer
Co-Advisor

Thomas J. Krenc
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Commercial IP geolocation databases provide a capability to associate IP addresses or prefixes to a physical location. These services provide critical information for emergency services, commercial entities, and government agencies. The accuracy of these databases can vary to a degree that degrades their utility to an unacceptable level, and the algorithms that are making location determinations are typically proprietary. This study seeks to identify patterns in, or otherwise characterize, the set of network prefixes that exhibit geolocation change between weekly snapshots of a particular commercial geolocation database: MaxMind.

We employ ground-truth correlations using active Internet measurements to characterize discernable patterns of prefix movements in the database. By measuring round-trip times from known-location vantage points to prefixes with location changes, and identifying the closest vantage point to the likely actual prefix location, we categorize and correlate possible causes of location instability in MaxMind. We find that approximately 7.5% of MaxMind prefix-location variance possibly results from geolocation granularity changes. Our methodology demonstrates a scalable technique to use Internet measurements to characterize movement shown by geolocation databases. Finally, we propose methodology enhancements for future employment.

This study illuminates the efficacy of IP geolocation databases for intelligence community, DOD, academic, and commercial use.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Scope 3
1.2 Summary of Contributions 3
1.3 Thesis Organization 4

2 Background **5**
2.1 Constraint-Based Geolocation 5
2.2 Database Driven Geolocation 7
2.3 Previous Work Exploring Accuracy of Commercial Databases 12

3 Methodology **15**
3.1 Overview 15
3.2 Data Collection 18
3.3 Limitations. 26

4 Results **29**
4.1 Overall Observations 29
4.2 Observations from a Single Prefix. 33
4.3 Analysis of RTT Anomalies 46
4.4 Evaluation of Granularity Effects 51

5 Conclusions and Future Work **55**
5.1 Findings and Contributions 55
5.2 Further Development. 56
5.3 Future Research 58
5.4 Conclusion. 58

Appendix: Data Collection Module Code **59**

A.1 Maxmind-data-clean.py 59
A.2 Target-pre-screen.py 62
A.3 Get-atlas-probes.py 66
A.4 Best-probe-finder.py 68
A.5 Measurement-request.py 73
A.6 Measurement-request-PARALLEL.py 76
A.7 Retrieve-Measurement-Results.py 80
A.8 Retrieve-Measurement-Results-by-tag.py 83
A.9 Retrieve-Measurement-Results-NO-MEAS-FILE.py 87

List of References **93**

Initial Distribution List **97**

List of Figures

Figure 2.1 Overview of constraint-based geolocation 6

Figure 3.1 Overview of possible prefix locations with probe locations 16

Figure 3.2 Graphic showing result of ping RTT differences for two possible prefix locations 17

Figure 3.3 Histogram of MaxMind prefix sizes 20

Figure 3.4 Overall methodology flow 22

Figure 3.5 Data collection module flow 23

Figure 4.1 CDF of probe distances to MaxMind reported prefix locations across all data collection snapshots 31

Figure 4.2 CDF of RTT percentage difference 33

Figure 4.3 Visualization of MaxMind movement data for individual prefix over course of data collection 35

Figure 4.4 Visualization of MaxMind movement data for individual prefix (week 1) 37

Figure 4.5 Visualization of MaxMind location and probes for individual prefix zoomed in on original location (week 1) 38

Figure 4.6 Visualization of MaxMind location and probes for individual prefix zoomed in on new location (week 1) 38

Figure 4.7 Visualization of MaxMind movement data for individual prefix (week 2) 40

Figure 4.8 Visualization of MaxMind location and probes for individual prefix zoomed in on original location (week 2) 41

Figure 4.9 Visualization of MaxMind location and probes for individual prefix zoomed in on new location (week 2) 41

Figure 4.10 Visualization of MaxMind movement data for individual prefix (week 3) 43

Figure 4.11 Visualization of MaxMind location and probes for individual prefix zoomed in on original location (week 3) 43

Figure 4.12 Visualization of MaxMind location and probes for individual prefix zoomed in on new location (week 3) 44

Figure 4.13 Histogram of absolute RTT differences when the MaxMind move was confirmed 47

Figure 4.14 Histogram of absolute RTT differences when the MaxMind move was not confirmed 48

Figure 4.15 Map of prefixes found in absolute RTT difference spike 49

Figure 4.16 Results of traceroute measurement against prefix contained in RTT difference spike 50

Figure 4.17 Results of traceroute from probe ID 55562 against prefix contained in RTT difference spike 51

Figure 4.18 CDF of number of occurrences for each MaxMind location . . . 52

Figure 4.19 Map of most prevalent prefix locations in Europe 53

List of Tables

Table 3.1 MaxMind data set statistics 19

Table 4.1 Data collection statistics 30

Table 4.2 Data collection prefix movement statistics 32

Table 4.3 Measurement results for 104.225.187.198/31 over the course of five weeks of data collection 34

Table 4.4 Single-week measurement results for 104.225.187.198/31 (Week 1) 36

Table 4.5 Single-week measurement results for 104.225.187.198/31 (Week 2) 39

Table 4.6 Single-week measurement results for 104.225.187.198/31 (Week 3) 42

Table 4.7 Minimum RTT and probe distance comparisons through three weeks for 104.225.187.198/31 44

Table 4.8 Granularity change impact on MaxMind movement data 54

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ACLs access control lists

ANT Analysis of Network Traffic

API application programming interface

AS autonomous system

ASN autonomous system number

BGP Border Gateway Protocol

CAIDA Center for Applied Internet Data Analysis

CBG constraint-based Internet Protocol (IP) geolocation

CDN content delivery network

CIDR Classless Inter-Domain Routing

CLLI Common Language Location Identifier

DDoS distributed denial-of-service

DNS Domain Name System

DRM digital rights management

EULA End User License Agreement

FQDN fully qualified domain name

IANA Internet Assigned Numbers Authority

IATA International Air Transport Association

ICAO International Civil Aviation Organization

ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPGeo	IP Geolocation
ISP	internet service provider
JSON	JavaScript Object Notation
NAT	Network Address Translation
PTR	pointer
RTT	round-trip time
TCP	Transmission Control Protocol
TLD	Top-Level Domain
UN/LOCODE	United Nations Code for Trade and Transport Locations
USC	University of Southern California

Acknowledgments

This thesis is dedicated to my wife Paula, whose love, patience, determination, courage, and support made this possible. I could not ask for a better partner in this Navy adventure. I love you so much, honey, and thank you for everything you do for us.

Thank you to my advisors, Dr. Beverly, Dr. Rohrer, and Dr. Krenc, for guiding me through this process. A thesis is an incredibly daunting prospect and your support pushed me to the finish line.

Thank you to Chris and Emile at RIPE NCC for your support of our research and cooperation with increasing our RIPE Atlas rate limits. RIPE Atlas is a terrific system and hopefully others will continue the work we started with you.

Thank you to the NPS faculty and staff for your mentorship and education these past two years.

Thank you to my classmates who kept me sane, lent a frequent hand when I hit a wall, and whom I counted on throughout my time at NPS. I could not have made it through this degree program without all of you.

Thank you to my parents, Beverly and James; my parents-in-law, Jesus and Rosa; sister Veronica; and the rest of our family all over the country for the continuous love and support you give Paula and me.

Finally, thank you to my superiors, peers, and especially the Sailors, Soldiers, Marines, Airmen, Coastguardsman, civilians, and contractors whom I have served with over the years. Your exertions, leadership, mentorship, and dedication made it possible for me to come here and complete this degree.

BOLD AND DARING

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Internet Protocol (IP) Geolocation (IPGeo) allows the correlation of an IP address to a physical location, a topic of great interest in both the government and commercial sectors. Commercial entities find interest in IPGeo capabilities to deliver targeted advertisements, perform large scale customer analytics, detect fraud, perform digital forensics, implement location-based pricing models, enforce digital rights management (DRM), and optimize their business operations. Government institutions utilize IPGeo data to provide emergency services and support law enforcement functions [1]. IPGeo databases provide an Internet-scale capability to associate an IP address or prefix to a real-world location. To truly understand the capabilities and limitations of these commercial activities, one needs to compare the database results to a ground truth geolocation.

These databases often display significant error and unexplained changes in the geolocations of IP addresses and prefixes. IPGeo databases frequently show significant deviations from each other, with [2] highlighting that over 50% of analyzed IP addresses displayed geolocation discrepancies of at least 100 km when compared across various IPGeo databases. Additional research demonstrates the longitudinal instability of geolocation data provided by IPGeo databases, with [3] finding that approximately 75% of MaxMind GeoLite IP addresses moved roughly twice per year. A number of factors complicate accurate IPGeo, including but not limited to:

- Continuous changes in logical network topology
- Inherent inaccuracy of latency-based distance measurements
- Circuitous paths through network infrastructure
- Inaccurate or outdated information in databases (WHOIS, DNS, etc.)
- Security policies which inhibit network measurement (for example disabling Internet Control Message Protocol [ICMP] echo replies)

Inaccurate IPGeo data can have significant implications to the previously discussed end-users. For example, law enforcement and emergency services may be unable to accurately determine the location of criminal elements or persons requiring assistance [4]. Commercial

entities can encounter difficulties enforcing DRM or providing correctly localized services. Network and security researchers may be unable to accurately analyze utilization of network resources, network topology, and sources of malicious activity [1]. With these implications in mind, we strive to better understand why longitudinal instability occurs in IPGeo databases to improve geolocation accuracy and enable users to better understand the validity of the geolocation data used.

One of the most popular open-source geolocation databases is MaxMind [5], [6]. MaxMind provides weekly geolocation data for over 99% of IP addresses in use and is utilized by thousands of businesses worldwide. The ubiquity of MaxMind makes it an ideal candidate for study. While the GeoLite2 database used for this study is open source, the algorithm and method by which the data is obtained is proprietary and unavailable for analysis.

Previous research demonstrates the longitudinal instability of the MaxMind data set [3]. Knowing that the geolocation data provided by MaxMind changes (sometimes substantially) between data sets, this study seeks to ascertain the veracity of this prefix movement, and conjecture as to the causal factors involved in this instability. Insight into the causation of this prefix movement could allow researchers to better understand and reverse-engineer the MaxMind algorithm, improving community implementation of IPGeo techniques.

To guide the structure of this study, we make the following hypothesis: temporal variations to locations as included in IPGeo databases may be due to the following factors:

- Physical (true) movement of the specific prefix (physical movement of a device with assigned IP address or reallocation of IP address space to devices in another location)
- A change in the algorithm (for example a substantive modification to the methodology used to determine a prefix's geolocation)
- A change in the inputs to the algorithm (either a change in the source of input data or a substantial alteration to the data itself)

This study seeks to identify patterns in or otherwise characterize the set of prefixes that show a geolocation change between updates of the MaxMind database. With this primary research question we aim to explain why location data changes over time in the MaxMind data set.

1.1 Scope

Our study is bounded by the following constraints:

- Analysis is limited to the free MaxMind GeoLite2 City data set. While MaxMind states that GeoLite2 is less accurate than their paid GeoIP database, our methodology is general and can be applied to any database
- Only the IPv4 portion of the data set is considered
- We ignore changes in prefix size in the database, and no not analyze instances of overlapping prefixes
- Our ground truth is inferred through ping measurements and limited by available vantage points
- Some anchor vantage-point geolocations may be incorrect
- Only prefixes with ICMP echo reachable IP addresses are analyzed

We further discuss many of these constraints in Chapter 3.

Furthermore, while we seek to correlate data trends to movement data observed in the MaxMind data set, correlation does not imply causation. Given the variety of factors influencing IPGeo determinations, and the unknown implementation of the MaxMind geolocation algorithm, our observations cannot definitively attribute the source of observed prefix movement.

1.2 Summary of Contributions

We demonstrate that the MaxMind Geolite2 City data set displays significant variation in the number of prefixes which change geolocation between snapshots, ranging between approximately .06% to 21% of the total prefixes available in the snapshot. Also, we attribute approximately 7.5% of the prefix movements analyzed to granularity changes in the MaxMind data (country-level to city-level granularity, etc.). Additionally, we evaluate the impact of last-mile effects (latency caused by last hop link characteristics) on the accuracy of constraint-based IP geolocation (CBG) and show how measurements can be correlated with database information data to characterize last-mile links. This insight provides a methodology to improve further implementations of our system by avoiding vantage points using high-latency links (specifically satellite). Finally, we demonstrate a scalable methodology

to use Internet measurements to determine the veracity of longitudinal movement shown by IPGeo databases, specifically whether the movement shown is a result of physical movement of the prefix.

1.3 Thesis Organization

Chapter 2 provides background on IPGeo, prior research in the field, and describes the MaxMind data set and RIPE Atlas system used in the study. Chapter 3 provides the methodology employed during the study, including details of the data collection, preparation, and analysis performed. Chapter 4 provides detailed analysis and results. Chapter 5 presents final conclusions and proposals for future work.

CHAPTER 2:
Background

Multiple methods currently exist to execute IPGeo. These include database methods such as WHOIS and Domain Name System (DNS), as well as CBG. Database methods make use of stored data regarding Internet registry information to obtain or infer geolocation data. In the case of WHOIS, a query about a specific domain name, IP address, or autonomous system (AS) can return information with details of the registrant of the chosen identifier. Additionally, as outlined in [7], internet service providers (ISPs) can publish privileged geolocation information regarding their IP address space to allow third parties to deliver higher quality services to their customers when those services depend on accurate geolocation data. This also aids emergency services when responding to 911 calls and other time critical situations dependent on accurate location data.

2.1 Constraint-Based Geolocation

CBG, as described in [8], calculates a geolocation estimate based on "multilateration with distance constraints." These distance constraints are obtained by measuring the total delay between a known-location source and the target. Using assumptions regarding the components of network delay described in Section 2.1.1, the delay is then converted to a distance estimate using the propagation speed of data over physical-layer mediums. By obtaining multiple distance estimates, these circles of possible locations (isochrones) are aggregated, creating an intersecting area that circumscribes the possible locations of the target.

The authors of [8] establish that the speed of digital information through a fiber optic cable as roughly 2/3 the speed of light in a vacuum. Knowing that fiber optic cables provide the highest propagation speed of physical layer mediums used by network systems, this creates an upper-bound which therefore provides the longest possible distance between the source and target.

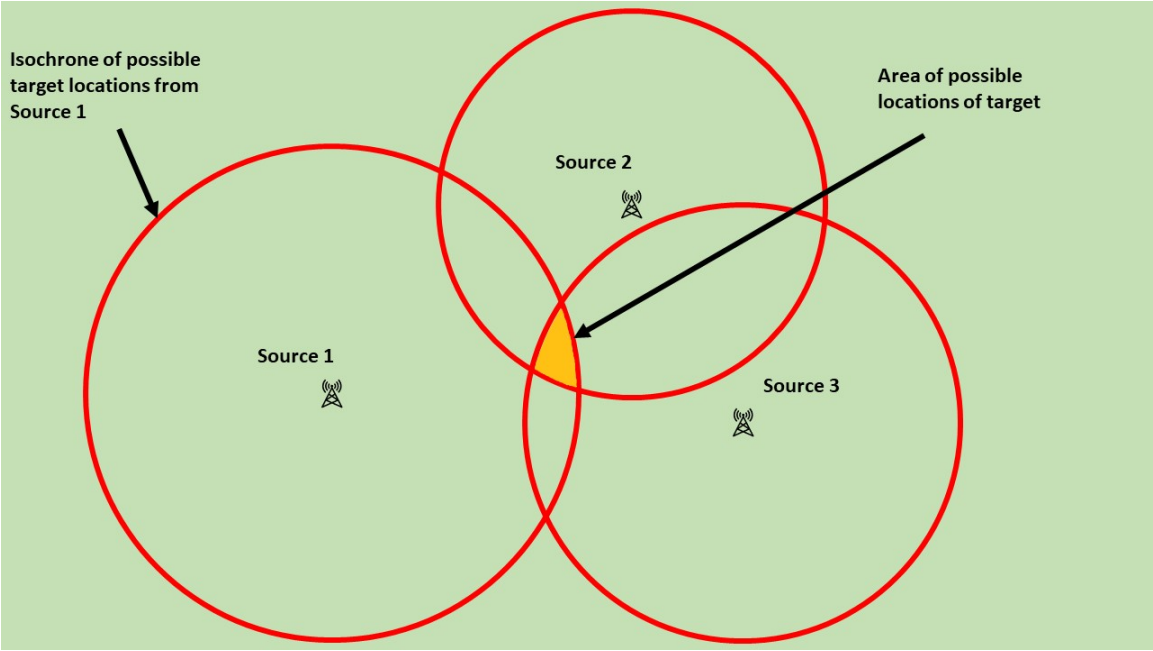

Figure 2.1. The intersection of three isochrones creates an area which contains the possible locations of a target (area shown in yellow). The source locations are indicated with antennas.

Figure 2.1 demonstrates how CBG determines the location of a target using active sources with known locations. In this case, three sources are used to measure the network delay between the sources and the target. Using an approach which obtains an upper-bound based on propagation speed and empirical network delay data, the authors of [8] determine the distance constraint within which the target should reside. By repeating this process using multiple sources (in this case three), they obtain an area representing the location estimate of the target. The centroid of this area is used as the point estimate of the target location, with the authors providing a confidence region (measured in km^2) based on the size of the area created by the intersection of the isochrones.

2.1.1 Sources of Network Delay

One common protocol used in CBG is the ICMP echo, more commonly known as a "ping." Four elements comprise a ping's round-trip time (RTT) [9]:

- Processing delay: the time for a network node to inspect packet header information to determine routing, check against access control lists (ACLs) and other filters, and other processing at the network node
- Queuing delay: the time that a packet waits at a node for the outbound interface to become available
- Transmission delay: the time it takes for a packet's data to be transmitted by the network node onto the carrying medium
- Propagation delay: the time for a single bit to travel the physical distance from one network node to another across the carrying medium

To determine physical distance using delay measurement, propagation delay must be isolated from the remaining contributing delay factors. We discuss our methodology to obtain propagation delay and convert ICMP echo RTT to distance in Chapter 3.

2.2 Database Driven Geolocation

Database driven geolocation seeks to incorporate data from a variety of sources (WHOIS, DNS, proprietary algorithms, etc.) to provide users with a centralized source for IPGeo data. Here we describe several sources used by database IPGeo providers.

2.2.1 Sources

WHOIS

The WHOIS protocol, described in [10] and [11], provides Internet resource information related to a domain name, IP address, or autonomous system number (ASN). This information is broken into the following blocks:

- Domain information
- Registrant contact information
- Administrative contact information
- Technical contact information

An example of a WHOIS lookup is shown below. Note that the query result includes the geographic location of the domain name registrant, as well as administrative and technical

contact information with geographic locations. This information can be used to infer the geolocation of the web server for this domain.

```
$whois nps.edu

Whois v1.21 - Domain information lookup
Copyright (C) 2005-2019 Mark Russinovich
Sysinternals - www.sysinternals.com

Connecting to EDU.whois-servers.net...
        Naval Postgraduate School
        555 Dyer Rd., Ingersoll 130
        Monterey, CA 93943
        USA
Administrative Contact:
        [NAME]
        [ADDRESS]
        [PHONE]
        [EMAIL]
Technical Contact:
        [NAME]
        [ADDRESS]
        [PHONE]
        [EMAIL]
Name Servers:
        NS1ERN.NPS.EDU
        NS2ERN.NPS.EDU
```

DNS

Domain names can provide low granularity geolocation information, typically down to the country level. This is particularly useful for domains which use a country-code Top-Level Domain (TLD). Reference [12] contains a list of Root Zones allocated by the Internet Assigned Numbers Authority (IANA). TLDs make up the final portion of a fully qualified domain name (FQDN) and serve as the highest level of address space segmentation in DNS [13].

These TLDs are further divided by type, with the overwhelming majority falling into the following categories [12], [14]:

- Generic (gTLD): Created based on the category of the organizations using the domain (.com, .edu, .org, .int, .net) with two reserved for US only use (.mil, .gov)
- Country Code (ccTLD): Managed by local administrators who set policies in the best interest of users within the specified country
- Sponsored (sTLD): Specialized TLD for specific user community. An organization (sponsor) executes policy for the specific sTLD

DNS can be used to geolocate FQDNs in several ways. Registry information as described in Section 2.2.1 can reveal the street-level location of the registrant. For example, the FQDN www.tsn.ca belongs to Canadian broadcaster "The Sports Network".

```
$whois tsn.ca

[....]
Registry Registrant ID: 73713289-CIRA
Registrant Name: The Sports Network Inc.
Registrant Organization:
Registrant Street: 9 Channel Nine Court
Registrant City: Scarborough
Registrant State/Province: ON
Registrant Postal Code: M1S4B5
Registrant Country: CA
Registrant Phone: +1.5148702477
[....]
```

While for many purposes country-level granularity is insufficient, for applications such as content filtering a user may find that country-level data sufficiently refines the geolocation for a specific IP address.

Another method uses inferences from the FQDN to establish geolocation with city-level granularity. Geographic hints are frequently embedded in FQDNs, with the following types often occurring [15]:

- International Air Transport Association (IATA) or International Civil Aviation Organization (ICAO) airport codes
- Common Language Location Identifier (CLLI) telecommunication equipment identifiers, often using an encoding of the city and state
- United Nations Code for Trade and Transport Locations (UN/LOCODE) identifiers used to identify major economic locations such as ports, rail terminals, etc.

For an example of this process, consider the host name "te0-3-1-5.nr51.b046470-0.dfw06.atlas.cogentco.com", taken from the 1 July, 2020 Center for Applied Internet Data Analysis (CAIDA) ARK data set [16]. The string "dfw" appears in the host name, corresponding to Dallas-Fort Worth International Airport IATA code. The CAIDA ARK data set attributes the IP address 154.24.41.2 to this host. When this IP address is submitted to the public geolocation site ipinfo.io, the returned geolocation information shows this address present in Dallas, Texas [17].

2.2.2 Limitations

IP Address / Domain Resolution

To use DNS-based geolocation against an IP address, one would need to obtain the domain name associated with the IP address in question using a DNS pointer (PTR) record. Challenges arise when attempting this methodology against a prefix containing multiple IP addresses, as one would need to account for multiple domains within a single prefix and IP addresses located at multiple locations. This process also depends on the ability to resolve the domain name to an IP address, which is not always possible.

Content Delivery Networks

A notable issue with this methodology stems from the ubiquity of content delivery networks (CDNs), where the IP address obtained through DNS returns a result to an edge server which will typically be located near the user requesting content from the desired site, as opposed to contacting a web server hosted locally by the content provider. This issue can be seen with the tsn.ca example, with content provided by the Akamai CDN:

```
$nslookup tsn.ca
```

```
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
Name:    tsn.ca
Address:  96.6.137.159

$whois 96.6.137.159

[...]
WHOIS Server: whois.akamai.com
   Registrar URL: http://www.akamai.com
   Updated Date: 2020-08-20T18:59:45Z
   Creation Date: 1998-08-18T04:00:00Z
   Registry Expiry Date: 2022-08-17T04:00:00Z
   Registrar: Akamai Technologies, Inc.
[...]
```

Additionally, CDNs often employ Anycast routing to protect against distributed denial-of-service (DDoS) attacks and spread server loads across multiple data centers. In Anycast routing, multiple hosts share the same IP address, allowing Internet routing algorithms to select the destination host with the optimal route from the client to the desired server [18]. This results in individual IP addresses residing in multiple locations, complicating accurate geolocation of the IP address.

Vanity TLDs

Vanity TLDs also impair the use of ccTLDs to infer country-level geolocation. For example, the vanity TLD "bryank.cn", available for purchase at the time of writing, would allow a US based domain to use the country-code for China. Since vanity TLDs do not need to reside in the country corresponding to their ccTLD, using country-codes to infer country-level geolocation comes with significant limitations.

2.3 Previous Work Exploring Accuracy of Commercial Databases

2.3.1 Reliability

The authors of [2] describe the overall reliability of database IPGeo sources, as well as the limitations inherent to this approach. They assessed the accuracy of several IPGeo databases through comparison of database supplied prefix locations to ground truth locations of ISP points of presence (the network interface between the ISPs and the wider Internet). Their criticisms focus on several main issues:

- Focus on densely populated countries and regions (US and Europe for example)
- Inconsistent correlation between database entries to the original allocation of IP blocks and Border Gateway Protocol (BGP) announcements
- Unrealistic claimed granularity of location accuracy

This study notes that in an effort to obtain and advertise city-level and finer granularity, their overall accuracy degrades instead of improving. This desire to obtain fine granularity IPGeo results creates unintended consequences, as IPGeo database providers are forced to choose default coordinates within countries for targets which only resolve to country-level accuracy. The choice of default coordinates within a particular country depends on several factors (geography, population distribution, etc.), with these factors described further in Section 4.4. The author of [4] describes this issue with MaxMind in particular, where MaxMind's algorithm assigned US based IP addresses with country-level granularity to a small farm in Kansas, at the approximate center of the Continental US.

The authors of [19] explore the reliability of commercial databases, including MaxMind GeoLite2 (used in our study) for performing IPGeo of network infrastructure, particularly routers. Using a database of approximately 1.64 million router interfaces, they compared these commercial databases against "ground truth" measurements consisting of a DNS-based methodology (outlined in [15]), and RIPE-Atlas measurements similar to those described in Chapter 3 of our study. This study notes the unreliability of commercial database accuracy against routers and recommends against using MaxMind GeoLite2 against this type of target as it provides insufficient city-level accuracy and coverage.

2.3.2 Stability

Research outlined in [6] demonstrates the instability of MaxMind geolocation data and makes recommendations to enable reproducible analysis when using MaxMind databases. By analyzing 214 MaxMind "snapshots" over the course of approximately ten years, the authors show that the following characteristics vary increasingly with time between snapshots:

- The difference in IP address coverage (the IP addresses present in one snapshot but not another, and vice versa)
- The overall distance difference across the covered IP space (computed as the mean log of all distance differences between the two snapshots)

The authors also demonstrate that when the maximum distance difference between snapshots is considered for every sampled IP address in a calendar year, different years demonstrate considerable variation in the longitudinal instability of locations in the MaxMind database. In particular, they show that specific prefixes can remain nearly stationary in one year, while moving up to a maximum of 1,000 km the next.

With these observations, the authors of [6] make several recommendations when using MaxMind in further research:

- When conducting Internet measurements related to MaxMind geolocation data, the measurements should be conducted in conjunction with the corresponding MaxMind snapshots to the maximum extent possible
- Research referencing MaxMind data should reference the specific dates of the snapshots used

These recommendations are intended to enhance the accuracy and reproducibility of research using MaxMind. As discussed in Chapter 3, our study follows both of these recommendations.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

3.1 Overview

This study assesses temporal geolocation variance in MaxMind data by comparison to ping RTT via RIPE Atlas. The methodology consists of two distinct phases. Data collection occurs in five iterations. We obtain two consecutive weeks of MaxMind GeoLite2 City data, extract prefixes that change location between the two snapshots, then execute ping measurements from vantage points through the RIPE Atlas application programming interface (API). During analysis, we seek to draw conclusions pertaining to correlation of the displayed MaxMind geolocation variance to the RIPE Atlas ping RTT results, and illuminate opportunities for further study using our data set and future work to optimize our process.

The core tenant underpinning our study relies on the constant propagation speed of packets across the Internet through various physical mediums. Throughout our study, several assumptions are made regarding the sources of delay discussed in Section 2.1.1:

- The ping sources (hereafter known as "probes") are reasonably close to the target, allowing us to assume an "as the crow flies" straight path between the probe and the target. Selecting the closest probes possible to the target minimizes RTT increases due to physical network topology not following straight paths (as is highly common for packets traversing long distances)
- Transmission delay is negligible due to the small size of ping packets (typically about 64 bytes when including ICMP header information)
- Processing and queuing delay are compensated by taking the minimum RTT of three measurements per probe
- Delay is symmetrical, meaning that the total delay for for packets traveling to the target host equals the total delay for packets returning to the source

With these assumptions made, we draw a direct correlation between the round trip distance from the probe to the target and back using the following formula:

$$d = \frac{2}{3}c * RTT/2$$

We therefore conclude that if a ping source has a lower RTT to a target than another source, then the probe with lowest RTT is physically closer to the target.

To illustrate how this logic applies to our study, we use the following example. In Figure 3.1, we have a single prefix, with two distinct geolocations provided by consecutive MaxMind data sets (location A and location B). To determine which location represents the most likely current location of the prefix, we select three RIPE Atlas probes per location, located as close to each MaxMind geolocation (one from each snapshot) as possible. Section 3.2.7 describes the selection methodology for these probes.

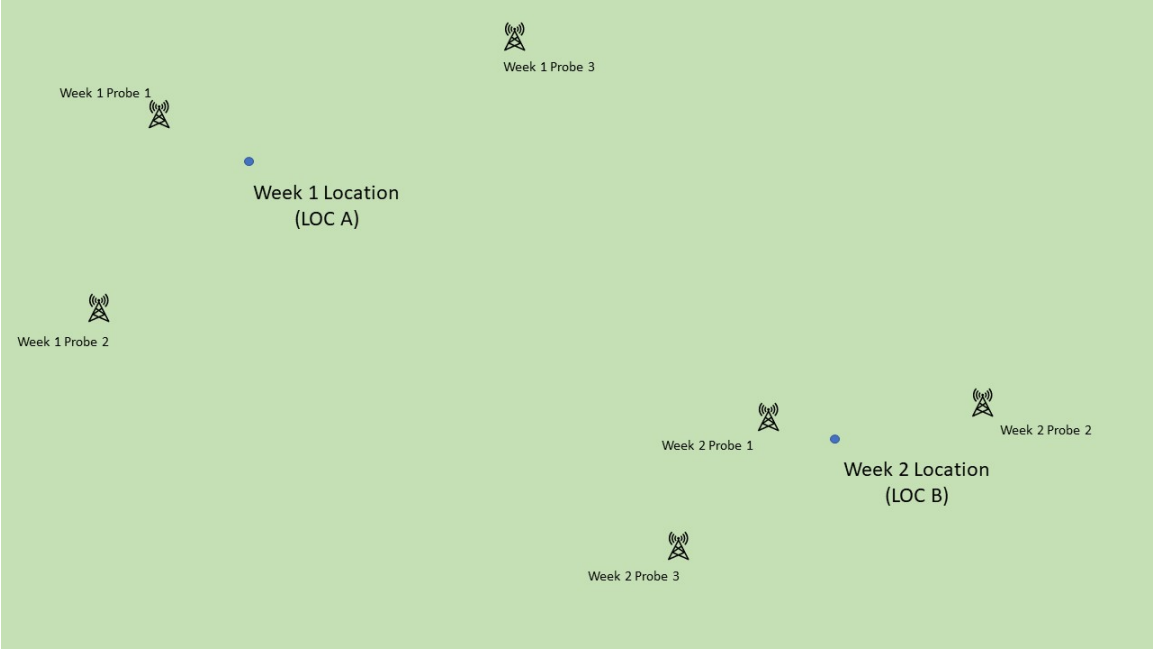

Figure 3.1. The two MaxMind indicated locations of a prefix are shown with blue dots (LOC A and LOC B). The probe locations are indicated with antennas.

We submit ping measurements to the RIPE Atlas API using each of the six probes, targeting

an IP address within the prefix (we discuss IP address selection later). We find the minimum of the minimum RTTs for each set of three probes per location (min(RTT A) and min(RTT B)). The inferred prefix location is then the location associated with the lowest RTT between min(RTT A) and min(RTT B), as illustrated in the following Figure 3.2.

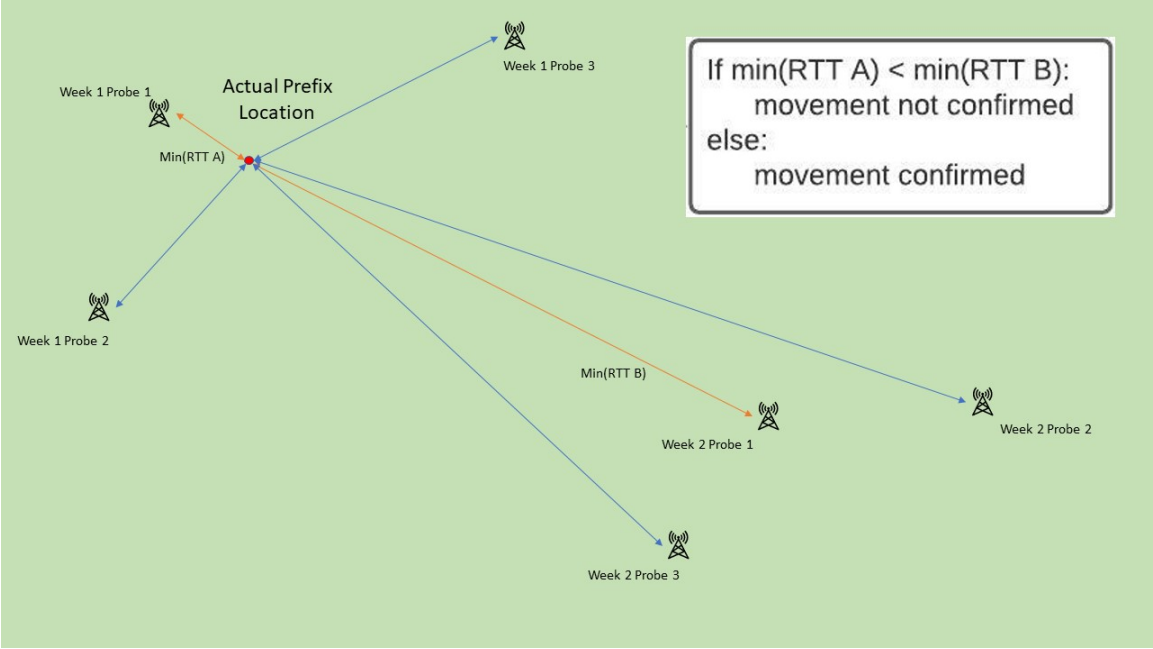

Figure 3.2. In this case, Week 1 Probe 1 possesses the minimum RTT for LOC A associated probes (Min(RTT A)), while Week 2 Probe 1 has the same for the LOC B associated probes (Min(RTT B)). Since Min(RTT A) is less than Min(RTT B), the prefix is assessed to be located at LOC A.

We then determine if the movement displayed in the MaxMind data is corroborated by the RIPE Atlas ping results. If the actual location using the steps above correlates with the current week's location in MaxMind, we call the movement "confirmed." Conversely, if the actual location correlates with the previous week's location in MaxMind, we assess the movement as "not confirmed".

We illustrate the above methodology using Algorithm 1.

Algorithm 1 Corroboration of MaxMind movement for a single prefix

```
Obtain location A (LOC A) from week 1 MaxMind data
Obtain location B (LOC B) from week 2 MaxMind data
Determine three closest active RIPE Atlas probes for LOC A and LOC B
Select target IP address
Submit ping measurement requests from each probe to the target IP address
if At least 1 ICMP echo reply for LOC A probes AND At least 1 ICMP echo reply for
LOC B probes then
    min(RTT A) ← minimum RTT for LOC A probes
    min(RTT B) ← minimum RTT for LOC B probes
    if min(RTT A) < min(RTT B) then
        Movement not confirmed
    else
        Movement confirmed
    end if
end if
```

3.2 Data Collection

The data used for analysis comes from two primary sources, the MaxMind GeoLite2 City data set, and correlated RIPE Atlas data obtained during the data collection phase of this study.

3.2.1 MaxMind

MaxMind data is available via their website and is downloaded as compressed folders containing .csv files. These data sets are updated weekly on Tuesdays. MaxMind makes several weeks of data sets available, allowing users to download the current and previous week's data easily through their web interface. In accordance with the MaxMind End User License Agreement (EULA), our measurements use MaxMind data less than 30 days old (typically less than 7-10 days old) [20]. This study uses six consecutive iterations of this data set. Table 3.1 contains key statistics of the data.

Table 3.1. MaxMind data set statistics

Date	Number of Prefixes	Prefixes with Geolocation Data	Unique Locations
03/09/21	3,472,912	3,471,178	147,207
03/16/21	3,473,766	3,472,038	147,207
03/23/21	3,430,232	3,428,496	146,695
03/30/21	3,418,348	3,416,585	146,554
04/06/21	3,369,042	3,367,333	144,688
04/13/21	3,494,500	3,492,742	147,788

"Prefixes with Geolocation Data" refers to prefixes to which MaxMind assigned a latitude and longitude. "Unique Locations" refers to the number of unique combinations of latitude and longitude within each snapshot. The combination of these two statistics results in an average of approximately 23.46 prefixes for each unique location attributed by MaxMind, indicating a substantial reuse of locations possibly caused by factors described in Section 4.4.

The prefix sizes range from /7 to /32 using Classless Inter-Domain Routing (CIDR) notation. Figure 3.3 shows a histogram of MaxMind prefix sizes observed in the six MaxMind snapshots used in this study. The x-axis indicates the CIDR notation prefix size. The overwhelming majority of prefixes range from /19 to /32, with /24 and /30 showing the highest representation in the data. /24 prefixes correspond to Class C networks in the now outmoded Classful addressing scheme, while /30 prefixes likely correspond to home or small business networks, with only two assignable IP host addresses (four addresses total with one reserved for the network and one for broadcast, not accounting for Network Address Translation (NAT) to increase available addresses).

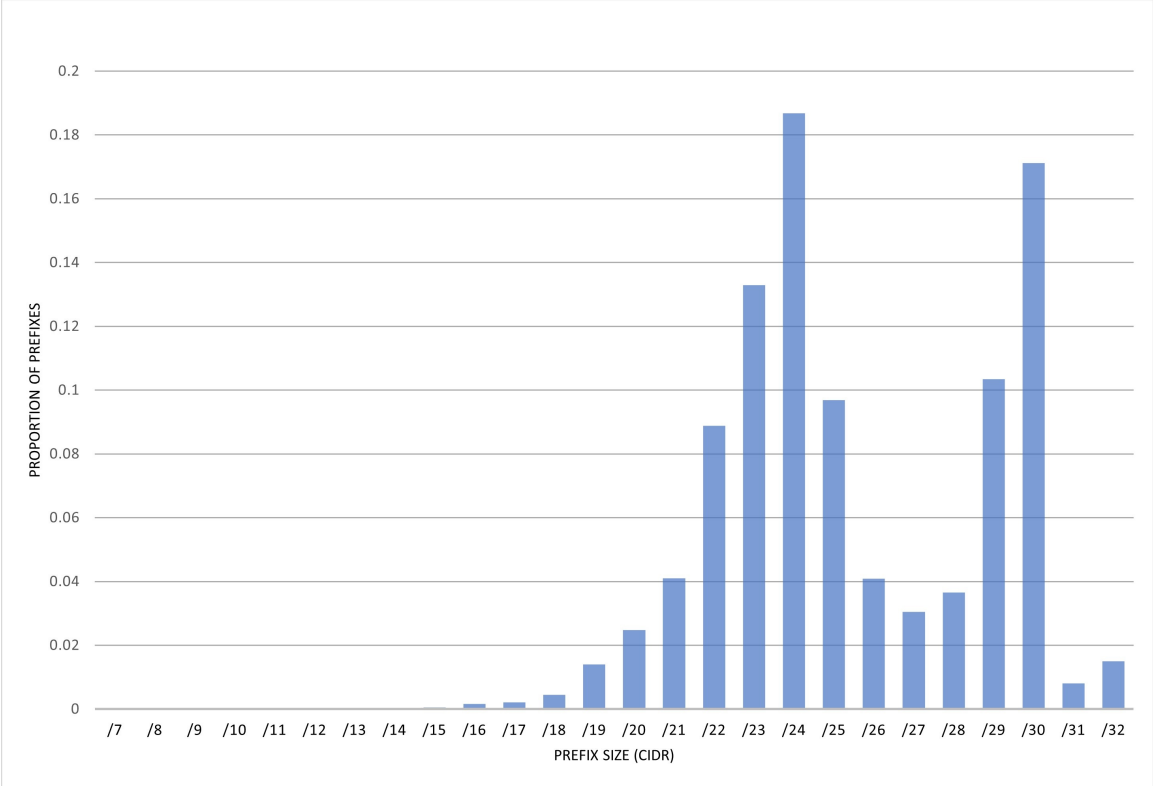

Figure 3.3. Histogram of MaxMind prefix sizes for all snapshots.

3.2.2 RIPE Atlas

The Réseaux IP Européens Network Coordination Centre (RIPE NCC) provides Regional Internet Registry (RIR) and other Internet infrastructure support throughout Europe, the Middle East, and several other regions as described in [21]. In addition to these responsibilities, RIPE coordinates and manages an Internet measurement platform known as RIPE Atlas.

RIPE Atlas is an open source Internet measurement network which employs a global network of probes that measure Internet connectivity and reachability. RIPE Atlas has over 11,000 probes spread over every continent, with the highest concentration of probes located in North America, Europe, and East Asia [22]. The global reach and fixed probe locations

RIPE Atlas employs make it ideal for conducting constraint-based IPGeo queries against specific prefixes.

In addition to providing the aforementioned Internet measurement network, RIPE Atlas developers have created and maintained two Python packages critical to this study. RIPE Atlas Cousteau, described in [23] is a Python library which provides the majority of the RIPE Atlas v2 API calls needed to submit measurement requests to RIPE Atlas. RIPE Atlas Sagan, described in [24], is a Python library containing API calls used to retrieve measurement results and parses JavaScript Object Notation (JSON) formatted results to simplify analysis of these results.

3.2.3 Process Description

The basic data collection flow employs the following steps, summarized in Figure 3.4:

1. Download two concurrent weeks of MaxMind GeoLite2 City data (for this example "Week 0" and "Week 1"). This download occurs on the day that the current week data (Week 1) is published
2. Compare the geolocation data between each data set, keeping only prefixes which demonstrate movement. (Week 0 location we will call "Location A" and Week 1 will be at "Location B")
3. Obtain a list of currently active RIPE Atlas probes with their locations
4. Determine the three closest probes to Location A and Location B respectively using methodology in Section 3.2.7
5. Submit ping measurement requests via the RIPE Atlas API using the probes found in step 4 against target IP addresses within the network prefixes from step 2 (IP address selection will be discussed in detail). The measurement request submission begins within approximately two hours after the two MaxMind data sets are compared to ensure that the data is as accurate as possible
6. Retrieve the measurement results using the RIPE Atlas API and save the data for later analysis

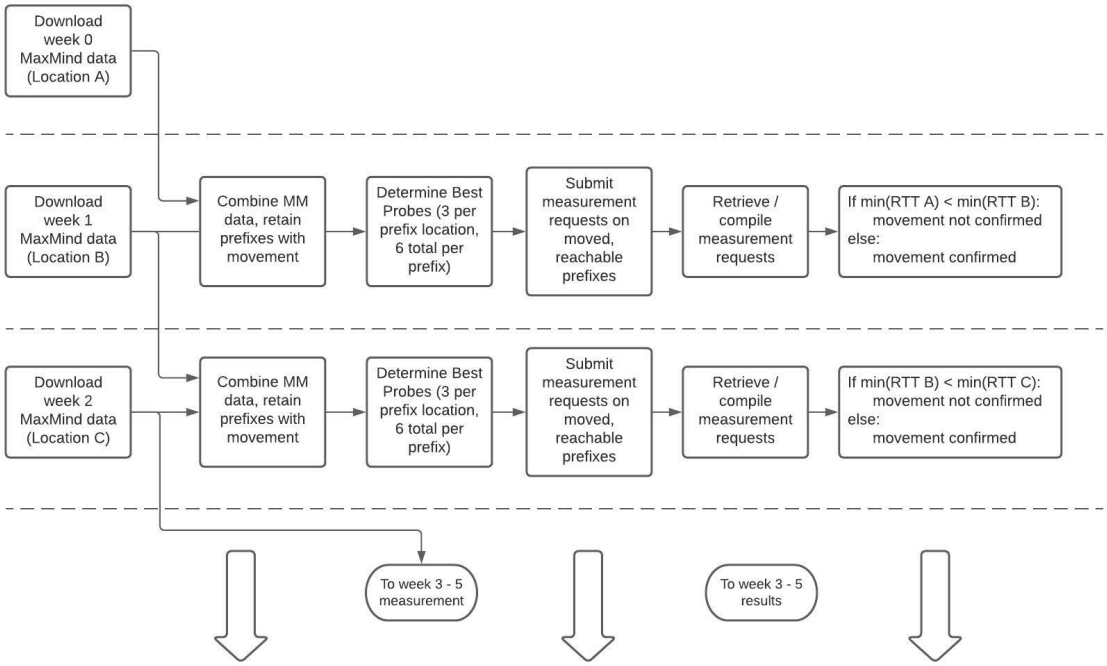

Figure 3.4. Visualization of logical workflow for data collection and analysis.

To facilitate data collection, we construct a six-step program flow using Python modules designed to each complete a step of the data collection process, visualized in Figure 3.5. Nine modules comprise the code body, with the last two steps of the program flow having two and three modules created respectively to improve performance. The modules primarily use Pandas to manage the data structures and ensure data integrity. They also make use of several Python libraries developed by RIPE Atlas (Cousteu and Sagan). We discuss each of these modules below.

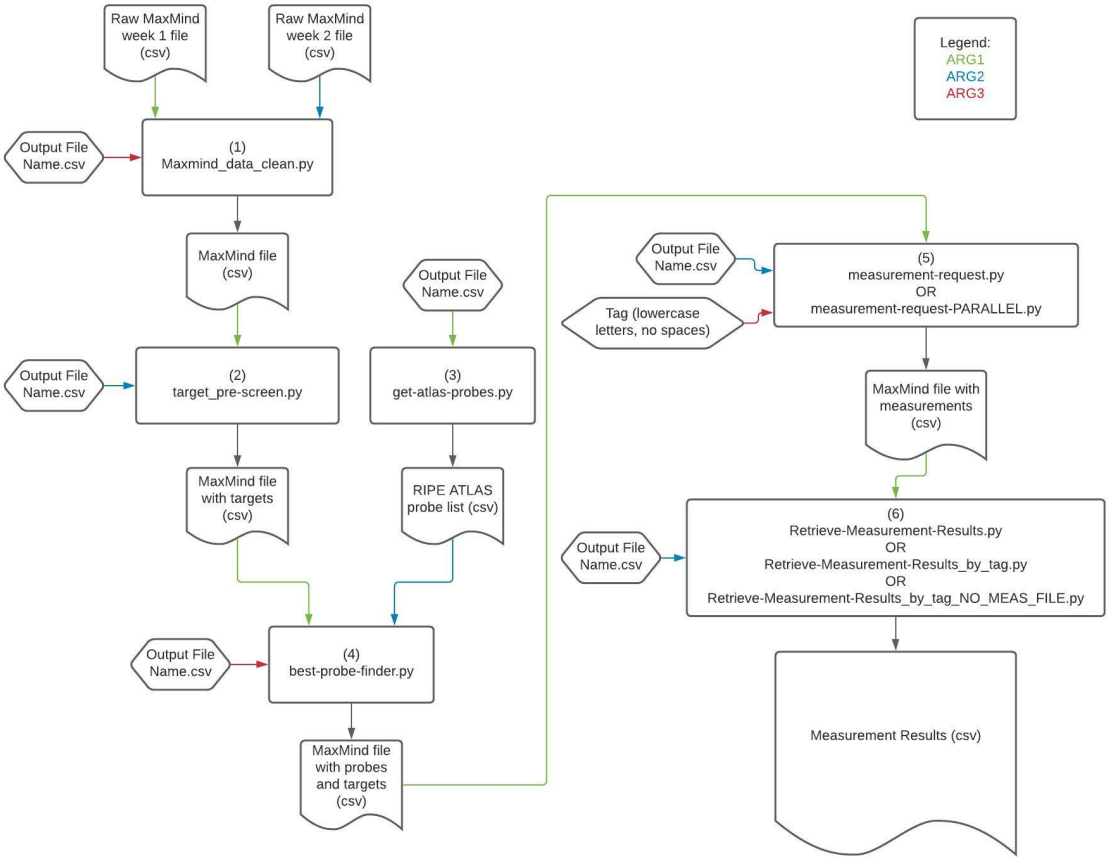

Figure 3.5. Visualization of module flow for data collection.

3.2.4 Maxmind-data-clean.py

This module reads two concurrent weeks of MaxMind data, compares the location data for each prefix in the combined data, retaining prefixes which demonstrate movement and exporting the resultant data to a .csv file. This module compares matching prefixes without compensation for prefix size changes between snapshots. For example, if the prefix 128.61.0.0/16 appeared in one week and 128.61.0.0/17 appeared in the next, they would not be compared for location change. The prefix address and length must therefore both match (through string comparison) to be considered for movement determination.

3.2.5 Target-pre-screen.py

This module provides the target IP address used for the ping measurements and attempts to increase the yield of results from the measurement requests by finding ICMP reachable target IP addresses. This step was implemented after discovering during testing that roughly 72% of the prefixes measured failed to receive an ICMP echo reply from the target and therefore did not return an RTT.

The module starts by assigning the "+1" address as the target IP for each prefix (for example if the prefix is 69.142.46.0/24, the starting address is 69.142.46.1). We then determine the reachability of these target IP addresses by sending ICMP ping requests from a non-RIPE dedicated host (a separate host used only for testing IP address reachability). If the target address is reachable, it will be used as the target address in follow-on modules. If not, the module increments the address by one (to 69.142.46.2 in this example), then re-attempts the ping. It tries up to five address before either saving the reachable IP address or determining that the prefix is unreachable. The pings are executed using parallelization to reduce runtime. We save the resultant target IP addresses with the data from the previous step in a .csv file. During testing, this module increased the number of prefixes reachable by our system by roughly 35% and reduced the number of measurements submitted to the RIPE Atlas API by approximately 52%. These results indicate an increased yield of results with reduced overhead on the RIPE Atlas system.

3.2.6 Get-atlas-probes.py

This module obtains a list of all currently active RIPE Atlas IPv4 probes. It obtains this list as individual JSON pages, extracting the needed information (probe ID and location), and exports this data to a .csv file. This module executes during each week of data collection to ensure that the probes selected by the next module are active. We note several characteristics of the reported locations of RIPE Atlas probes [25]:

- Probe locations are user-reported and not confirmed by RIPE Atlas
- Probe locations are obfuscated by RIPE Atlas for host privacy by up to 1 km

3.2.7 Best-probe-finder.py

This module uses the MaxMind data with targets from step 2, along with the RIPE Atlas probe data from step 3, to determine the three closest probes to each location provided by MaxMind (Location A and Location B) for a total of six probes. The module determines the Euclidean distance between the prefix location and each of the probe locations, using the three closest for each location and providing the distance from each prefix to each probe in kilometers for future analysis. We export this data to a .csv file. Section 4.1.1 and Figure 4.1 describe probe distance distribution as a result of this module.

3.2.8 Measurement-request.py

This module, with its related module measurement-request-PARALLEL.py, submits measurement requests via the RIPE Atlas API. The main difference between these two modules is the use of parallelization in the later to expedite the submission process. We noted a dramatic increase in the number of moved prefixes with reachable IP addresses during week 5 of data collection. In order to scale our measurements with the increase of identified prefix movements, this module can be executed in a parallelized fashion.

We use the Cousteau library [23] provided by RIPE Atlas to construct and submit each ping request. Importantly, all measurements in a group can be affixed with a "tag" that identifies the group and allows for retrieval via this tag. The module works sequentially through the data from step 4, using the target IP addresses and probe IDs to construct a properly formatted request to RIPE Atlas to begin a ping measurement. The response from RIPE Atlas contains the Measurement ID associated with a specific measurement, which can be used for measurement result retrieval. The Measurement ID is added to the data input to this module, and we save the resultant data in a .csv file.

3.2.9 Retrieve-Measurement-Results.py

This module, with its related modules Retrieve-Measurement-Results-by-tag.py and Retrieve-Measurement-Results-by-tag-NO-MEAS-FILE.py, retrieves the results for the measurements submitted to RIPE Atlas in step 5. The differences between these three modules are the method by which the measurements are retrieved, and the input file structure to the module.

The module retrieves measurement results using the Measurement IDs obtained during step 5 when the requests are submitted. Using the Sagan library from RIPE Atlas, it iterates through the list of prefixes, sequentially requesting the results for a particular Measurement ID, and retrieving the minimum RTT for each probe used in the measurement. We associate these RTTs with the applicable probe ID and save this information with the rest of the data in the final .csv file containing the results for the week's data collection. The remaining modules in this section employ modified methods to retrieve results through the RIPE Atlas API.

3.3 Limitations

This section highlights several limitations of our system to enable further development and refinement for future studies.

3.3.1 RIPE Atlas Rate Limits

A initial limitation of methodology stemmed from the need to submit large volumes of measurement requests through the RIPE Atlas API. To maintain the stability of the platform, RIPE Atlas administrators impose rate limitations on measurement requests. The rate limits that specifically impacted this study consisted of the following [26]:

- No more than 100 simultaneous measurements.
- No more than 100,000 results can be generated per day.
- No more than 1,000,000 credits may be used each day.

These rate limits created several concerns. During testing, we noticed that as measurement requests were submitted sequentially, they would pause submission approximately every 100 measurements for roughly 10 minutes. This queuing delay came as a result of the nominal runtime for measurements (approximately 10 minutes), which meant that new requests could not be successfully submitted until earlier measurements had completed. Our team met virtually with several RIPE Atlas developers early in the project to discuss the impact the simultaneous measurement limit would have on the study. The RIPE Atlas team graciously raised the simultaneous measurement limit from 100 to 750 which eliminated the queuing issues.

However, even with the increased daily use limits, other rate limitations impacted data collection. While testing the data collection modules, we observed the volume of prefixes for which MaxMind indicated movement consistently amounted to roughly 32,000 prefixes, therefore requiring an upper limit of 32,000 measurements per week, well below the daily submission limit. During data collection however, several weeks showed significantly higher numbers of prefix movement, upwards of 700,000 prefixes at maximum. While we reduced the number of submitted measurements using the pre-screening module discussed in Section 3.2.5, the measurement count still eclipsed the 100,000 per-day limit imposed by RIPE Atlas. This caused the measurement submission process to take several days to complete, resulting in data that was less timely than intended.

3.3.2 Stateless Data Collection Methodology

As described in Algorithm 1 and shown in Figure 3.4, each iteration of the data collection process compared two successive weeks of MaxMind data to determine if the movement shown over the course of one week could be correlated by RIPE Atlas ping measurements. A major limitation of the architecture as described is that the system does not maintain state from week to week. For example, if a specific prefix's movement in one week was determined to be potentially spurious based on RIPE Atlas measurements, this would have no effect on the following week's determination, as we would then compare the possibly invalid location to a new location, resulting in a correlation determination using faulty data. We discuss possible solutions to this issue in Section 4.2.5.

3.3.3 Target IP Address Selection

As discussed in Section 3.2.5, our target IP address selection depends ICMP echo requests issued by a non-RIPE dedicated host, incrementing the IP address a limited number of times until a reply is received or the prefix is deemed unreachable. This approach most likely reduced the number of prefixes measured through the RIPE Atlas API and resulted in less data for analysis, as the module only tries five IP addresses per prefix before characterizing the prefix as unreachable. We explored the use of the University of Southern California (USC) Analysis of Network Traffic (ANT) Lab's IPv4 Hitlist [27] to select target IP addresses, however due to implementation complexity we propose this approach for future development. The IPv4 Hitlist provides an ICMP echo reachable IP address in each /24 block of the

internet, updated four to six times per year. Use of the IPv4 Hitlist could improve the data yield from the measurement process in future research.

3.3.4 Prefix Size Changes

In Section 3.2.4 we discuss the methodology for finding matching prefixes for location comparison (a string comparison between prefixes from two MaxMind snapshots) and that the prefixes need to match in both address and size to undergo location comparison and measurement. We propose that follow-on researchers supplement our methodology by adding the ability to consider changing prefix sizes between MaxMind snapshots (subnet or supernet, prefix splits, etc.). Numerous methods to identify prefix overlaps exist, including Python libraries and PATRICIA trees, which could provide deeper insight as to the impact of changing prefix sizes on the longitudinal stability of the MaxMind geolocation data.

CHAPTER 4:
Results

In this Chapter we explore the results of the data collection process described in Chapter 3. Our results comprise both the macro analysis of the corpus resulting from the data collection process and micro analysis of specific instances or anomalies observed in the data.

4.1 Overall Observations

This Section provides a high level overview of the results of each step of the data collection process, along with general characterizations of the movement shown in the MaxMind data against the RIPE Atlas measurements.

4.1.1 Data Collection Parameters

As outlined in Chapter 3, each step of the data collection process refines the MaxMind data set to result in a list of prefixes with the greatest probability of returning responsive results when submitted for measurements in RIPE Atlas. Table 4.1 demonstrates the results of this process. The following terms appear in Table 4.1:

- Prefixes Moved: The number of prefixes in the MaxMind data set from the corresponding date whose position does not match the data set from the preceding week
- Reachable Prefixes: The number of moved prefixes which return successful ICMP echo replies when requests are sent from a non-RIPE dedicated host
- Responsive RIPE Results: The number of reachable prefixes for which at least one of three measurements using current week associated probes AND at least one of three measurements using previous week associated probes returns an RTT

The percentages listed describe the relative fraction of the previous body of prefixes which exhibit the listed behavior. For example, 22.77% of the prefixes which move during the 03/30/21 data collection snapshot return ICMP echo replies. The number of moved prefixes is a statistic determined by the data provided by MaxMind. In Section 3.3.3 we address methodology limitations driving the relatively low number of reachable prefixes and provide

suggestions for future improvement to target IP address selection in Section 5.2.1. The number of responsive RIPE results depends on various factors inherent to the RIPE Atlas system including changes to probe status (active or inactive) and network routing and latency between the probes and target IP.

Table 4.1. Data collection statistics

Date	Prefixes Moved	Reachable Prefixes	Responsive RIPE Results
03/16/21	2,086 (.06%)	815 (39.07%)	378 (46.38%)
03/23/21	260,902 (7.61%)	73,961 (28.35%)	5,851 (7.91%)
03/30/21	25,836 (.76%)	5,883 (22.77%)	1,829 (31.09%)
04/06/21	87,477 (2.60%)	18,935 (21.65%)	4,342 (22.93%)
04/13/21	726,603 (20.80%)	195,294 (26.88%)	31,959 (16.36%)

Figure 4.1 demonstrates the results of probe selection described in Section 3.2.7 across all snapshots. Since we aim to select probes as close to the MaxMind reported location as possible, we need to understand the effectiveness of our probe selection algorithm. The x-axis describes the distance from the self-reported probe locations to the associated MaxMind reported prefix location as defined in Section 4.1.2. Of note, 50% of selected probes reside within 13 km and nearly 90% reside within 100 km of the reported MaxMind prefix location.

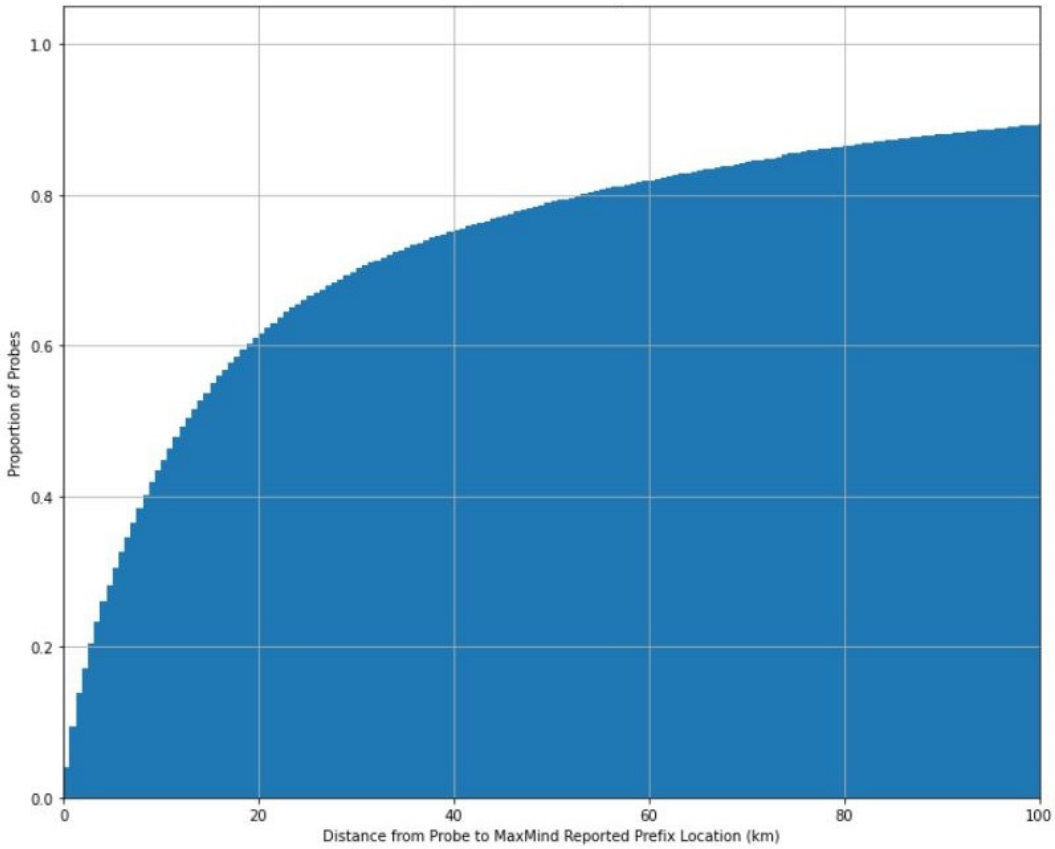

Figure 4.1. CDF of probe distances to MaxMind reported prefix locations across all data collection snapshots.

4.1.2 General Result Characterizations

Our data collection results in a corpus containing 44,359 measurements conducted over five weeks, which corresponds to the sum of the column "Responsive RIPE Results" from Table 4.1. To establish definitions used throughout this Chapter, we include the following list of terms:

- Probe association: A probe is "associated" with a given prefix location if it is one of the three geographically-closest probes active on the RIPE Atlas platform as determined by the procedure described in Section 3.2.7
- Move confirmed: The RIPE Atlas probe with the lowest minimum ping RTT associates with the new prefix location

- Move not confirmed: The RIPE Atlas probe with the lowest minimum ping RTT associates with the old prefix location

Using these definitions we determine that approximately 56% of MaxMind indicated movement is not confirmed by RIPE Atlas measurements (24,927 / 44,359). Table 4.2 details the movement characterizations for each snapshot.

Table 4.2. Data collection prefix movement statistics

Date	Responsive RIPE Results	Move Confirmed	Move Not Confirmed
03/16/21	378	293 (77.51%)	85 (22.49%)
03/23/21	5,851	2,537 (36.89%)	3,314 (63.11%)
03/30/21	1,829	805 (44.01%)	1,024 (55.99%)
04/06/21	4,342	1,953 (44.98%)	2,389 (55.02%)
04/13/21	31,959	13,844 (43.32%)	18,115 (56.68%)

Observing the relatively high rate of MaxMind reported prefix movements not confirmed through RIPE Atlas measurements, we explore an individual prefix and aberrations in our data to attempt to correlate this discrepancy.

We attempt to identify a logical threshold at which a third option, “undetermined” or some other adjective to that effect, could be implemented. Our corpus fails to reveal a threshold that could be used for this purpose, as the relative RTT differences between the two minimum RTT probes increase gradually and without a significant change, as shown in Figure 4.2. Figure 4.2 presents a CDF where the x-axis indicates the percentage difference between the two minimum RTT probes for a specific measurement.

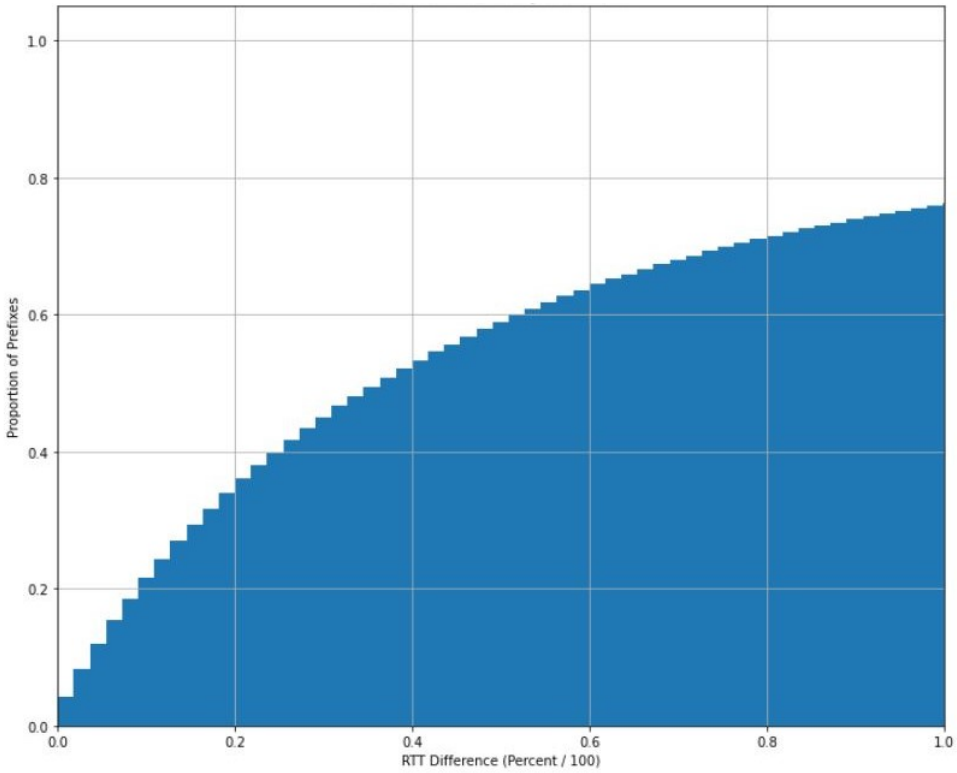

Figure 4.2. CDF of RTT percentage difference.

The lack of a noticeable "elbow" or shift in the rate of change of the proportion of prefixes as the RTT percentage difference increases precludes identification of a threshold for this third characterization.

4.2 Observations from a Single Prefix

To illustrate the progression of the data collection process, we detail the behavior observed in a single prefix. We select this prefix (104.225.187.198/31) since it demonstrates the most frequent geolocation change with the most reliable RIPE Atlas data returned of any of the prefixes observed in our study. This prefix resides in AS 396253, owned by iboss Inc., a US-based cloud network security company located in Boston, MA. Based on this information and the MaxMind provided locations (shown in Figure 4.3), this prefix likely serves as a cloud hosting server.

Over five weeks of data collection, MaxMind indicates movement for each of the first four weeks, with movement ranging from about 360 km to 1,280 km between consecutive weeks. We submit RIPE Atlas measurement requests for these first four weeks, with the fifth week not undergoing measurements in accordance with our methodology as the prefix does not indicate movement.

Figure 4.3 contains a map overlay of the MaxMind location data, along with the determinations made based on RIPE Atlas data. Of note, MaxMind returns the same geolocation for weeks two, four, and five. This location, just outside Wichita, Kansas, is the default location used by MaxMind for IP addresses with country-level geolocation granularity as described in Section 2.3 [4].

Table 4.3 contains relevant data fields demonstrating the behavior of this prefix over the course of the data collection process. We define the fields as follows:

- MaxMind Dist Diff: Euclidian distance between the original and new prefix locations according to MaxMind
- Wk1 min(RTT): Minimum RTT of the three probes associated with the original location
- Wk2 min(RTT): Minimum RTT of the three probes associated with the new location
- Move Confirmed: Determination of whether RIPE Atlas ping RTT results corroborate the MaxMind demonstrated prefix movement based on the criteria described in Section 4.1.2

Table 4.3. Measurement results for 104.225.187.198/31 over the course of five weeks of data collection

Date	MaxMind Dist Diff	Wk1 min(RTT)	Wk2 min(RTT)	Move Confirmed
03/16/21	358.79 km	8.49 ms	5.78 ms	True
03/23/21	534.42 km	6.01 ms	23.95 ms	False
03/30/21	1280.04 km	22.75 ms	19.63 ms	True
04/06/21	1280.04 km	19.74 ms	18.09 ms	True
04/13/21	0 km	N/A	N/A	N/A

RIPE Atlas measurements corroborate the movement data observed from MaxMind for weeks one, three, and four, while the movement during week two is not corroborated. To highlight the logic used to corroborate MaxMind movement data using RIPE Atlas, we detail week one, where the movement is confirmed, and week two, where the movement data in MaxMind is not confirmed by RIPE Atlas measurements. We additionally explore the impact of methodology limitations using week three results, and describe longitudinal oscillations observed during week four.

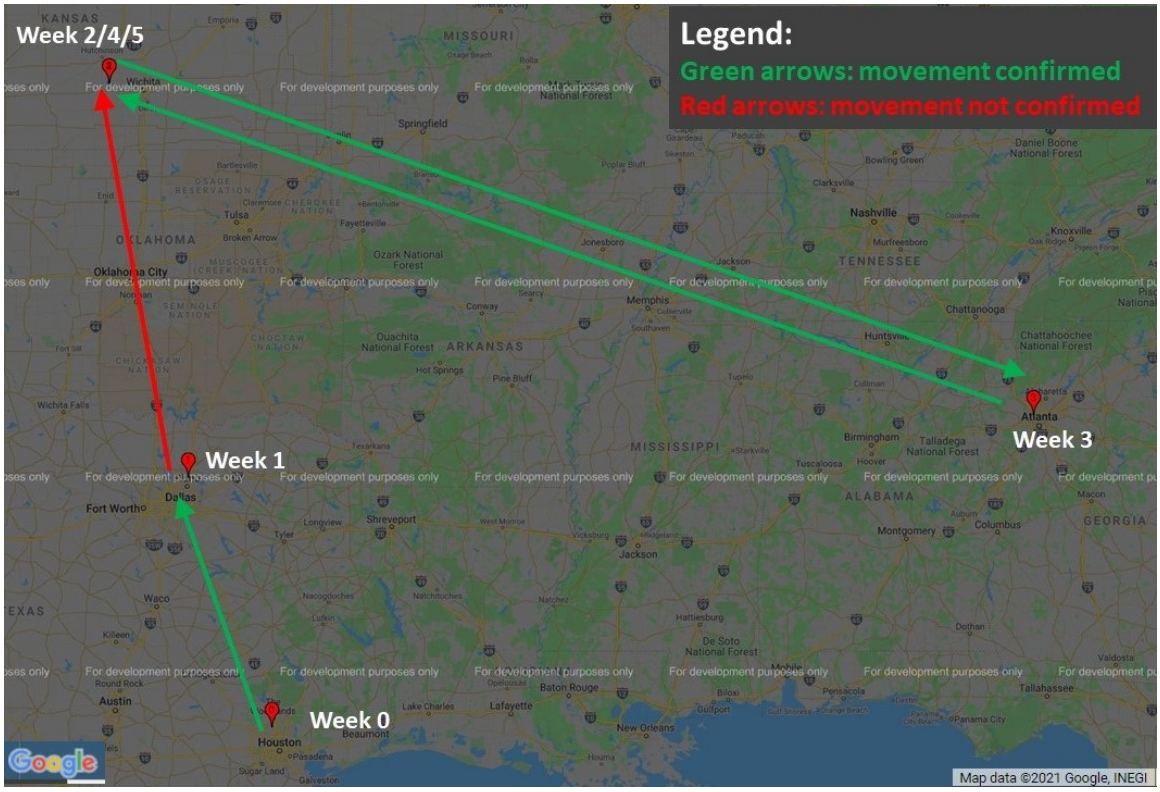

Figure 4.3. Visualization of MaxMind location data and measurement results for 104.225.187.198/31.

4.2.1 Move Confirmed: Week 1

MaxMind GeoLite2 City data for 9 March (week 0) and 16 March (week 1) 2021 reveals that our highlighted prefix moves approximately 360 km, from just north of Houston to just

outside Dallas. Prefix pre-screening determines reachability and selects target IP address (for this week 104.225.187.199). Three probes are selected closest to the two MaxMind locations, for a total of six probes. We submit a ping measurement request through the RIPE Atlas API using this target IP address and selected probes and obtain results for this measurement request.

According to the probes' self-reported location data, the probes used for this week have a mean and median distance to the MaxMind prefix location of approximately 8 km. Five of the six probes return an RTT for analysis. As shown in Table 4.4, the probe with the lowest minimum RTT is probe 2 for the week 1 location (probe ID 13283, RTT 5.78 ms). The distances shown in Table 4.4 reflect the distance from the self-reported probe location to the MaxMind prefix location for the associated snapshot (week 0 or week 1). Since the probe with the lowest RTT is associated with the new location (the current week for data collection), we assess the MaxMind indicated movement as confirmed by RIPE Atlas.

Table 4.4. Single-week measurement results for 104.225.187.198/31 (Week 1)

Probe association/number	Probe ID	Dist to MaxMind Location	Min(RTT)
Week 0 / Probe 1	6903	2.03 km	8.54 ms
Week 0 / Probe 2	27214	1.96 km	15.89 ms
Week 0 / Probe 3	50143	6.48 km	8.49 ms
Week 1 / Probe 1	1198	9.56 km	No result
Week 1 / Probe 2	13283	14.59 km	5.78 ms
Week 1 / Probe 3	50456	12.94 km	5.84 ms

We demonstrate this process using several map overlays to show the prefix and probe locations. Figure 4.4 shows an overview of the MaxMind reported prefix locations and the probes selected for use in the RIPE Atlas measurement. Figures 4.5 and 4.6 show closer vantages of the original and new prefix locations respectively.

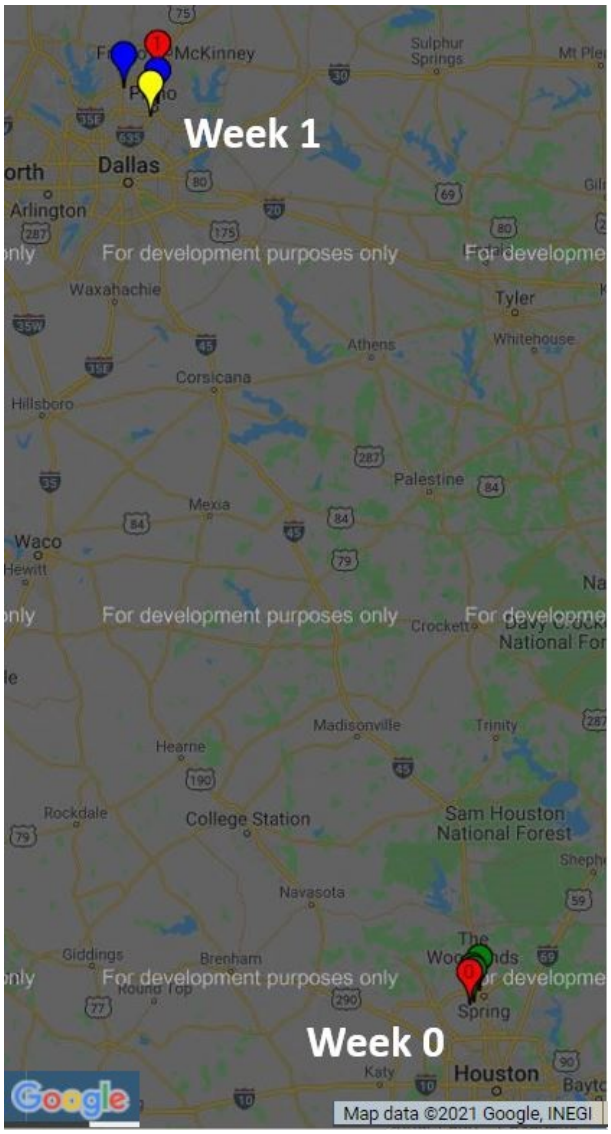

Figure 4.4. Visualization of MaxMind location data and measurement results for 104.225.187.198/31 during week 1 of data collection. The red markers represent the MaxMind reported prefix locations. The green markers represent the week 0 associated probes. The blue markers represent the week 1 associated probes. The yellow marker represents the min(RTT) probe.

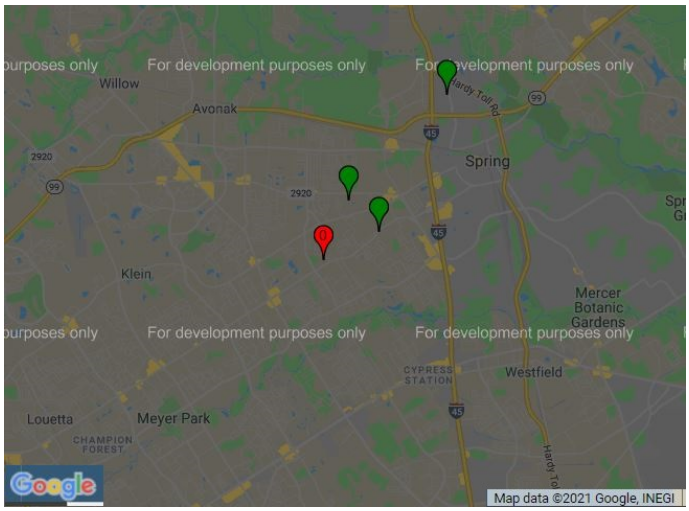

Figure 4.5. Visualization of MaxMind original location and probes for 104.225.187.198/31 during week 1 of data collection, zoomed in on original location. The red marker represents the week 0 MaxMind reported prefix location. The green markers represent the week 0 associated probes.

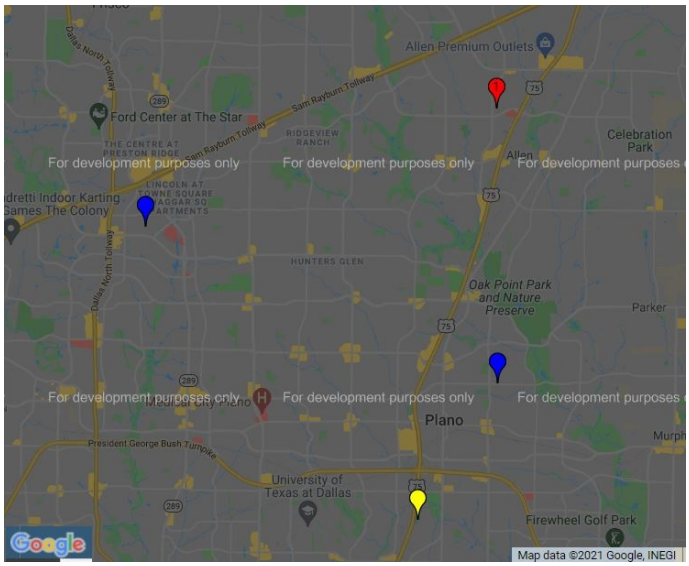

Figure 4.6. Visualization of MaxMind new location and probes for 104.225.187.198/31 during week 1 of data collection, zoomed in on new location. The red marker represents the week 1 MaxMind reported prefix location. The blue markers represent the week 1 associated probes. The yellow marker represents the min(RTT) probe.

4.2.2 Move Not Confirmed: Week 2

We execute the same process during week 2 of data collection, using MaxMind GeoLite2 City data for 16 March (week 1) and 23 March (week 2) 2021. Initial processing of these two MaxMind data sets reveals that the prefix moves approximately 525 km, from just outside Dallas to a lake just west of Wichita, Kansas. Using target IP 104.225.187.202 and with six probes selected using the module described in Section 3.2.7, we submit a ping measurement request to RIPE Atlas.

The probes used for this week have a mean distance to the prefix location of approximately 27 km and median of approximately 14 km. Four of the six probes return an RTT, with the lowest minimum RTT belonging to a probe associated with the original location (probe ID 50456, RTT 6.01 ms), as shown in Table 4.5. Since the probe with the lowest RTT is closest to the original location (the previous week's location), we assess the MaxMind indicated movement as not confirmed by RIPE Atlas.

Table 4.5. Single-week measurement results for 104.225.187.198/31 (Week 2)

Probe association/number	Probe ID	Dist to MaxMind Location	Min(RTT)
Week 1 / Probe 1	1198	9.56 km	No result
Week 1 / Probe 2	13283	14.59 km	8.74 ms
Week 1 / Probe 3	50456	12.94 km	6.01 ms
Week 2 / Probe 1	10507	0.07 km	29.88 ms
Week 2 / Probe 2	55178	53.18 km	23.95 ms
Week 2 / Probe 3	52177	73.95 km	No result

As shown for the week 1 iteration, we demonstrate this process using several map overlays to show the prefix and probe locations. Figure 4.7 shows an overview of the MaxMind reported prefix locations and the probes selected for use in the RIPE Atlas measurement. Figures 4.8 and 4.9 show closer vantages of the original and new prefix locations respectively.

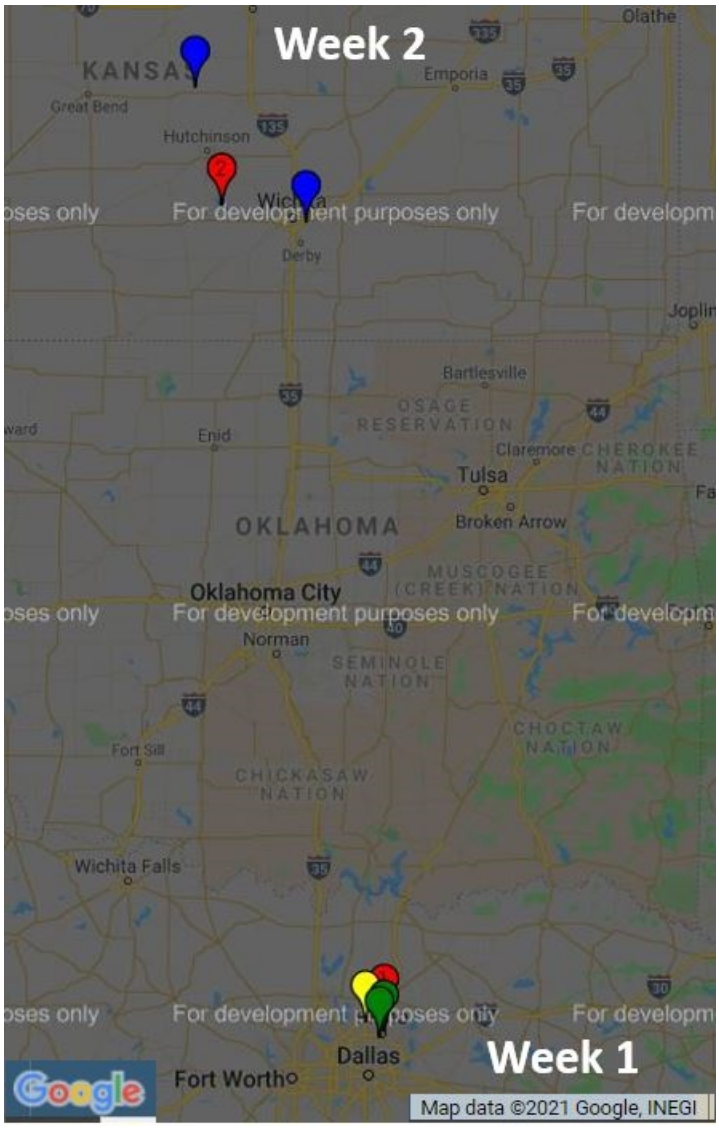

Figure 4.7. Visualization of MaxMind location data and measurement results for 104.225.187.198/31 during week 2 of data collection. The red markers represent the MaxMind reported prefix locations. The green markers represent the week 1 associated probes. The blue markers represent the week 2 associated probes. The yellow marker represents the min(RTT) probe.

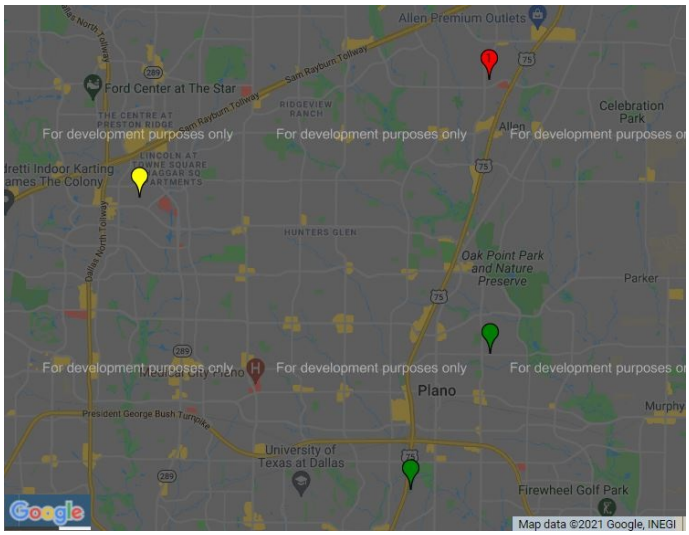

Figure 4.8. Visualization of MaxMind original location and probes for 104.225.187.198/31 during week 2 of data collection, zoomed in on original location. The red marker represents the week 1 MaxMind reported prefix location. The green markers represent the week 1 associated probes. The yellow marker represents the min(RTT) probe.

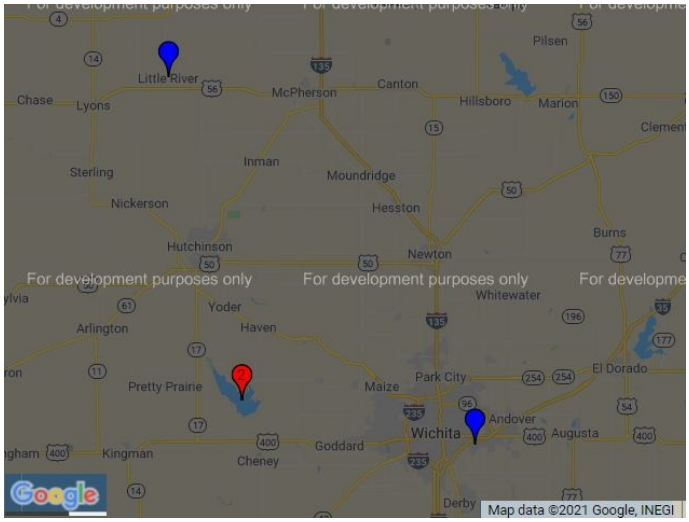

Figure 4.9. Visualization of MaxMind new location and probes for 104.225.187.198/31 during week 2 of data collection, zoomed in on new location. The red marker represents the week 2 MaxMind reported prefix location. The blue markers represent the week 2 associated probes. Note one of the probes is co-located with the prefix and the marker is obscured.

4.2.3 Move Confirmed After Previous Week Not Confirmed: Week 3

Week 3 of data collection uses MaxMind GeoLite2 City data for 23 March (week 2) and 30 March (week 3) 2021. According to MaxMind data the prefix moves approximately 1,280 km, from the location just west of Wichita, Kansas to north of Atlanta. Using target IP 104.225.187.200 and with six probes, we submit a ping measurement request to RIPE Atlas.

The probes used for this week have a mean distance to the prefix location of approximately 24 km and median of approximately 7 km. Five of the six probes return an RTT, with the lowest minimum RTT belonging to a probe associated with the new location (probe ID 6859, RTT 19.63 ms), as shown in Table 4.6. Since the probe with the lowest RTT is associated with the new location (the current week for data collection), we assess the MaxMind indicated movement as confirmed by RIPE Atlas.

Table 4.6. Single-week measurement results for 104.225.187.198/31 (Week 3)

Probe association/number	Probe ID	Dist to MaxMind Location	Min(RTT)
Week 2 / Probe 1	10507	0.07 km	64.76 ms
Week 2 / Probe 2	55178	53.18 km	22.75 ms
Week 2 / Probe 3	52177	73.95 km	No result
Week 3 / Probe 1	6859	1.97 km	19.63 ms
Week 3 / Probe 2	23903	5.58 km	38.21 ms
Week 3 / Probe 3	51006	9.21 km	30.55 ms

Figure 4.10 shows an overview of the MaxMind reported prefix locations and the probes selected for use in the RIPE Atlas measurement. Figures 4.11 and 4.12 show closer vantages of the original and new prefix locations respectively.

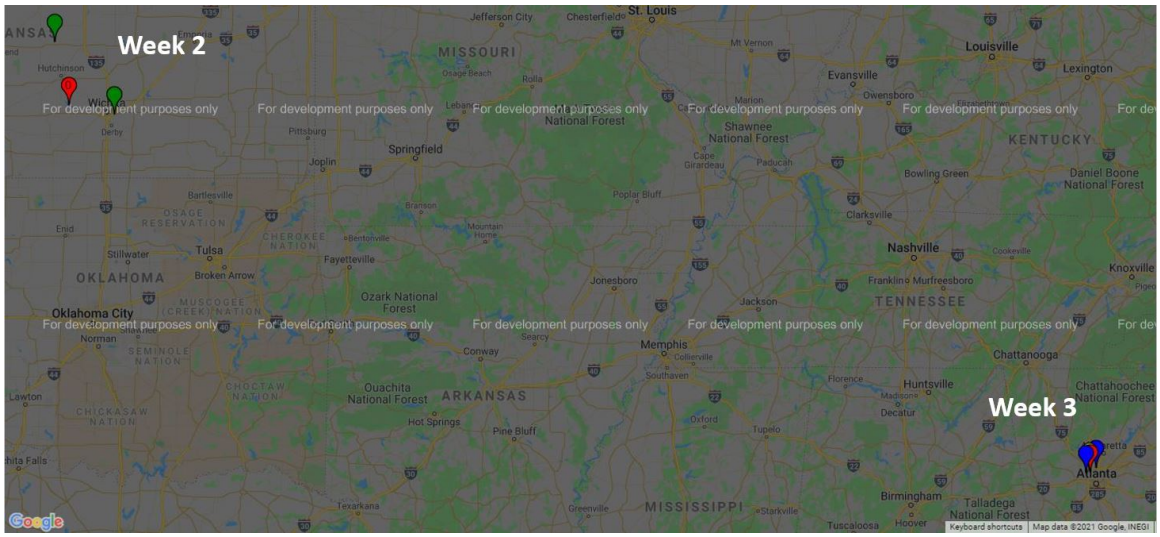

Figure 4.10. Visualization of MaxMind location data and measurement results for 104.225.187.198/31 during week 3 of data collection. The red markers represent the MaxMind reported prefix locations. The green markers represent the week 2 associated probes. The blue markers represent the week 3 associated probes. The yellow marker (obscured, located in bottom right cluster) represents the min(RTT) probe.

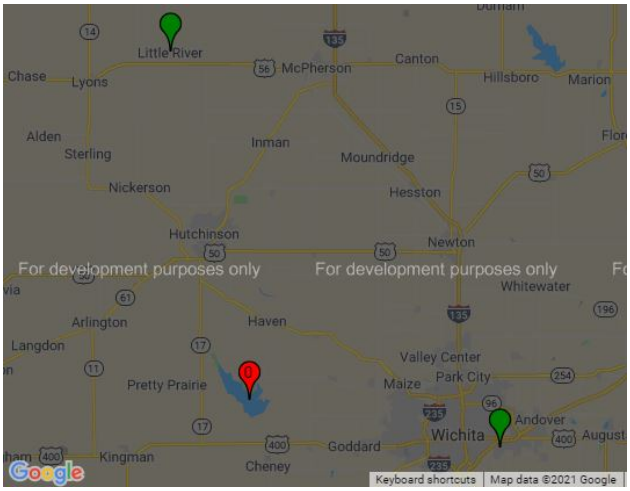

Figure 4.11. Visualization of MaxMind original location and probes for 104.225.187.198/31 during week 3 of data collection, zoomed in on original location. The red marker represents the week 2 MaxMind reported prefix location. The green markers represent the week 2 associated probes. Note one of the probes is co-located with the prefix and the marker is obscured.

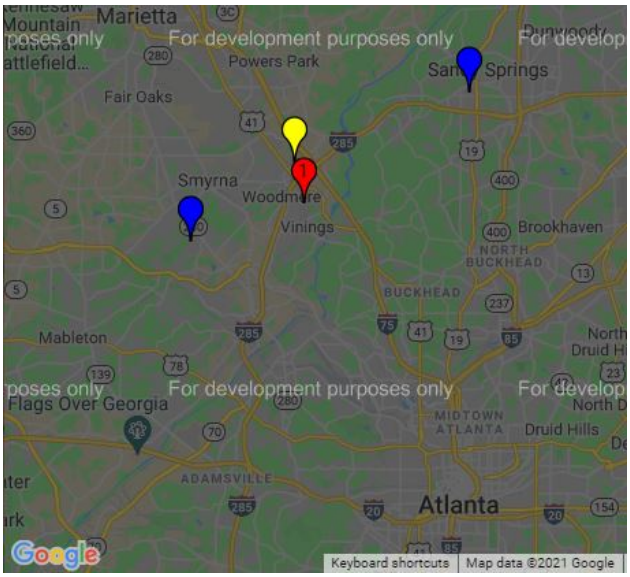

Figure 4.12. Visualization of MaxMind new location and probes for 104.225.187.198/31 during week 3 of data collection, zoomed in on new location. The red marker represents the week 3 MaxMind reported prefix location. The blue markers represent the week 3 associated probes. The yellow marker represents the min(RTT) probe.

This collection iteration highlights the limitation described in Section 3.3.2, as we determine the movement in the previous week's iteration described in Section 4.2.2 is "not confirmed" by the RIPE Atlas ping measurements. Therefore, we likely employ a possibly invalid location for the week 3 starting location, resulting in a correlation tainted by possibly faulty data. We reinforce this conclusion by observing the relative increase in the minimum RTT during the week 3 iteration compared to the previous two weeks, as shown in Table 4.7.

Table 4.7. Minimum RTT and probe distance comparisons through three weeks for 104.225.187.198/31

Data Collection Iteration	Min(RTT)	Min(RTT) Probe Dist to MaxMind Location
Week 1	5.78 ms	14.59 km
Week 2	6.01 ms	12.94 km
Week 3	19.63 ms	1.97 km

The significant increase in the minimum RTT for the week 3 iteration possibly implies that the prefix is actually located at neither the original location nor the new location used in this iteration. With a stateful data collection architecture, we could use the original location from the previous iteration (located near Dallas) to compare to the Atlanta location to determine if the prefix had actually moved from the Dallas location.

4.2.4 MaxMind Location Oscillation: Week 4

Weeks 3 and 4 demonstrate an oscillation in the MaxMind data, with the location shifting from outside Wichita, Kansas to Atlanta and back again as demonstrated in Figure 4.3. This oscillation reflects findings regarding MaxMind longitudinal instability as described in [3]. This movement could also reflect a granularity change (from city-level to country-level granularity and vice-versa), as we describe in Section 4.4.

4.2.5 Impressions from Single Prefix Analysis

After analyzing five weeks of data obtained for 104.225.187.198/31, we conjecture the following scenario to explain the behavior observed in both the MaxMind geolocation data and RIPE Atlas measurement results. We opine that the prefix actually resides at the Dallas location reflected in Section 4.2.1 and in the 16 March 2021 MaxMind data. The confirmed move during week 1 of data collection from Houston to Dallas, combined with the not confirmed move from Dallas to Kansas supports this scenario. We further reinforce this conclusion by observing the RTT increases shown in Table 4.7 and described in Section 4.2.3. The MaxMind location oscillates from the country-level granularity location in Kansas to Atlanta as the granularity improves, then back again as the granularity degrades as we discuss in Section 4.2.4. We correlate these findings by noting that the AS owner, iboss Inc., maintains a cloud data center in the Dallas area [28].

The use of a stateful data collection methodology, as discussed in Section 3.3.2, could allow us to more definitively determine if this prefix had remained stationary at the Dallas location. We propose the following mythology improvement for future implementations of our system. Perform movement characterization after every iteration of data collection. If a prefix movement is not confirmed, carry this information forward to the following data collection iteration. Use the original (confirmed) location instead of the new (not confirmed) location for measurement on the following data collection iteration. This would allow the

location resulting from a confirmed move to be compared against subsequent MaxMind reported locations to refine the actual movement of the prefix.

4.3 Analysis of RTT Anomalies

As described in Section 4.1.1, the primary data point driving the characterization of prefix movement observed in MaxMind data ("confirmed" or "not confirmed") is the minimum RTT of probes associated with the original and new location assigned to a prefix. We seek to observe the variation in the magnitude of the minimum RTT difference between original and new location probes to determine if anomalies exist in the collected data. A specific cluster of measurements displayed unusual behavior with respect to the rest of the data.

4.3.1 Absolute RTT Differences

We calculate the absolute RTT difference for each measurement through the following formula:

$$diff = abs(min(RTT)_0 - min(RTT)_1)$$

where $min(RTT)_0$ is the minimum RTT for probes associated with the original prefix location and $min(RTT)_1$ is the minimum RTT for probes associated with the new prefix location. We divide the data collection results into two categories, "move confirmed" and "move not confirmed," and plot histograms showing the absolute RTT differences for each subset. We show these histograms in Figures 4.13 and 4.14, where the x-axis indicates the RTT difference as previously defined (note that the y-axis is highly zoomed in for clarity).

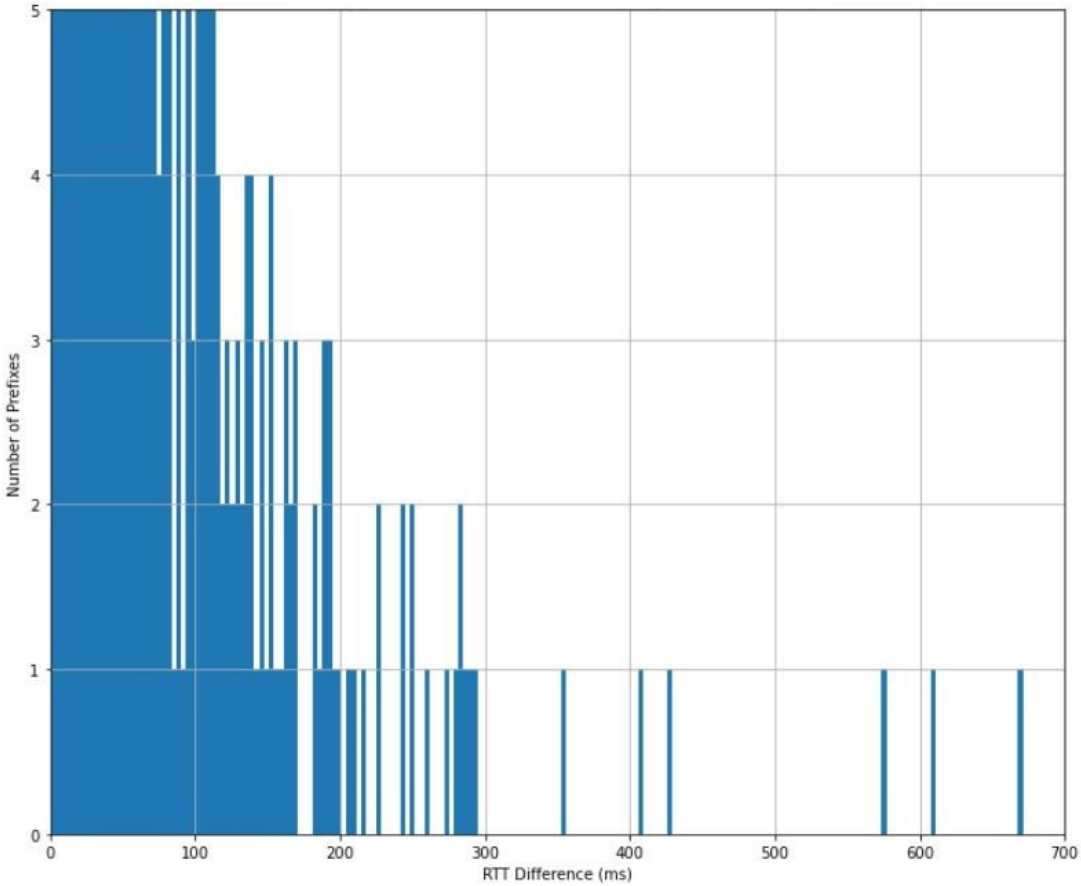

Figure 4.13. Histogram of absolute RTT differences when the MaxMind move was confirmed.

In Figure 4.13, the prefix count with respect to absolute RTT difference decreases asymptotically to zero as the absolute RTT difference increases.

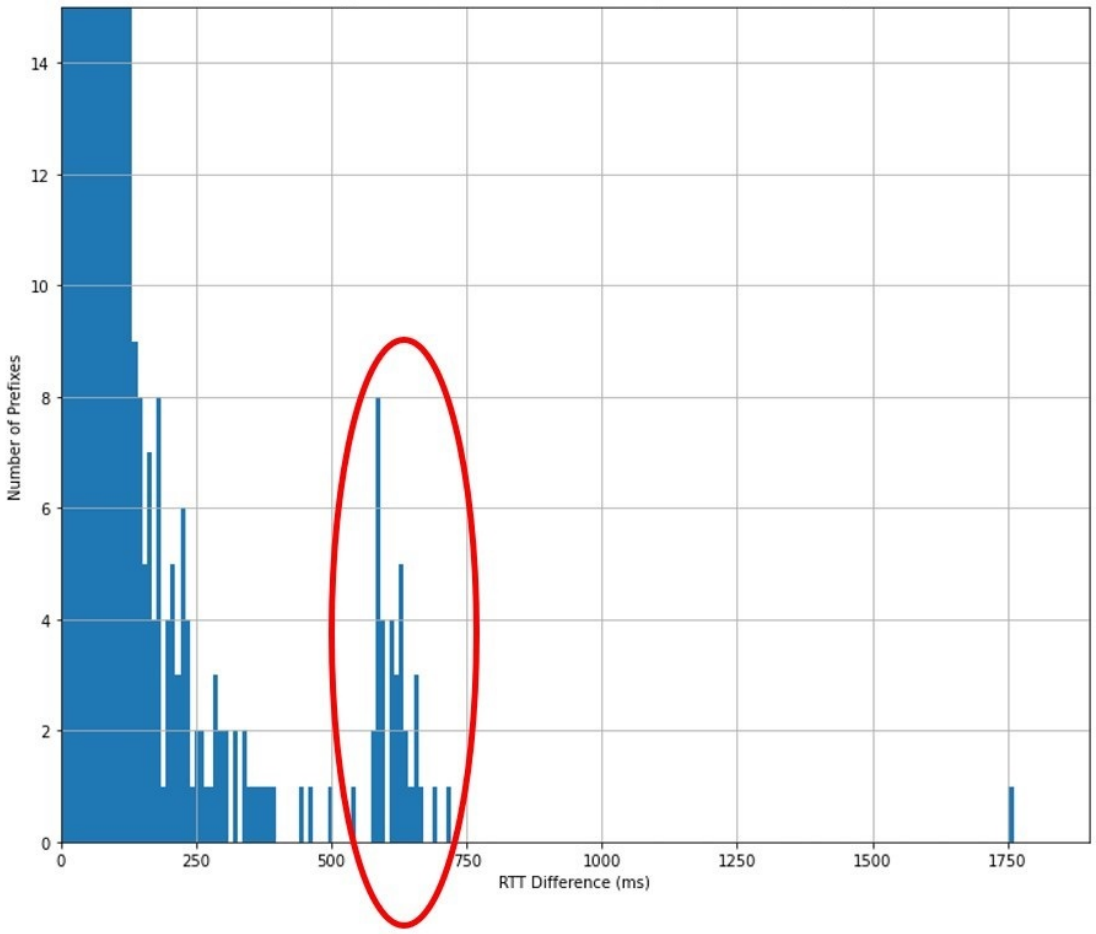

Figure 4.14. Histogram of absolute RTT differences when the MaxMind move was not confirmed. Note the prefix count spike at approximately 625ms (circled).

In Figure 4.14, we observe a spike in the number of prefixes showing an absolute RTT difference between approximately 500 and 750 milliseconds. Given this aberration, we delve further into this behavior.

4.3.2 Geospatial Analysis of RTT Difference Spike

To better understand this spike in absolute RTT difference, we isolate the prefixes showing an absolute RTT difference between 500 and 750 milliseconds in which the MaxMind

movement was not confirmed. We plot the 36 prefixes meeting this criteria on a map, with 31 of the 36 appearing in Wisconsin and the Upper Peninsula of Michigan, as shown in Figure 4.15. The other five prefixes appear in Ohio, Florida, France, Switzerland, and Italy, and are disregarded for analysis of this spike.

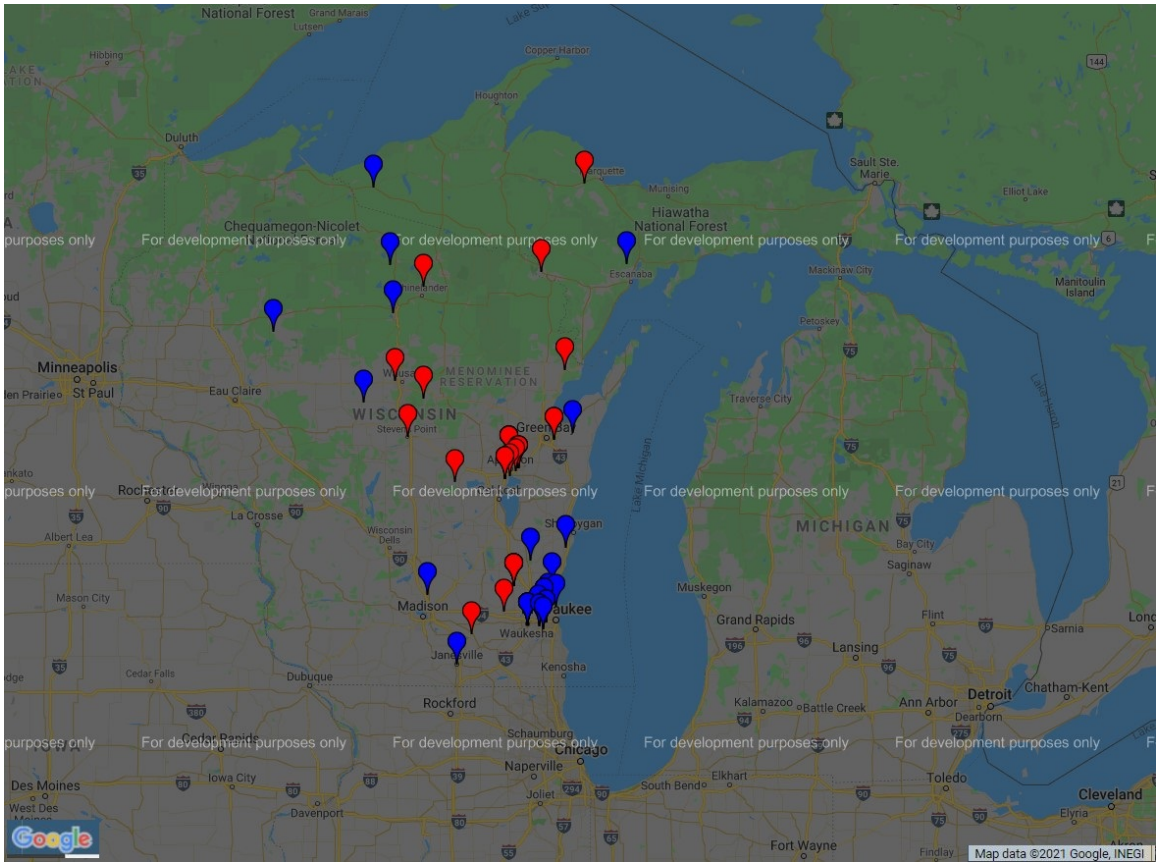

Figure 4.15. Map of prefixes found in absolute RTT difference spike. The blue markers indicate the original location for each prefix and the red markers indicate the new location for each prefix (two markers on the map per prefix unless they are so close that they overlap).

4.3.3 Probe Evaluation and Hypothesized Source of RTT Difference Spike

Upon isolating the cluster of 31 measurements located in Wisconsin and Michigan, we evaluate the measurement parameters to identify commonality. We determine that the probes used for the "current week" locations (meaning the new locations as indicated by MaxMind) are one of two probes, probe ID 100244 or probe ID 55562. Probe information, available through the RIPE Atlas browser interface, reveals that they share an IP address (99.198.53.11), are located approximately 140 meters from each other, and both belong to AS7155 (VIASAT-SP-BACKBONE).

Viasat is a satellite internet provider serving residential, commercial, and defense customers [29]. To determine if the abnormal delay demonstrated by the spike in absolute RTT difference is a result of a first hop over a satellite link, we run a traceroute measurement in RIPE Atlas using the same parameters as one of the measurements observed in the spike (same target IP address and same probes). The measurement results demonstrate the same abnormally high RTT difference as the ping measurement performed during data collection, as shown in Figure 4.16.

Probe	ASN (IPv4)	ASN (IPv6)		Time (UTC)	RTT		Hops
53493	7018	7018		2021-05-12 23:35	21.347		11
55562	7155			2021-05-12 23:35	717.810		14

Figure 4.16. Results of traceroute measurement against prefix contained in RTT difference spike. Probe 55562 shows a significantly higher RTT, consistent with ping results shown during data collection.

Further analysis of the traceroute results shows a significant delay over the second hop in the traceroute to the target IP as shown in Figure 4.17, consistent with the delay typically associated with a satellite link.

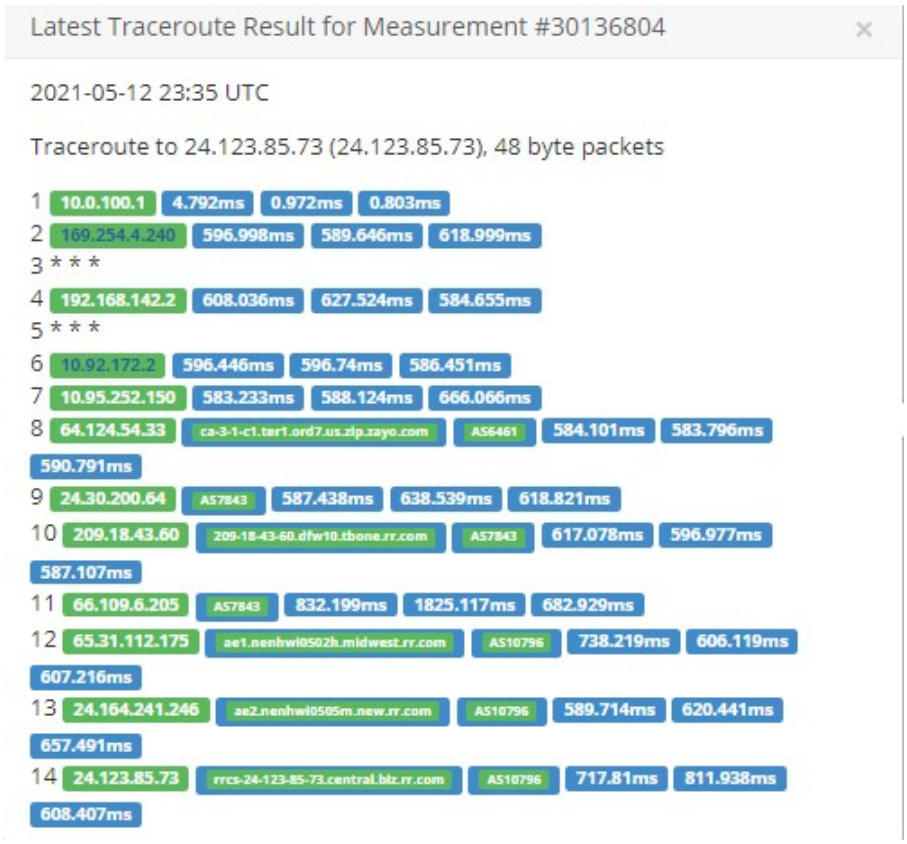

Figure 4.17. Results of traceroute from probe ID 55562 against prefix contained in RTT difference spike. The first hop out of the local network demonstrates a significant delay consistent with latency over satellite links.

Ultimately we determine that the majority of the measurements shown in the absolute RTT difference spike likely demonstrate latency inherent in satellite links. While this subset comprises a small portion of the overall corpus, this process demonstrates the ability to use our methodology to characterize links within the overall internet structure. It also provides a criteria to screen probes using satellite links from selection for use in future implementations of our system.

4.4 Evaluation of Granularity Effects

As described in [4], MaxMind appears to select a "default" location for a prefix whenever country-level granularity is the most refined result their geolocation algorithm can achieve.

For the United States, this point resides in Kansas, roughly at the geographic center of the Continental US. We hypothesize that some significant portion of the movement data observed in the corpus can be attributed to changes in the granularity of MaxMind's geolocation data (country-level to city-level or vice-versa).

We start by working to identify the default country-level location for each country using a full MaxMind data set (16 March 2021) prior to filtering for ICMP echo reachable prefixes. This data set contains 147,207 unique locations (as previously outlined in Table 3.1), and we look to identify which locations occur frequently enough to be considered as a default location. Figure 4.18, where the x-axis indicates the number times a location occurs in the MaxMind data, displays a flattening in the CDF curve at approximately 100 occurrences per location.

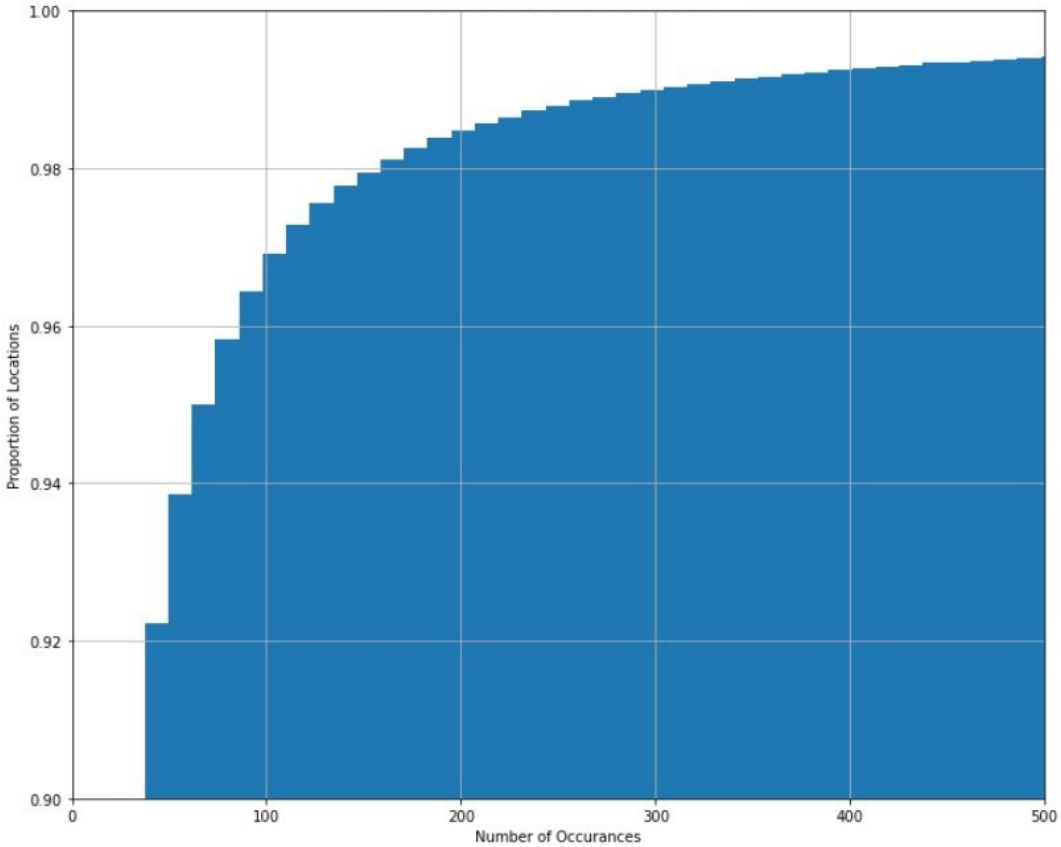

Figure 4.18. CDF of number of occurrences for each MaxMind location. The curve begins to flatten at approximately 100 occurrences per location.

With this result, we retain any locations occurring at least 100 times in the data set (yielding 5,126 locations), use a reverse geocoder to identify in which country the location resides, and select the most prevalent location for each country. This process yields a list of 136 locations, some of which we show in Figure 4.19. In contrast to the US default location (which occurs the most frequently (121,236 times or roughly 3.5% of the total data set)) many other countries appear to default to capital or most populated cities (Toronto, Mexico City, and Tokyo for example). Other locations appear to lie near the center of population for a country (such as in Henan province in China) [30]. We believe these locations probably serve as default locations due to the number of prefixes attributed to these specific latitude and longitude combinations, in contrast to a cluster of geolocation results located within populated areas.

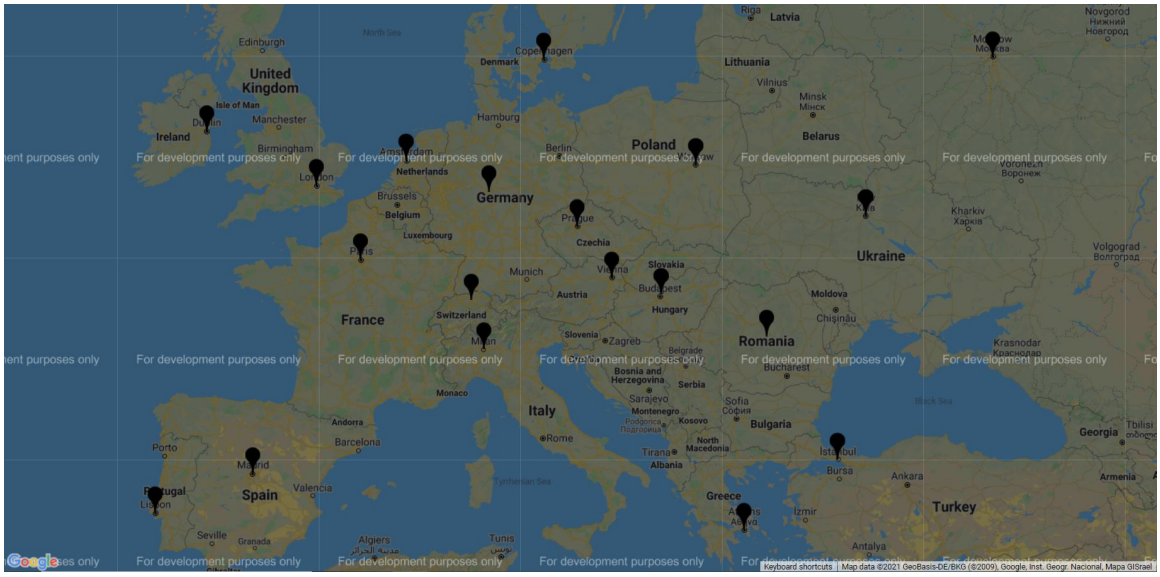

Figure 4.19. Map of most prevalent prefix locations in Europe.

Using these default locations in conjunction with the corpus collected through data collection, roughly 12% of the measurements in the corpus (4,862 / 44,359) have one of the default locations as either their previous or current location in MaxMind. Table 4.8 shows the same methodology applied to the overall MaxMind data set.

Table 4.8. Granularity change impact on MaxMind movement data

Date	Prefixes Moved	Moved Prefixes with Default Location	Percentage
03/16/21	2,086	467	21.58%
03/23/21	260,902	21,545	8.26%
03/30/21	25,836	3,746	14.50%
04/06/21	87,477	5,987	6.84%
04/13/21	726,603	50,699	6.97%

By using these default locations for comparison against the overall MaxMind data set, we determine that approximately 7.48% (82,444 / 1,102,904) of the prefix movements observed during data collection are possibly the result of granularity changes in the MaxMind algorithm, as opposed to actual physical movement of the prefix itself.

CHAPTER 5:
Conclusions and Future Work

5.1 Findings and Contributions

This study identifies behaviors inherent in the MaxMind IPGeo database, and establishes the foundation of a scalable methodology and architecture to use publicly accessible Internet measurements to assess the validity of temporal variations in IPGeo databases.

5.1.1 MaxMind Behavior

We establish that MaxMind regularly attempts to refine prefix location accuracy, with the single prefix detailed in Section 4.2 exhibiting four significant location changes in a five week period. Our analysis demonstrates that approximately 5.34% of the reported MaxMind prefix locations changed over the course of our observation (1,102,904 / 20,648,372), with single-week rates of change ranging from .06% to 20.80%. While we cannot definitively identify a cause for this location variation, we find that granularity changes could account for approximately 7.48% of the overall geolocation variation noted in the five weeks we analyzed MaxMind data. This initial insight provides the foundation for future exploration of the causes of IPGeo database location instability. We propose future research identify if this variance results from changes in BGP route advertisements, a data source which exceeds the the scope of our study.

5.1.2 Measurement Architecture

Our study demonstrates a scalable methodology and architecture to use publicly accessible Internet measurements to assess the validity of geolocation variations in IPGeo databases. Our system allows for the identification of moving prefixes across multiple snapshots of the MaxMind database. It also identifies sets of active RIPE Atlas probes near a specified location to facilitate Internet measurements. We establish an analysis methodology to exclude from these sets probes which appear to use high-latency links such as satellite to improve the effectiveness of measurements used to characterize IPGeo database behavior.

5.2 Further Development

We propose several improvements to the data collection and analysis architecture and methodology outlined in this study. These improvements exceed the time and resource constraints of our study, and future development could help refine the foundation we establish.

5.2.1 Movement Correlation Refinement

We describe the limitation imposed by our target IP address selection in Section 3.3.3, specifically the finite number of IP addresses per prefix considered for selection for RIPE Atlas ping measurements. We also discuss the use of the USC ANT Lab's IPv4 Hitlist to identify ICMP echo reachable IP addresses in /24 prefixes. Since, as shown in Figure 3.3, /24 prefixes comprise approximately 18.67% of the observed MaxMind prefixes (the most common prefix size in our data), the IPv4 Hitlist's body of reachable IP addresses in /24 prefixes could significantly increase the data yield in future research.

Other sources to aid IP address selection include fping and zmap. Fping employs ICMP echo requests to determine target prefix reachability and is designed to execute against multiple targets, which may be specified by an input file [31]. The zmap network scanner uses Transmission Control Protocol (TCP) SYN scans to determine prefix reachability, outputting reachable IP addresses within specified prefixes [32]. Both of these tools would enhance the quantity of prefixes with reachable target IP addresses for RIPE Atlas measurements and increase the overall data yield. To expedite the weekly process of finding reachable target IP addresses, we propose maintaining an asynchronous list of reachable target addresses. This list could be referenced each week as a starting point to see if previously reachable target IP addresses remain usable for measurements.

Additional sources of data to characterize prefix movement observed in MaxMind geolocation data could help to better understand possible causes behind this behavior. We discuss various IPGeo methodologies in Section 2.2, specifically utilizing DNS, and describe prior work using these methods to assess IPGeo database performance. We additionally identify BGP route advertisements as a promising avenue to help correlate prefix movement exhibited by MaxMind. We propose future researchers combine the process used in our study with these geolocation sources to further refine ground-truth comparisons against IPGeo databases.

5.2.2 Methodology Expansion

Our methodology presents several opportunities for expansion. The duration of our data collection process covered five consecutive comparisons of MaxMind data over a five week period utilizing six consecutive MaxMind snapshots. We show significant variance of several parameters in these snapshots in Tables 3.1, 4.1, and 4.2. While the duration of our data collection process allowed for methodology validation and initial correlation of data anomalies to observed movement behavior, a longer duration research effort could improve the general understanding of MaxMind's behavior.

Another avenue for expansion of our methodology arises from the process used to select prefixes for RIPE Atlas measurement described in Section 3.2.4. We only submit prefixes which demonstrate movement between MaxMind snapshots for measurement using RIPE Atlas, while disregarding prefixes that remain stationary. While this filtering reduces the number of RIPE Atlas measurements required and allows for characterization of prefix movement reflected by MaxMind, opportunities exist to determine if prefix movement occurs which MaxMind fails to identify. Performing ground truth comparison on all prefixes present in MaxMind snapshots would allow for this analysis. We propose incorporating the methods discussed in Section 5.2.1 to build this ground truth data while minimizing the excessive loading of the RIPE Atlas system.

We propose further refinement in the utilization of RIPE Atlas probes. We demonstrate in Section 4.2 that the distance between probes and the reported MaxMind prefix location varies widely. Future iterations of our methodology could account for this variation to improve the accuracy of the movement characterizations. The probe locations used in future characterization algorithms could account for the probe location obfuscation (approximately 1km) described in Section 3.2.6 by employing probe location areas instead of the single points used in our study. Finally, for MaxMind prefix movements over small distances, the probe selection algorithm could verify that probes selected for one location (either original or new) are closer to their associated location (as defined in Section 4.1.2) than the other location in the iteration, preventing probe overlap between sets.

5.3 Future Research

We identify future research avenues beyond the scope of this study. Researchers should determine if IPv6 geolocation variation in MaxMind differs from IPv4. This topic would illuminate differences in the geolocation algorithm implementation in the far more expansive IPv6 address space and enhance understanding of geolocation of IPv6 hosts. We additionally propose modification of our methodology to explore the behavior of other IPGeo databases to enhance community understanding of the processes used across various providers. Finally, we encourage employing CBG in conjunction with our methodology to further characterize IPGeo database location variation. The addition of CBG with passive BGP routing data could significantly enhance the insight our methodology provides.

5.4 Conclusion

This study enhances the understanding of the relationship between the logical and physical layers of the Internet. With this insight, we work to improve the functionality, accessibility, and security of the Internet environment for government agencies, commercial organizations, and Internet citizens alike.

APPENDIX: Data Collection Module Code

A.1 Maxmind-data-clean.py

```
# -*- coding: utf-8 -*-
"""
Takes two consecutive weeks of MaxMind data from .csv files, removes
   unneeded
data, and compares them to extract prefixes which display movement.

ARG1: Raw MAXMIND File Week 1
ARG2: Raw MAXMIND File Week 2
ARG3: Output file

"""

import sys
import pandas as pd
import numpy as np
from geopy.distance import distance
from datetime import datetime


def getPrefixes_wk1(infile):
  mm_data = pd.read_csv(infile, usecols=['network', 'latitude', '
    longitude'])
  mm_data.rename(columns={"latitude": "Week 1 Latitude", "longitude": "
    Week 1 Longitude"}, inplace = True)
  return mm_data

def getPrefixes_wk2(infile):
  mm_data = pd.read_csv(infile, index_col='network', usecols=['network',
     'latitude', 'longitude'])
  mm_data.rename(columns={"latitude": "Week 2 Latitude", "longitude": "
    Week 2 Longitude"}, inplace = True)
  return mm_data

def compilePrefixes(mm_wk1, mm_wk2):
```

```
mm_wk1_size = mm_wk1.shape[0]

for i in range(mm_wk1_size):
  net = mm_wk1.at[i, "network"]

  #DEBUGGING ONLY
  print(i, end='\r')

  mm_wk1.at[i, "MaxMind Shows Movement"] = False
  mm_wk1.at[i, "MaxMind Week 2 Data Available"] = True
  mm_wk1.at[i, "MaxMind Data Available"] = True

  try:
      wk2_lat = mm_wk2.at[net, "Week 2 Latitude"]
  except:
      mm_wk1.at[i, "MaxMind Week 2 Data Available"] = False
      mm_wk1.at[i, "MaxMind Data Available"] = False
      continue

  wk2_long = mm_wk2.at[net, "Week 2 Longitude"]
  mm_wk1.at[i, "Week 2 Latitude"] = wk2_lat
  mm_wk1.at[i, "Week 2 Longitude"] = wk2_long
  wk1_lat = mm_wk1.at[i, "Week 1 Latitude"]
  wk1_long = mm_wk1.at[i, "Week 1 Longitude"]

  is_nan_wk1lat = np.isnan(mm_wk1.at[i, "Week 1 Latitude"])
  is_nan_wk1long = np.isnan(mm_wk1.at[i, "Week 1 Longitude"])
  is_nan_wk2lat = np.isnan(mm_wk1.at[i, "Week 2 Latitude"])
  is_nan_wk2long = np.isnan(mm_wk1.at[i, "Week 2 Longitude"])

  if not is_nan_wk1lat and not is_nan_wk1long and not is_nan_wk2lat
 and not is_nan_wk2long:
      dist = float(distance((wk1_lat, wk1_long), (wk2_lat, wk2_long)).
 km)
      if dist < 0.5:
          dist = 0
      mm_wk1.at[i, "MaxMind Distance Difference"] = dist
      if dist != 0:
          mm_wk1.at[i, "MaxMind Shows Movement"] = True
  else:
```

```
        mm_wk1.at[i, "MaxMind Data Available"] = False

  print("\n")

  return mm_wk1

def writePrefixes(mm_data, outfile):

  output_prefixes = mm_data[mm_data['MaxMind Shows Movement'] == True]
  output_prefixes.set_index("network", inplace=True)
  out_size = output_prefixes.shape[0]
  print("Number of moved prefixes: ", out_size)

  #TESTING ONLY
  '''
  prefixes_sample = output_prefixes.sample(n=1000)
  sample_outfile = "sample_" + outfile
  prefixes_sample.to_csv(sample_outfile)
  '''

  #ACTUAL

  output_prefixes.to_csv(outfile)



#########################################################################

if __name__ == "__main__":
  if len(sys.argv) < 4:
    raise RuntimeError("Provide input files and output file")

  start = datetime.now()
  start_time = start.strftime("%H:%M:%S")
  print("Start Time =", start_time)

  wk1_infile = sys.argv[1]
  wk2_infile = sys.argv[2]
  outfile = sys.argv[3]

```

```
108   #ACTUAL
109
110   f = open(outfile, 'w', newline='')
111
112
113   #TESTING ONLY
114   '''
115   sample_outfile = "sample_" + outfile
116   f_s = open(sample_outfile, 'w', newline='')
117   '''
118
119   wk1_prefixes = getPrefixes_wk1(wk1_infile)
120   wk2_prefixes = getPrefixes_wk2(wk2_infile)
121   prefixes = compilePrefixes(wk1_prefixes, wk2_prefixes)
122   writePrefixes(prefixes, outfile)
123
124   #ACTUAL
125
126   f.close()
127
128
129   #TESTING ONLY
130   '''
131   f_s.close()
132   '''
133
134   end = datetime.now()
135   end_time = end.strftime("%H:%M:%S")
136   print("End Time =", end_time)
```

A.2 Target-pre-screen.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Applies pre-screening for target prefixes and assigns target IP
     addresses.  If
4 a prefix us unreachable on pings to five IP addresses within the prefix,
     the
5 prefix is deemed unreachable.  Program runs in parallel to expedite
```

```
    output.

ARG1: Maxmind Data File
ARG2: Output file

"""

import sys
from ipaddress import ip_address
import pandas as pd
from ping3 import ping
from multiprocessing import Pool, cpu_count
from datetime import datetime

def init_df_for_tgts(mm_data):

  mm_data_size = mm_data.shape[0]

  for i in range(mm_data_size):

    print("Initializing Targets, IP: ", i, end='\r')

    addr = mm_data.at[i, "network"]
    addr = addr.split("/")[0]
    addr_bin = ip_address(addr)
    addr_bin += 1

    mm_data.at[i, "Target IP"] = addr_bin

  print("\n")

  return mm_data


def ping_checks(mm_data):

  for i, row in mm_data.iterrows():
    print("Pre-Screening Targets, IP: ", i, end='\r')

    addr_bin = mm_data.at[i, "Target IP"]
```

```
    ping_result = ping(str(addr_bin))
    IP_tries = 1
    while (not ping_result) and (IP_tries < 5):
      addr_bin += 1
      ping_result = ping(str(addr_bin))
      IP_tries += 1

    mm_data.at[i, "Target IP"] = addr_bin
    mm_data.at[i, "IP Addresses Tried"] = IP_tries

    if not ping_result:
      mm_data.at[i, "Target Reachable"] = False
    else:
      mm_data.at[i, "Target Reachable"] = True

  print("\n")

  return mm_data

def parallelize_ping_checks(mm_df, func):

  mm_data_size = mm_df.shape[0]
  n_cores = 200
  print("Using %d cores\n" % n_cores)

  chunk_size = int(mm_data_size/n_cores)

  chunks = [mm_df_init.iloc[mm_df_init.index[i:i + chunk_size]] for i in
    range(0, mm_df_init.shape[0], chunk_size)]

  pool = Pool(n_cores)
  df = pd.concat(pool.map(func, chunks))
  pool.close()
  pool.join()


  return df


```

```
84 ##########################################################################
85
86 if __name__ == "__main__":
87   if len(sys.argv) < 3:
88     raise RuntimeError("Provide Input file, Output file")
89
90   start = datetime.now()
91   start_time = start.strftime("%H:%M:%S")
92   print("Start Time =", start_time)
93
94   mm_file = sys.argv[1]
95   df_outfile = sys.argv[2]
96   f = open(df_outfile, 'w', newline='')
97
98   #TESTING ONLY
99   '''
100  mm_df = pd.read_csv(mm_file)
101  mm_sample_df = mm_df.sample(n=100)
102  mm_sample_df.reset_index(drop=True, inplace=True)
103
104  mm_df_init = init_df_for_tgts(mm_sample_df)
105  output_df = parallelize_ping_checks(mm_df_init, ping_checks)
106  '''
107
108  #ACTUAL
109
110  mm_df = pd.read_csv(mm_file)
111
112  mm_df_init = init_df_for_tgts(mm_df)
113  output_df = parallelize_ping_checks(mm_df_init, ping_checks)
114
115  output_df.set_index('network', inplace=True)
116
117  output_df.to_csv(f, index_label = 'network')
118  f.close()
119
120  end = datetime.now()
121  end_time = end.strftime("%H:%M:%S")
122  print("End Time =", end_time)
```

A.3 Get-atlas-probes.py

```
#!/usr/bin/env python
"""
Adapted from the following:
Program:      $Id: fetch-atlas-v6.py $
Author:       Robert Beverly <rbeverly@nps.edu>

Searches for RIPE ATLAS probes in geographic proximity to specific geos
    of Maxmind provided prefixess

ARG1: Output file
"""


import sys
import requests
import pandas as pd
from datetime import datetime

# bryan's Atlas key
API_URL = "https://atlas.ripe.net/api/v2"
KEY = "PLACE RIPE API KEY HERE"


def getPage(page):
  payload = {'key' : KEY, 'system-ipv4-works' : 'true',  'format' : '
    json', 'page' : page}
  res = requests.get(API_URL + "/probes" , params=payload)

  if res.status_code != 200:
    print("error occurred: %s" % res.json()["error"])
    return None
  return res.json()


def getProbes(maxpages=999999999):
  i=0
  page = 1
  probes = dict()
  while True:
```

```
38      print("Fetching page %d" % (page))
39      data = getPage(page)
40      if not data:
41        break
42      if 'next' not in data:
43        break
44      for result in data['results']:
45        i+=1
46        v4addr = None
47        if 'address_v4' in result:
48          v4addr = result['address_v4']
49        status = result['status']
50        connect_status = status['id']
51        if v4addr and (connect_status == 1):
52          probe_id = result['id']
53          geometry = result['geometry']
54          if geometry:
55            coordinates = geometry['coordinates']
56            long = coordinates[0]
57            lat = coordinates[1]
58            probes[probe_id] = (v4addr, lat, long)
59
60      print("Processed %d results, found %d IPv4 probes." % (i, len(probes
        )))
61      page+=1
62      if page > maxpages:
63        break
64
65    print("Processed %d results, found %d IPv4 probes." % (i, len(probes))
       )
66    return probes
67
68
69  ######################################################################
70
71  if __name__ == "__main__":
72
73    start = datetime.now()
74    start_time = start.strftime("%H:%M:%S")
75    print("Start Time =", start_time)
```

```
76
77  if len(sys.argv) < 2:
78    raise RuntimeError("Provide output file")
79
80  outfile = sys.argv[1]
81  f = open(outfile, 'w', newline='')
82  probes = getProbes()
83  probes_df = pd.DataFrame.from_dict(probes, orient='index', columns = (
      'IP Address', 'Latitude', 'Longitude'))
84  probes_df.to_csv(f, index_label = 'Probe ID')
85  f.close()
86
87  end = datetime.now()
88  end_time = end.strftime("%H:%M:%S")
89  print("End Time =", end_time)
```

A.4 Best-probe-finder.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Finds three closest probes to each of two MaxMind locations per prefix
4
5  ARG1: Maxmind Data File with targets
6  ARG2: RIPE ATLAS Probe Data File
7  ARG3: Output file
8
9  """
10
11 import sys
12 import pandas as pd
13 import numpy as np
14 from scipy.spatial import distance
15 from geopy.distance import distance as gpdistance
16 from datetime import datetime
17
18
19 def init_df_for_probes(mm_clean):
20   mm_clean["Week 1 Probe 1 ID"] = np.nan
21   mm_clean["Week 1 Probe 1 Latitude"] = np.nan
```

```
  mm_clean["Week 1 Probe 1 Longitude"] = np.nan
  mm_clean["Week 1 Probe 1 Distance"] = np.nan

  mm_clean["Week 1 Probe 2 ID"] = np.nan
  mm_clean["Week 1 Probe 2 Latitude"] = np.nan
  mm_clean["Week 1 Probe 2 Longitude"] = np.nan
  mm_clean["Week 1 Probe 2 Distance"] = np.nan

  mm_clean["Week 1 Probe 3 ID"] = np.nan
  mm_clean["Week 1 Probe 3 Latitude"] = np.nan
  mm_clean["Week 1 Probe 3 Longitude"] = np.nan
  mm_clean["Week 1 Probe 3 Distance"] = np.nan

  mm_clean["Week 2 Probe 1 ID"] = np.nan
  mm_clean["Week 2 Probe 1 Latitude"] = np.nan
  mm_clean["Week 2 Probe 1 Longitude"] = np.nan
  mm_clean["Week 2 Probe 1 Distance"] = np.nan

  mm_clean["Week 2 Probe 2 ID"] = np.nan
  mm_clean["Week 2 Probe 2 Latitude"] = np.nan
  mm_clean["Week 2 Probe 2 Longitude"] = np.nan
  mm_clean["Week 2 Probe 2 Distance"] = np.nan

  mm_clean["Week 2 Probe 3 ID"] = np.nan
  mm_clean["Week 2 Probe 3 Latitude"] = np.nan
  mm_clean["Week 2 Probe 3 Longitude"] = np.nan
  mm_clean["Week 2 Probe 2 Distance"] = np.nan

  return mm_clean

def closest_probe(prefix_loc, probes):
  closest_index = distance.cdist([prefix_loc], probes).argmin()
  return closest_index

def determine_probes(mm_clean, probes_master):
  mm_clean_size = mm_clean.shape[0]

  for i in range(mm_clean_size):
    print(i, end='\r')

```

```
    probes = probes_master.copy()
    wk1_lat = mm_clean.at[i, 'Week 1 Latitude']
    wk1_long = mm_clean.at[i, 'Week 1 Longitude']
    wk2_lat = mm_clean.at[i, 'Week 2 Latitude']
    wk2_long = mm_clean.at[i, 'Week 2 Longitude']
    wk1_prefix_loc = (wk1_lat, wk1_long)
    wk2_prefix_loc = (wk2_lat, wk2_long)
    if (pd.isnull(wk1_lat)) or (pd.isnull(wk1_long)) or (pd.isnull(wk2_lat)) or (pd.isnull(wk2_long)):
        continue
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    #START WEEK 1
    wk1_probe_1_index = closest_probe(wk1_prefix_loc, ll_array)
    wk1_probe_1_id = probes.at[wk1_probe_1_index, 'Probe ID']
    wk1_probe_1_lat = probes.at[wk1_probe_1_index, 'Latitude']
    wk1_probe_1_long = probes.at[wk1_probe_1_index, 'Longitude']
    wk1_probe_1_dist = float(gpdistance((wk1_lat, wk1_long), (wk1_probe_1_lat, wk1_probe_1_long)).km)
    mm_clean.at[i, 'Week 1 Probe 1 ID'] = wk1_probe_1_id
    mm_clean.at[i, 'Week 1 Probe 1 Latitude'] = wk1_probe_1_lat
    mm_clean.at[i, 'Week 1 Probe 1 Longitude'] = wk1_probe_1_long
    mm_clean.at[i, 'Week 1 Probe 1 Distance'] = wk1_probe_1_dist

    probes.drop([wk1_probe_1_index], inplace = True)
    probes.reset_index(drop = True, inplace = True)
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    wk1_probe_2_index = closest_probe(wk1_prefix_loc, ll_array)
    wk1_probe_2_id = probes.at[wk1_probe_2_index, 'Probe ID']
    wk1_probe_2_lat = probes.at[wk1_probe_2_index, 'Latitude']
    wk1_probe_2_long = probes.at[wk1_probe_2_index, 'Longitude']
    wk1_probe_2_dist = float(gpdistance((wk1_lat, wk1_long), (wk1_probe_2_lat, wk1_probe_2_long)).km)
    mm_clean.at[i, 'Week 1 Probe 2 ID'] = wk1_probe_2_id
    mm_clean.at[i, 'Week 1 Probe 2 Latitude'] = wk1_probe_2_lat
    mm_clean.at[i, 'Week 1 Probe 2 Longitude'] = wk1_probe_2_long
    mm_clean.at[i, 'Week 1 Probe 2 Distance'] = wk1_probe_2_dist

    probes.drop([wk1_probe_2_index], inplace = True)
```

```
    probes.reset_index(drop = True, inplace = True)
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    wk1_probe_3_index = closest_probe(wk1_prefix_loc, ll_array)
    wk1_probe_3_id = probes.at[wk1_probe_3_index, 'Probe ID']
    wk1_probe_3_lat = probes.at[wk1_probe_3_index, 'Latitude']
    wk1_probe_3_long = probes.at[wk1_probe_3_index, 'Longitude']
    wk1_probe_3_dist = float(gpdistance((wk1_lat, wk1_long), (
wk1_probe_3_lat, wk1_probe_3_long)).km)
    mm_clean.at[i, 'Week 1 Probe 3 ID'] = wk1_probe_3_id
    mm_clean.at[i, 'Week 1 Probe 3 Latitude'] = wk1_probe_3_lat
    mm_clean.at[i, 'Week 1 Probe 3 Longitude'] = wk1_probe_3_long
    mm_clean.at[i, 'Week 1 Probe 3 Distance'] = wk1_probe_3_dist

    #START WEEK 2
    probes = probes_master.copy()
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    wk2_probe_1_index = closest_probe(wk2_prefix_loc, ll_array)
    wk2_probe_1_id = probes.at[wk2_probe_1_index, 'Probe ID']
    wk2_probe_1_lat = probes.at[wk2_probe_1_index, 'Latitude']
    wk2_probe_1_long = probes.at[wk2_probe_1_index, 'Longitude']
    wk2_probe_1_dist = float(gpdistance((wk2_lat, wk2_long), (
wk2_probe_1_lat, wk2_probe_1_long)).km)
    mm_clean.at[i, 'Week 2 Probe 1 ID'] = wk2_probe_1_id
    mm_clean.at[i, 'Week 2 Probe 1 Latitude'] = wk2_probe_1_lat
    mm_clean.at[i, 'Week 2 Probe 1 Longitude'] = wk2_probe_1_long
    mm_clean.at[i, 'Week 2 Probe 1 Distance'] = wk2_probe_1_dist

    probes.drop([wk2_probe_1_index], inplace = True)
    probes.reset_index(drop = True, inplace = True)
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    wk2_probe_2_index = closest_probe(wk2_prefix_loc, ll_array)
    wk2_probe_2_id = probes.at[wk2_probe_2_index, 'Probe ID']
    wk2_probe_2_lat = probes.at[wk2_probe_2_index, 'Latitude']
    wk2_probe_2_long = probes.at[wk2_probe_2_index, 'Longitude']
    wk2_probe_2_dist = float(gpdistance((wk2_lat, wk2_long), (
wk2_probe_2_lat, wk2_probe_2_long)).km)
    mm_clean.at[i, 'Week 2 Probe 2 ID'] = wk2_probe_2_id
```

```
    mm_clean.at[i, 'Week 2 Probe 2 Latitude'] = wk2_probe_2_lat
    mm_clean.at[i, 'Week 2 Probe 2 Longitude'] = wk2_probe_2_long
    mm_clean.at[i, 'Week 2 Probe 2 Distance'] = wk2_probe_2_dist

    probes.drop([wk2_probe_2_index], inplace = True)
    probes.reset_index(drop = True, inplace = True)
    ll_array = probes[['Latitude', 'Longitude']].to_numpy()

    wk2_probe_3_index = closest_probe(wk2_prefix_loc, ll_array)
    wk2_probe_3_id = probes.at[wk2_probe_3_index, 'Probe ID']
    wk2_probe_3_lat = probes.at[wk2_probe_3_index, 'Latitude']
    wk2_probe_3_long = probes.at[wk2_probe_3_index, 'Longitude']
    wk2_probe_3_dist = float(gpdistance((wk2_lat, wk2_long), (
   wk2_probe_3_lat, wk2_probe_3_long)).km)
    mm_clean.at[i, 'Week 2 Probe 3 ID'] = wk2_probe_3_id
    mm_clean.at[i, 'Week 2 Probe 3 Latitude'] = wk2_probe_3_lat
    mm_clean.at[i, 'Week 2 Probe 3 Longitude'] = wk2_probe_3_long
    mm_clean.at[i, 'Week 2 Probe 3 Distance'] = wk2_probe_3_dist

  mm_clean.set_index('network', inplace=True)

  return mm_clean

#####################################################################

if __name__ == "__main__":
  if len(sys.argv) < 4:
    raise RuntimeError("Provide Maxmind file, Probe file, and output
   file")

  start = datetime.now()
  start_time = start.strftime("%H:%M:%S")
  print("Start Time =", start_time)

  mm_file = sys.argv[1]
  probe_file = sys.argv[2]
  outfile = sys.argv[3]
  mm_clean_df = pd.read_csv(mm_file, usecols=['network', 'Week 1
   Latitude', 'Week 1 Longitude', "Week 2 Latitude" , "Week 2 Longitude
   ", "MaxMind Distance Difference", "Target Reachable", "Target IP", "
```

```
    IP Addresses Tried"])
172 probe_df = pd.read_csv(probe_file, usecols=['Probe ID', 'IP Address',
    'Latitude', 'Longitude'])
173 f = open(outfile, 'w', newline='')
174
175 mm_clean_df = init_df_for_probes(mm_clean_df)
176 output_df = determine_probes(mm_clean_df, probe_df)
177
178 output_df.to_csv(f, index_label = 'network')
179 f.close()
180
181 end = datetime.now()
182 end_time = end.strftime("%H:%M:%S")
183 print("End Time =", end_time)
```

A.5 Measurement-request.py

```
1  # -*- coding: utf-8 -*-
2  """
3  Program submits measurement requests through RIPE ATLAS API on reachable
4  prefixes.
5
6  ARG1: Maxmind Data File with Probes and Targets
7  ARG2: Output file
8  ARG3: Tag (lowercase letters with no spaces)
9
10 """
11
12 import sys
13 import time
14 import pandas as pd
15 import numpy as np
16 from ripe.atlas.cousteau import Ping, AtlasCreateRequest, AtlasSource
17 from datetime import datetime
18
19 KEY = "PLACE RIPE API KEY HERE"
20
21 def init_df_for_measurements(mm_data):
22
```

```
  mm_with_targets = mm_data[mm_data["Target Reachable"] == True]
  mm_with_targets.reset_index(drop=True, inplace=True)

  return mm_with_targets


def request_measurements(mm_data, tag):

  mm_data_size = mm_data.shape[0]
  mm_data["Measurement ID"] = np.nan

  for i in range(mm_data_size):
    tries = 0

    print("Request: ", i, end='\r')

    tgt = str(mm_data.at[i, "Target IP"])
    wk1_source1_ID = str(int(mm_data.at[i, "Week 1 Probe 1 ID"]))
    wk1_source2_ID = str(int(mm_data.at[i, "Week 1 Probe 2 ID"]))
    wk1_source3_ID = str(int(mm_data.at[i, "Week 1 Probe 3 ID"]))
    wk2_source1_ID = str(int(mm_data.at[i, "Week 2 Probe 1 ID"]))
    wk2_source2_ID = str(int(mm_data.at[i, "Week 2 Probe 2 ID"]))
    wk2_source3_ID = str(int(mm_data.at[i, "Week 2 Probe 3 ID"]))
    mes_description = mm_data.at[i, "network"]

    ping = Ping(af=4, target=tgt, description=mes_description, tags=[tag
    ])
    wk1_source1 = AtlasSource(type="probes", requested=1, value =
    wk1_source1_ID)
    wk1_source2 = AtlasSource(type="probes", requested=1, value =
    wk1_source2_ID)
    wk1_source3 = AtlasSource(type="probes", requested=1, value =
    wk1_source3_ID)
    wk2_source1 = AtlasSource(type="probes", requested=1, value =
    wk2_source1_ID)
    wk2_source2 = AtlasSource(type="probes", requested=1, value =
    wk2_source2_ID)
    wk2_source3 = AtlasSource(type="probes", requested=1, value =
    wk2_source3_ID)
    atlas_request = AtlasCreateRequest(
```

```
56        start_time=datetime.utcnow(),
57        key=KEY,
58        measurements=[ping],
59        sources=[wk1_source1, wk1_source2, wk1_source3, wk2_source1,
      wk2_source2, wk2_source3],
60        is_oneoff=True)
61
62      (is_success, response) = atlas_request.create()
63
64      while (not is_success) and (tries <= 60):
65        tries += 1
66        time.sleep(2)
67        atlas_request = AtlasCreateRequest(
68          start_time=datetime.utcnow(),
69          key=KEY,
70          measurements=[ping],
71          sources=[wk1_source1, wk1_source2, wk1_source3, wk2_source1,
      wk2_source2, wk2_source3],
72          is_oneoff=True)
73        (is_success, response) = atlas_request.create()
74
75      if tries >= 59:
76        continue
77
78      measurement_ID = response['measurements'][0]
79      mm_data.at[i, "Measurement ID"] = measurement_ID
80
81
82    mm_data.set_index('network', inplace=True)
83
84    return mm_data
85
86
87
88  ##########################################################################
89
90  if __name__ == "__main__":
91    if len(sys.argv) < 4:
92      raise RuntimeError("Provide Input file, Output file, and tag")
93
```

```
start = datetime.now()
start_time = start.strftime("%H:%M:%S")
print("Start Time =", start_time)

mm_file = sys.argv[1]
df_outfile = sys.argv[2]
tag = sys.argv[3]
f = open(df_outfile, 'w', newline='')

#TESTING ONLY
'''
mm_df = pd.read_csv(mm_file)
mm_sample_df = mm_df.sample(n=2)
mm_sample_df.reset_index(inplace=True)
mm_sample_df.drop(columns=['index'], inplace=True)
mm_df_init = init_df_for_measurements(mm_sample_df)
'''

#ACTUAL
mm_df = pd.read_csv(mm_file)
mm_df_init = init_df_for_measurements(mm_df)


output_df = request_measurements(mm_df_init, tag)
output_df.to_csv(f, index_label = 'network')
f.close()

end = datetime.now()
end_time = end.strftime("%H:%M:%S")
print("End Time =", end_time)
```

A.6 Measurement-request-PARALLEL.py

```
# -*- coding: utf-8 -*-
"""
Program submits measurement requests through RIPE ATLAS API on reachable
prefixes.

ARG1: Maxmind Data File with Probes and Targets
```

```
ARG2: Output file
ARG3: Tag (lowercase letters with no spaces)

"""

import sys
import time
import pandas as pd
import numpy as np
from ripe.atlas.cousteau import Ping, AtlasCreateRequest, AtlasSource
from multiprocessing import Pool, cpu_count
from datetime import datetime

KEY = "PLACE RIPE API KEY HERE"

def init_df_for_measurements(mm_data):

  mm_with_targets = mm_data[mm_data["Target Reachable"] == True]
  mm_with_targets.reset_index(drop=True, inplace=True)

  return mm_with_targets


def request_measurements(mm_data, tag = "onethreeapriltest"):

  mm_data_size = mm_data.shape[0]
  mm_data["Measurement ID"] = np.nan

  for i, row in mm_data.iterrows():
    tries = 0

    print("Request: ", i, end='\r')

    tgt = str(mm_data.at[i, "Target IP"])
    wk1_source1_ID = str(int(mm_data.at[i, "Week 1 Probe 1 ID"]))
    wk1_source2_ID = str(int(mm_data.at[i, "Week 1 Probe 2 ID"]))
    wk1_source3_ID = str(int(mm_data.at[i, "Week 1 Probe 3 ID"]))
    wk2_source1_ID = str(int(mm_data.at[i, "Week 2 Probe 1 ID"]))
    wk2_source2_ID = str(int(mm_data.at[i, "Week 2 Probe 2 ID"]))
    wk2_source3_ID = str(int(mm_data.at[i, "Week 2 Probe 3 ID"]))
```

```
47      mes_description = mm_data.at[i, "network"]
48
49      ping = Ping(af=4, target=tgt, description=mes_description, tags=[tag
        ])
50      wk1_source1 = AtlasSource(type="probes", requested=1, value =
        wk1_source1_ID)
51      wk1_source2 = AtlasSource(type="probes", requested=1, value =
        wk1_source2_ID)
52      wk1_source3 = AtlasSource(type="probes", requested=1, value =
        wk1_source3_ID)
53      wk2_source1 = AtlasSource(type="probes", requested=1, value =
        wk2_source1_ID)
54      wk2_source2 = AtlasSource(type="probes", requested=1, value =
        wk2_source2_ID)
55      wk2_source3 = AtlasSource(type="probes", requested=1, value =
        wk2_source3_ID)
56      atlas_request = AtlasCreateRequest(
57        start_time=datetime.utcnow(),
58        key=KEY,
59        measurements=[ping],
60        sources=[wk1_source1, wk1_source2, wk1_source3, wk2_source1,
        wk2_source2, wk2_source3],
61        is_oneoff=True)
62
63      (is_success, response) = atlas_request.create()
64
65      while (not is_success) and (tries <= 60):
66        tries += 1
67        time.sleep(2)
68        atlas_request = AtlasCreateRequest(
69          start_time=datetime.utcnow(),
70          key=KEY,
71          measurements=[ping],
72          sources=[wk1_source1, wk1_source2, wk1_source3, wk2_source1,
        wk2_source2, wk2_source3],
73          is_oneoff=True)
74        (is_success, response) = atlas_request.create()
75
76      if tries >= 59:
77        continue
```

```
78
79      measurement_ID = response['measurements'][0]
80      mm_data.at[i, "Measurement ID"] = measurement_ID
81
82    mm_data.set_index('network', inplace=True)
83
84    return mm_data
85
86  def parallelize_requests(mm_df, func):
87
88    mm_data_size = mm_df.shape[0]
89    print(mm_data_size)
90    n_cores = 20
91    print("Using %d cores\n" % n_cores)
92
93    chunk_size = int(mm_data_size/n_cores)
94    print(chunk_size)
95
96    chunks = [mm_df_init.iloc[mm_df_init.index[i:i + chunk_size]] for i in
        range(0, mm_df_init.shape[0], chunk_size)]
97
98    pool = Pool(n_cores)
99    df = pd.concat(pool.map(func, chunks))
100   pool.close()
101   pool.join()
102
103
104   return df
105
106
107
108 ##########################################################################
109
110 if __name__ == "__main__":
111   if len(sys.argv) < 4:
112     raise RuntimeError("Provide Input file, Output file, and tag")
113
114   start = datetime.now()
115   start_time = start.strftime("%H:%M:%S")
116   print("Start Time =", start_time)
```

```
117
118   mm_file = sys.argv[1]
119   df_outfile = sys.argv[2]
120   tag = sys.argv[3]
121   f = open(df_outfile, 'w', newline='')
122
123   #TESTING ONLY
124
125   # mm_df = pd.read_csv(mm_file)
126   # mm_df_init = init_df_for_measurements(mm_df)
127   # mm_df_init = mm_df.sample(n=20)
128   # mm_df_init.reset_index(inplace=True)
129   # mm_df_init.drop(columns=['index'], inplace=True)
130
131   # print(mm_df_init)
132
133
134   #ACTUAL
135   mm_df = pd.read_csv(mm_file)
136   mm_df_init = init_df_for_measurements(mm_df)
137
138
139   output_df = parallelize_requests(mm_df_init, request_measurements)
140   output_df.to_csv(f, index_label = 'network')
141   f.close()
142
143   end = datetime.now()
144   end_time = end.strftime("%H:%M:%S")
145   print("End Time =", end_time)
```

A.7 Retrieve-Measurement-Results.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Program retrieves mesurement results based on measurement ID.
4
5 ARG1: Maxmind Data File
6 ARG2: Output file
7 """
```

```
8
9  import sys
10 import requests
11 import pandas as pd
12 import numpy as np
13 from ripe.atlas.sagan import PingResult
14 from datetime import datetime
15
16
17 def init_df_for_measurements(meas_df):
18
19   meas_df["Week 1 Probe 1 Min RTT"] = np.nan
20   meas_df["Week 1 Probe 2 Min RTT"] = np.nan
21   meas_df["Week 1 Probe 3 Min RTT"] = np.nan
22   meas_df["Week 2 Probe 1 Min RTT"] = np.nan
23   meas_df["Week 2 Probe 2 Min RTT"] = np.nan
24   meas_df["Week 2 Probe 3 Min RTT"] = np.nan
25
26   return meas_df
27
28 def request_results(meas_df):
29   meas_df_size = meas_df.shape[0]
30
31   for i in range(meas_df_size):
32     measurement = i+1
33     print("Measurement: ", measurement)
34     meas_id = meas_df.at[i, "Measurement ID"]
35
36     if np.isnan(meas_id):
37       continue
38
39     meas_id = int(meas_id)
40
41     source = "https://atlas.ripe.net/api/v2/measurements/" + str(meas_id
       ) + "/results"
42     resp = requests.get(source)
43     while(resp.status_code != 200):
44       resp = requests.get(source)
45     response = requests.get(source).json()
46
```

```
47
48      for element in response:
49        result = PingResult(element)
50        result_probe = result.probe_id
51        wk1_probe_1 = int(meas_df.at[i, "Week 1 Probe 1 ID"])
52        wk1_probe_2 = int(meas_df.at[i, "Week 1 Probe 2 ID"])
53        wk1_probe_3 = int(meas_df.at[i, "Week 1 Probe 3 ID"])
54        wk2_probe_1 = int(meas_df.at[i, "Week 2 Probe 1 ID"])
55        wk2_probe_2 = int(meas_df.at[i, "Week 2 Probe 2 ID"])
56        wk2_probe_3 = int(meas_df.at[i, "Week 2 Probe 3 ID"])
57        min_rtt = result.rtt_min
58        if result_probe == wk1_probe_1:
59          meas_df.at[i, "Week 1 Probe 1 Min RTT"] = min_rtt
60        elif result_probe == wk1_probe_2:
61          meas_df.at[i, "Week 1 Probe 2 Min RTT"] = min_rtt
62        elif result_probe == wk1_probe_3:
63          meas_df.at[i, "Week 1 Probe 3 Min RTT"] = min_rtt
64        elif result_probe == wk2_probe_1:
65          meas_df.at[i, "Week 2 Probe 1 Min RTT"] = min_rtt
66        elif result_probe == wk2_probe_2:
67          meas_df.at[i, "Week 2 Probe 2 Min RTT"] = min_rtt
68        elif result_probe == wk2_probe_3:
69          meas_df.at[i, "Week 2 Probe 3 Min RTT"] = min_rtt
70
71    meas_df.set_index('network', inplace=True)
72
73    print("Results Retrieved")
74
75    return meas_df
76
77
78
79
80  ######################################################################
81
82  if __name__ == "__main__":
83    if len(sys.argv) < 3:
84      raise RuntimeError("Provide Input file and Output file")
85
86    start = datetime.now()
```

```
  start_time = start.strftime("%H:%M:%S")
  print("Start Time =", start_time)

  mm_file = sys.argv[1]
  outfile = sys.argv[2]
  f = open(outfile, 'w', newline='')

  mm_df = pd.read_csv(mm_file)

  init_df = init_df_for_measurements(mm_df)

  output_df = request_results(init_df)

  output_df.to_csv(f, index_label = 'network')
  f.close()

  end = datetime.now()
  end_time = end.strftime("%H:%M:%S")
  print("End Time =", end_time)
```

A.8 Retrieve-Measurement-Results-by-tag.py

```
# -*- coding: utf-8 -*-
"""
Program retrieves mesurement results based on measurement ID using the
    assigned Tag.

ARG1: Maxmind Data File
ARG2: Output file
"""

import sys
import requests
import pandas as pd
import numpy as np
from ripe.atlas.sagan import PingResult
from datetime import datetime
import urllib3
import json
```

```
17
18 API_URL = "https://atlas.ripe.net/api/v2"
19 TAG = "sixaprilactual"
20 KEY = "af5f9084-9e02-42db-a726-cad579996607"
21 HTTP = urllib3.PoolManager()
22
23 def init_df_for_measurements(meas_df):
24
25   meas_df["Week 1 Probe 1 Min RTT"] = np.nan
26   meas_df["Week 1 Probe 2 Min RTT"] = np.nan
27   meas_df["Week 1 Probe 3 Min RTT"] = np.nan
28   meas_df["Week 2 Probe 1 Min RTT"] = np.nan
29   meas_df["Week 2 Probe 2 Min RTT"] = np.nan
30   meas_df["Week 2 Probe 3 Min RTT"] = np.nan
31
32   meas_df.set_index('network', inplace=True)
33
34   return meas_df
35
36 def getPage(page):
37   payload = {'key' : KEY, 'tags' : TAG, 'page' : page}
38   res = HTTP.request('GET', API_URL + "/measurements" , fields = payload
      )
39   tries = 1
40   while (res.status != 200) and (tries <= 20):
41     tries += 1
42     res = HTTP.request('GET', API_URL + "/measurements" , fields =
     payload)
43   if (res.status != 200) and (tries > 20):
44     print("page error on page %d", page)
45     return None
46   return json.loads(res.data.decode('utf-8'))
47
48 def getUrl(url):
49   tries = 1
50   resp = HTTP.request('GET', url)
51   while(resp.status != 200) and (tries <= 20):
52     tries += 1
53     resp = HTTP.request('GET', url)
54   if (resp.status != 200) and (tries > 20):
```

```
55      print("could not retrieve url %s", url)
56      return None
57    return json.loads(resp.data.decode('utf-8'))
58
59  def request_results(meas_df):
60    measurement = 0
61    page = 1
62
63    next_url = API_URL + "/measurements" + "?key=" + KEY + "&page=" + str(
       page) + "&tags=" + TAG
64    while True:
65
66      if not next_url:
67        break
68
69      tag_response = getUrl(next_url)
70
71      if not tag_response:
72        break
73
74      try:
75        tag_results = tag_response["results"]
76      except:
77        break
78
79      next_url = tag_response["next"]
80
81      for tag_element in tag_results:
82        measurement += 1
83        print("Measurement: ", measurement, end = '\r')
84
85        result_html = tag_element["result"]
86        meas_id = tag_element["id"]
87        description = tag_element["description"]
88        meas_df.at[description, "Measurement ID"] = meas_id
89
90        resp = HTTP.request('GET', result_html)
91        while(resp.status != 200):
92          resp = HTTP.request('GET', result_html)
93        response = json.loads(resp.data.decode('utf-8'))
```

```
94
95      for element in response:
96        result = PingResult(element)
97        result_probe = result.probe_id
98        wk1_probe_1 = int(meas_df.at[description, "Week 1 Probe 1 ID"])
99        wk1_probe_2 = int(meas_df.at[description, "Week 1 Probe 2 ID"])
100       wk1_probe_3 = int(meas_df.at[description, "Week 1 Probe 3 ID"])
101       wk2_probe_1 = int(meas_df.at[description, "Week 2 Probe 1 ID"])
102       wk2_probe_2 = int(meas_df.at[description, "Week 2 Probe 2 ID"])
103       wk2_probe_3 = int(meas_df.at[description, "Week 2 Probe 3 ID"])
104       min_rtt = result.rtt_min
105       if result_probe == wk1_probe_1:
106         meas_df.at[description, "Week 1 Probe 1 Min RTT"] = min_rtt
107       elif result_probe == wk1_probe_2:
108         meas_df.at[description, "Week 1 Probe 2 Min RTT"] = min_rtt
109       elif result_probe == wk1_probe_3:
110         meas_df.at[description, "Week 1 Probe 3 Min RTT"] = min_rtt
111       elif result_probe == wk2_probe_1:
112         meas_df.at[description, "Week 2 Probe 1 Min RTT"] = min_rtt
113       elif result_probe == wk2_probe_2:
114         meas_df.at[description, "Week 2 Probe 2 Min RTT"] = min_rtt
115       elif result_probe == wk2_probe_3:
116         meas_df.at[description, "Week 2 Probe 3 Min RTT"] = min_rtt
117     page += 1
118
119   print("Measurement: ", measurement)
120   print("Results Retrieved")
121
122   return meas_df
123
124
125
126
127 ################################################################
128
129 if __name__ == "__main__":
130   if len(sys.argv) < 3:
131     raise RuntimeError("Provide Input file and Output file")
132
133   start = datetime.now()
```

```
  start_time = start.strftime("%H:%M:%S")
  print("Start Time =", start_time)

  mm_file = sys.argv[1]
  outfile = sys.argv[2]
  f = open(outfile, 'w', newline='')

  mm_df = pd.read_csv(mm_file)

  init_df = init_df_for_measurements(mm_df)

  output_df = request_results(init_df)

  output_df.to_csv(f, index_label = 'network')
  f.close()

  end = datetime.now()
  end_time = end.strftime("%H:%M:%S")
  print("End Time =", end_time)
```

A.9 Retrieve-Measurement-Results-NO-MEAS-FILE.py

```
# -*- coding: utf-8 -*-
"""
Program retrieves mesurement results based on measurement ID using the
    assigned Tag.

ARG1: Maxmind Data File
ARG2: Output file
"""

import sys
import requests
import pandas as pd
import numpy as np
from ripe.atlas.sagan import PingResult
from datetime import datetime
import urllib3
import json
```

```
17
18 API_URL = "https://atlas.ripe.net/api/v2"
19 TAG = "onethreeaprilactualv2"
20 KEY = "af5f9084-9e02-42db-a726-cad579996607"
21 HTTP = urllib3.PoolManager()
22
23 def init_df_for_measurements(meas_df):
24
25   mm_reach_only = meas_df[meas_df["Target Reachable"] == True]
26
27   mm_reach_only["Week 1 Probe 1 Min RTT"] = np.nan
28   mm_reach_only["Week 1 Probe 2 Min RTT"] = np.nan
29   mm_reach_only["Week 1 Probe 3 Min RTT"] = np.nan
30   mm_reach_only["Week 2 Probe 1 Min RTT"] = np.nan
31   mm_reach_only["Week 2 Probe 2 Min RTT"] = np.nan
32   mm_reach_only["Week 2 Probe 3 Min RTT"] = np.nan
33
34   mm_reach_only["Measurement ID"] = np.nan
35
36   mm_reach_only.set_index('network', inplace=True)
37
38   return mm_reach_only
39
40 def getPage(page):
41   payload = {'key' : KEY, 'tags' : TAG, 'page' : page}
42   res = HTTP.request('GET', API_URL + "/measurements" , fields = payload
      )
43   tries = 1
44   while (res.status != 200) and (tries <= 20):
45     tries += 1
46     res = HTTP.request('GET', API_URL + "/measurements" , fields =
     payload)
47   if (res.status != 200) and (tries > 20):
48     print("page error on page %d", page)
49     return None
50   return json.loads(res.data.decode('utf-8'))
51
52 def getUrl(url):
53   tries = 1
54   resp = HTTP.request('GET', url)
```

```
  while(resp.status != 200) and (tries <= 20):
    tries += 1
    resp = HTTP.request('GET', url)
  if (resp.status != 200) and (tries > 20):
    print("could not retrieve url %s", url)
    return None
  return json.loads(resp.data.decode('utf-8'))

def request_results(meas_df):
  measurement = 0
  page = 1

  next_url = API_URL + "/measurements" + "?key=" + KEY + "&page=" + str(
   page) + "&tags=" + TAG
  while True:

    if not next_url:
      break

    tag_response = getUrl(next_url)

    if not tag_response:
      break

    try:
      tag_results = tag_response["results"]
    except:
      break

    next_url = tag_response["next"]

    for tag_element in tag_results:
      measurement += 1
      print("Measurement: ", measurement, end = '\r')

      result_html = tag_element["result"]
      meas_id = tag_element["id"]
      description = tag_element["description"]
      meas_df.at[description, "Measurement ID"] = meas_id

```

```
94        resp = HTTP.request('GET', result_html)
95        while(resp.status != 200):
96          resp = HTTP.request('GET', result_html)
97        response = json.loads(resp.data.decode('utf-8'))
98
99        for element in response:
100         result = PingResult(element)
101         result_probe = result.probe_id
102         wk1_probe_1 = int(meas_df.at[description, "Week 1 Probe 1 ID"])
103         wk1_probe_2 = int(meas_df.at[description, "Week 1 Probe 2 ID"])
104         wk1_probe_3 = int(meas_df.at[description, "Week 1 Probe 3 ID"])
105         wk2_probe_1 = int(meas_df.at[description, "Week 2 Probe 1 ID"])
106         wk2_probe_2 = int(meas_df.at[description, "Week 2 Probe 2 ID"])
107         wk2_probe_3 = int(meas_df.at[description, "Week 2 Probe 3 ID"])
108         min_rtt = result.rtt_min
109         if result_probe == wk1_probe_1:
110           meas_df.at[description, "Week 1 Probe 1 Min RTT"] = min_rtt
111         elif result_probe == wk1_probe_2:
112           meas_df.at[description, "Week 1 Probe 2 Min RTT"] = min_rtt
113         elif result_probe == wk1_probe_3:
114           meas_df.at[description, "Week 1 Probe 3 Min RTT"] = min_rtt
115         elif result_probe == wk2_probe_1:
116           meas_df.at[description, "Week 2 Probe 1 Min RTT"] = min_rtt
117         elif result_probe == wk2_probe_2:
118           meas_df.at[description, "Week 2 Probe 2 Min RTT"] = min_rtt
119         elif result_probe == wk2_probe_3:
120           meas_df.at[description, "Week 2 Probe 3 Min RTT"] = min_rtt
121     page += 1
122
123   print("Measurement: ", measurement)
124   print("Results Retrieved")
125
126   return meas_df
127
128
129
130
131 ############################################################
132
133 if __name__ == "__main__":
```

```
134  if len(sys.argv) < 3:
135    raise RuntimeError("Provide Input file and Output file")
136
137  start = datetime.now()
138  start_time = start.strftime("%H:%M:%S")
139  print("Start Time =", start_time)
140
141  mm_file = sys.argv[1]
142  outfile = sys.argv[2]
143  f = open(outfile, 'w', newline='')
144
145  mm_df = pd.read_csv(mm_file)
146
147  init_df = init_df_for_measurements(mm_df)
148
149  output_df = request_results(init_df)
150
151  output_df.to_csv(f, index_label = 'network')
152  f.close()
153
154  end = datetime.now()
155  end_time = end.strftime("%H:%M:%S")
156  print("End Time =", end_time)
```

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] B. Huffaker, M. Fomenkov, and K. Claffy, "Geocompare: A comparison of public and commercial geolocation databases - Technical report," Cooperative Association for Internet Data Analysis (CAIDA), Tech. Rep., 2011-05.

[2] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: Unreliable?" *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 53–56, Apr. 2011. Available: https://dl.acm.org/doi/10.1145/1971162.1971171

[3] J. A. Culbert, "Toward understanding the longitudinal stability of an IP geolocation database," M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, Mar. 2020. Available: https://calhoun.nps.edu/handle/10945/64848

[4] O. Solon, "Kansas family sues mapping company for years of 'digital hell.'" Accessed Aug. 9, 2021 [Online], Aug. 2016. Available: http://www.theguardian.com/technology/2016/aug/09/maxmind-mapping-lawsuit-kansas-farm-ip-address

[5] MaxMind, "GeoIP® databases & services: Industry leading IP intelligence." Accessed Aug. 9, 2021 [Online]. Available: https://www.maxmind.com/en/geoip2-services-and-databases

[6] M. Gouel, K. Vermeulen, O. Fourmaux, T. Friedman, and R. Beverly, "IP geolocation database stability and implications for network research," in *Proceedings of the Network Traffic Measurement and Analysis (TMA) Conference*, Sep. 2021.

[7] E. Kline, K. Duleba, Z. Szamonek, S. Moser, and W. Kumari, "A format for self-published IP geolocation feeds," RFC Editor, Tech. Rep. RFC8805, Aug. 2020. Available: https://www.rfc-editor.org/info/rfc8805

[8] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1219–1232, Dec. 2006. Available: http://ieeexplore.ieee.org/document/4032725/

[9] J. F. Kurose and K. W. Ross, *Computer networking: A top-down approach*, 7th ed. Boston, MA: Pearson, 2017.

[10] Jithin, "WHOIS lookup explained." Accessed Aug. 9, 2021 [Online]. Available: https://www.interserver.net/tips/kb/whois-lookup-explained/

[11] L. Daigle, "WHOIS protocol specification," RFC Editor, Tech. Rep. RFC3912, Sep. 2004. Available: https://www.rfc-editor.org/info/rfc3912

[12] IANA, "Root zone database." Accessed Aug. 9, 2021 [Online]. Available: https://www.iana.org/domains/root/db

[13] J. Postel, "Domain Name System structure and delegation," RFC Editor, Tech. Rep. RFC1591, Mar. 1994. Available: https://www.rfc-editor.org/info/rfc1591

[14] ICANN, "Top-level domains (gTLDs)." Accessed Aug. 9, 2021 [Online]. Available: http://archive.icann.org/en/tlds/

[15] B. Huffaker, M. Fomenkov, and K. Claffy, "DRoP: DNS-based router positioning," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 5–13, July 2014. Available: https://dl.acm.org/doi/10.1145/2656877.2656879

[16] CAIDA, "CAIDA data - Overview of datasets, monitors, and reports." Accessed Aug. 9, 2021 [Online]. Available: https://www.caida.org/catalog/datasets/overview/

[17] IPinfo.io, "154.24.41.2 IP address details." Accessed Aug. 9, 2021 [Online]. Available: https://ipinfo.io/154.24.41.2

[18] Cloudflare, "What is anycast?" Accessed Aug. 9, 2021 [Online]. Available: https://www.cloudflare.com/learning/cdn/glossary/anycast-network/

[19] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, "A look at router geolocation in public and commercial databases," in *Proceedings of the 2017 Internet Measurement Conference*. London United Kingdom: ACM, Nov. 2017, pp. 463–469. Available: https://dl.acm.org/doi/10.1145/3131365.3131380

[20] MaxMind, "MaxMind end user license agreement." Accessed Aug. 9, 2021 [Online]. Available: https://www.maxmind.com/en/end-user-license-agreement

[21] RIPE NCC, "What we do." Accessed Aug. 9, 2021 [Online]. Available: https://www.ripe.net/about-us/what-we-do/functions

[22] RIPE NCC, "Global RIPE Atlas network coverage." Accessed Aug. 9, 2021 [Online]. Available: https://atlas.ripe.net/results/maps/network-coverage/

[23] RIPE NCC, "RIPE Atlas Cousteau — RIPE Atlas Cousteau 1.1 documentation." Accessed Aug. 9, 2021 [Online]. Available: https://ripe-atlas-cousteau.readthedocs.io/en/latest/

[24] RIPE NCC, "Welcome to RIPE Atlas Sagan's documentation! — RIPE Atlas Sagan 1.2 documentation," Accessed Aug. 9, 2021 [Online]. Available: https://ripe-atlas-sagan.readthedocs.io/en/latest/

[25] RIPE NCC, "Atlas console." Accessed Aug. 9, 2021 [Online]. Available: https://atlas.ripe.net/about/faq

[26] RIPE Atlas, “Starting your own measurements (user-defined measurements). Accessed Aug. 9, 2021 [Online]. Available: https://beta-docs.atlas.ripe.net/getting-started/user-defined-measurements.html

[27] ANT Lab, “IPv4 hitlists.” Accessed Aug. 9, 2021 [Online]. Available: https://ant.isi.edu/datasets/ip_hitlists/

[28] iboss, “Cloud Data Centers - Global SASE Fabric.” Accessed Aug. 17, 2021 [Online]. Available: https://www.iboss.com/cloud-data-centers/

[29] Viasat, “Home satellite Internet.” Accessed Aug. 9, 2021 [Online]. Available: https: //www.viasat.com/home-internet/

[30] J. Fan, A. Tao, and C. Lv, “The coupling mechanism of the centroids of economic gravity and population gravity and its effect on the regional gap in China,” *Progress In Grography*, vol. 29, no. 1, pp. 87–95, Jan. 2010. Available: http://sourcedb.igsnrr.cas.cn/zw/lw/201007/P020100706529106697457.pdf

[31] Fping.org, “Fping man-page.” Accessed Aug. 16, 2021 [Online]. Available: https: // fping.org/fping.1.html

[32] Z. Durumeric, “Zmap Wiki.” Accessed Aug. 16, 2021 [Online]. Available: https: // github.com/zmap/zmap

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California