國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

對Twitch攝取服務器進行地理定位

Geolocalizing Twitch Ingest Servers

李昕耘

Hsin-Yun Lee

指導教授 : 黃寶儀 博士

Advisor: Polly Huang, Ph.D.

中華民國 111年12月

December, 2022

**Acknowledgement**

First, I am most grateful to my advisor, Polly Huang. She has always been very careful and patient in discussing the research content with me, and often pointed out the key points of the research. And she also taught me how to become a qualified graduate student, which makes me grow, and makes me have more expectations for the future. So, I am lucky that she is my teacher.

In addition, I am also grateful to the seniors, classmates, and juniors of NSLAB for their helpful opinions in my research, which lets me find more inspiration on the experiments and thesis.

Finally, I would like to thank my dear parents and friends. Their long-term support and love allow me to overcome one difficulty after another. And I will continue to move forward with everyone's encouragement.

摘要

隨著網際網路的普及，對網路上的主機進行地理定位顯得越來越重要，尤其是對於許多位置感知服務。過去有一些不同的測量方法，例如評估現有數據庫、基於DNS的定位方法、以及基於測量的方法。本論文嘗試使用基於測量的方法，用82個VPN作為輔助，對60個Twitch攝取服務器進行地理定位，而不使用PlanetLab節點或其他數據集。由於NordVPN提供的VPN操作的便利性和靈活性，例如可以通過簡單的命令從主機連接到VPN，接著將VPN當作基站來以探測其他地標或目標，或者簡單地將VPN視為地標進行探測。最後，我們評估以下地理定位方法Shortest Ping、GeoPing、CBG (基於約束的地理定位)，以及其他混合不同地理定位方法的方法，最後觀察何種地理定位方法在使用VPN作為輔助時會達到最佳性能。

關鍵字: 網路測量、Twitch、定位

**Abstract**

With the popularity of the Internet, Geolocalizing Internet hosts is more and more important for many location-aware services. There are some different measurement methods in the past such as checking of existing databases, DNS-based approach, and Measurement-based approach. This thesis tries to use measurement-based approaches to geolocalizing 60 Twitch ingest servers with 82 VPNs as an auxiliary instead of using PlanetLab nodes or other datasets. Owing to the convenience and flexibility of VPN operation provided by NordVPN, such as connecting to a VPN server from host with simple commands and then use the VPN server as a base site to probe other landmarks (LMs) or targets (Ts), or simply treat the VPN server as a LM to probe.

Finally, we evaluate the accuracy of the following geolocation methods Shortest Ping, GeoPing, CBG (Constraint-Based Geolocation), and some other Approaches that mix different geolocation methods, and finally analyze which geolocation method will achieve the best performance when using the VPN as an aid.

**Keywords:** Network Measurement, Twitch, Localization

**Contents**

**Acknowledgement** **I**

摘要 **II**

**Abstract** **III**

1 **Introduction** **1**

2 **Related Work** **6**

2.1 Prior Research from Hsi Chen 6

2.2 Check existing databases 6

2.3 Reverse DNS 7

2.4 Measurement-based geolocation 8

2.4.1 GeoPing 9

2.4.2 Shortest Ping 10

2.4.3 Constraint-Based Geolocation (CBG) 10

2.4.4 Topology-based Geolocation (TBG) 11

3 **Our Approach** **12**

4 **Measurement Methodology** **15**

4.1 Experiment Design 15

4.1.1 Establish a VPN List 15

4.1.2 NordVPN Selection 18

4.1.3 Exploit VPN to Implement These Geolocation Methods 19

4.1.3.1 Shortest ping 19

4.1.3.2 GeoPing 19

4.1.3.3 CBG 20

4.1.3.4 Shortest Distance 23

4.1.4 Three necessary RTTs in These Geolocation Methods 23

4.1.4.1 RTT1 between NSLAB and VPN 23

4.1.4.2 RTT2 between VPN and Twitch Ingest Server 23

4.1.4.3 RTT3 between VPNs 24

4.2 Packet Collection and RTT Measurement 27

4.2.1 Get RTT between NSLAB and VPN 27

4.2.1.1 Packet Collection 27

4.2.1.2 RTT Computation 28

4.2.2 Get RTT between VPN and Twitch Ingest Server 28

4.2.2.1 Packet Collection 28

4.2.2.2 RTT Computation 35

4.2.3 Get RTT between VPNs 36

4.2.3.1 Packet Collection 36

4.2.3.2 RTT Computation 37

5 **Measurement Examination** **39**

5.1 RTT1 between NSLAB and VPN 39

5.2 RTT2 between VPN and Twitch Ingest Server 40

5.2.1 RTT between NSLAB across VPN and Twitch Ingest Server 40

5.2.2 Unadjusted RTT2 between VPN and Twitch Ingest Server 41

5.3 RTT3 between VPNs 45

6 **Evaluate Experimental Result** **46**

6.1 Evaluation Methodology 46

6.2 Evaluation Parameters in Geolocation Methods 49

6.2.1 Shortest Ping 50

6.2.2 GeoPing 51

6.2.3 CBG 52

6.2.4 Shortest Distance 53

6.3 Evaluation Result 56

6.3.1 Evaluation on Rank Difference 56

6.3.2 Evaluation on Error Distance 59

6.3.3 Analysis of Evaluation Result 62

7 **Evaluate CBG - Shortest Distance Method** **64**

7.1 Introduction of CBG - Shortest Distance Method 64

7.2 Evaluation Parameters in CBG-Shortest Distance Method 67

7.3 Evaluation Result of CBG - Shortest Distance method 72

8 **Conclusion** **76**

**Reference** **83**

**Chapter 1**

**Introduction**

Geolocation is widely used today, and many companies use geolocation technology to restrict content to specific countries or regions. For example, many audio and video playbacks are often only available in specific countries or regions, they geolocalize the user's IP and filter out those users who do not meet the conditions. Besides, advertisers can provide different advertisements according to the people in different countries or regions. And this mechanism can be combined with the user's past search history to recommend ads. In addition, researchers can use geolocation to observe different online behaviors of people in different countries, such as the impact of different cultural backgrounds on online dating, or the crime rate under different national or different local policies. Moreover, some large network platforms often use CDN Server to help them distribute content. However, for users who want to receive the content, they want to get the content with short response time. If the system of the platforms can geolocalize the users, they can provide the content requested by the users in a CDN server that is near to the users with shorter delay time. Furthermore, Information security members sometimes can track the footprints of network attack by analyzing the abnormal traffic from an IP address and geolocalizing this IP address. Then they can impose some restrictions or block this IP

After knowing the importance of geolocalization, there are existing location methods in prior papers such as checking existing databases [1, 2], reverse DNS [3], or geolocation-based measurements. Here we specialize in measurement-based geolocation because it provides better coverage and accuracy. Instead of providing a new geolocation method, this thesis focusses on comparing these three measurement-based approaches "Shortest ping [4]", "GeoPing [5]", "Constraint-Based Geolocation (CBG) [6]", and some other Approaches that mix different geolocation methods.

Because most of the measurement-based algorithms need reference points to geolocalize targets (Ts), and there are two kinds of reference points: Vantage points (VPs) and landmarks (LMs). Figure 1 shows the relationship and properties of VPs, LMs, and Ts. Vantage points (VPs) have the ability to probe the landmarks (LMs) or targets (Ts) actively and they must be distributed in multiple locations to provide different perspectives to detect the locations of the LMs and Ts. On the other hand, landmarks (LMs) are some known locations that can passively be probed by VPs, and LMs must also be scattered in multiple places for reference. Thus, it is necessary to identify computers on the Internet to serve as VPs and LMs. However, most of previous studies used PlanetLab nodes with known locations at different universities as their VPs and LMs, which are large in number and distributed around the world. Such resources allow their geolocation methods to yield good coverage and accuracy.

Unfortunately, PlanetLab was officially shut down May 2020 [7]. We lost access to the machines that serve the role of VPs and LMs for the algorithms we study. In order to meet these two necessary needs for these geolocation methods, we seek the service of VPN. NordVPN [8] are distributed all over the world, and each VPN server provides its latitude and longitude as the geolocation. So, we can regard each VPN server has a known location. In addition, we can probe other LMs or Ts actively after connecting to a VPN server. Hence VPN can be treated as VP. Furthermore, NordVPN can be treated as LMs as well since NordVPN can be probe simply by ping and it can respond to requests passively.

As for the selection of T for evaluating these geolocation methods. We have to talk about Twitch first. Twitch is a large-scale live streaming platform. It accounts for a large proportion in all platforms in the world. On the other hand, Twitch is the leader in game streaming. In addition to accumulating a large number of viewers, the mechanism that viewers can interact with streamer in chat room is also mature. Twitch provides contribution list, music on demand, attaching instant messages with donation, and other functions. These functions help streamers to continuously create interaction on Twitch. As for the Twitch live video delivery architecture shown in Figure 2. For streamers, Twitch provide some servers called ingest servers where live videos can be uploaded by mobile iPhone, Pad, laptop, desktop etc. And then these live videos will be transcoded to different bitrate videos. After that, these videos will be distributed to multiple CDN servers, and viewers are directed to one of these servers to get the video. According to statistics, there are about 60 Twitch ingest servers scattered all over the world, and each server has several machines (about 3~15 machines). These machines in the same ingest servers has different IP address in the same subnet. Above all, Twitch provides the geolocation of each ingest servers as well, such as the country and city where these ingest servers are located. Therefore, the geolocation of Twitch ingest servers can be treated as ground truth locations and these servers can be used as Ts for known locations while evaluating these geolocation methods.

After getting reference points and targets, we use the two elements to do packet collection and get the three necessary RTTs: RTT between NSLAB and VPN, RTT between VPN and Twitch Ingest Server, and RTT between VPNs (Section 4 and 5). After that, we subtract the RTT between NSLAB and VPN from the RTT from NSLAB via VPN to Twitch Ingest Server. This gives us the RTT from the VPN to Twitch Ingest Server which is essentially the delay from the VP to T. The RTT between VPNs is essentially the delay from the VP to LM. With the VP-to-T and VP-to-LM delay measured, we are able to implement and compare the measurement-based geolocation methods as in Figure 1 (Section 6).

According to the experimental results, CBG and Shortest Distance (4.1.3.4) perform the best and GeoPing perform the worst among these geolocation methods. Note that Shortest Distance method is derived during the implementation of CBG. It transforms the RTT between VPs and T to the estimated distance between VPs and T. And then simply map the T to the VP with the shortest estimated distance from T. Therefore, we further combine CBG and Shortest Distance to form a hybrid method called “CBG-Shortest Distance” encompass the strength of the two methods performing the best. Surprisingly, we discover that the hybrid method outperforms than CBG and Shortest Distance.

Our first contribution is to provide a new tool – VPN for measurement-based geolocation methods since the locations of VPN servers are known and widely distributed, and it can be probed simply by ping. Our second contribution is to compare the performance by using VPN as a tool to geolocalize twitch ingest servers on various geolocation methods. Our third contribution is to provide a hybrid method by combining two different geolocation methods, and show that hybrid method may have better performance.

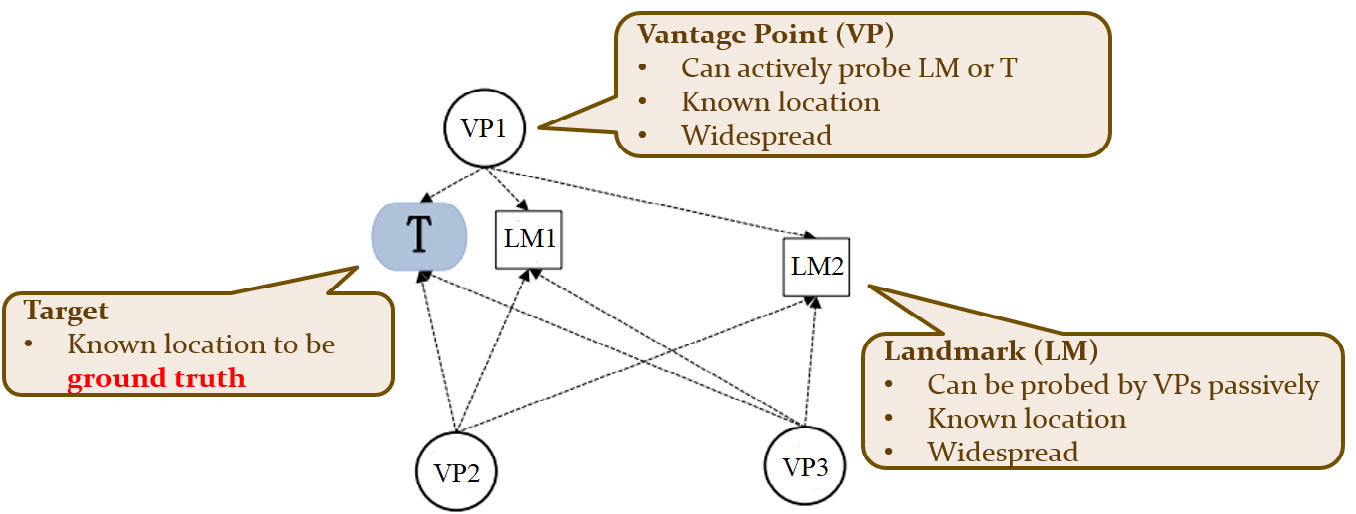


Figure 1. Relationship between VPs, LMs, and Ts

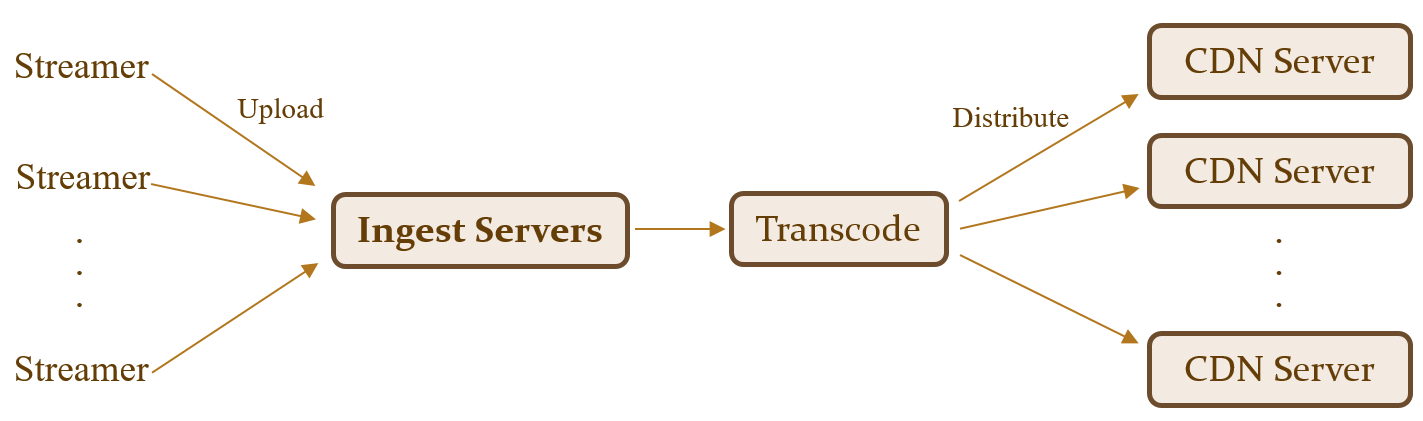


Figure 2. Twitch live video delivery architecture

**Chapter 2**

**Related Work**

There are some geolocation methods such as checking existing database [1, 2], reverse DNS [3], and measurement-based geolocation methods [4, 5, 6, 11]. And the comparison of these geolocation methods is shown in Table 1.

**2.1 Prior Research from Hsi Chen**

Referring to prior research [9] from Hsi Chen, one of our lab members in NSLAB. She did a preliminary analysis of various geolocation methods. And she tried to use the measurement-based geolocation methods to geolocalize Twitch's servers. It inspired my research on using VPNs as reference points and Twitch servers as the Ts. Since this prior research faced the challenge of calculating the delay time between VPN and Twitch server. Thus, I will solve this problem in this thesis and apply the collected delay time to implement the different measurement-based geolocation methods for performance evaluation.

**2.2 Check Existing Databases**

First, about checking existing databases, including HostIP.info, Spotter, GeoBytes, IP2Location, IPligence, MaxMind, and NetAcuity. Data in the database was generated in the past by various geolocation methods. And we can check correspondence and accuracy between ip and geolocation of databases to determine whether this method is suitable for geolocation.

However, the method has some flaws. Some [IP, Geolocation] pairs may not appear in each database. For example, database A has [IPA, GeolocationA] but no [IPB, GeolocationB], and database B has [IPB, GeolocationB] but no [IPA, GeolocationA]. In addition, the same IP in different databases may have different geolocation, causing inconsistencies between databases. Take IP 151.101.194.167 for example, GeoByte locates it at latitude and longitude [40.748, -73.984], IP2Location locates it at [37.775, -122.395], MaxMind locates it at [37.751, -97.822], and HostIP.info locates it at United States, but there is no latitude and longitude.

Therefore, this geolocation method is hard to adopt. Additionally, these databases are often outdated due to the difficulty of updating the geolocation of each IP in each database.

**2.3 Reverse DNS**

Another related geolocation method is reverse DNS. It uses PTR record which stores the domain name corresponding to an IP to find the geolocation hint of an IP address. PTR record is also known as Pointer Record, which is one of DNS resource record. It can resolve an IP address to a domain/hostname. For example, in Figure 3, the reverse DNS starts from the IP address *45.113.130.250*, and uses the PTR record to resolve the IP address to the domain/hostname *video-edge-6ab7ae.tpe03.justin.tv*. After that we can observe that whether the domain/hostname has the geolocation hint, including the airport code or city name. In this case we can take the city name “*tpe”* as the geolocation hint.

However, although this method is very convenient, but the PTR record is not forced to add. So not every IP address can be resolved to domain/hostname. In total IPv4 space, there are only one-third of IP addresses have valid reverse DNS hostnames. And only a small part of these hostnames has geolocation hints. Moreover, if there are multiple hint in the domain/hostname, it will be more difficult to determine the geolocation. For example, it’s hard to determine if ***sur****01.****tacoma****.wa.****seattle****.comcast.net* is located in *Tacoma*, *Seattle*, or maybe even ***Su****mne****r***, WA. Furthermore, even unambiguous city names may become ambiguous when they are abbreviated. For example, *nwmd* can stand for ***N****e****w******M****arylan****d****, NB*, or to***N****e****w*** *Rich****m****on****d****, WI*, or to neither of them. The ambiguous hints require more sophisticated approaches to classify as more precise geolocation hints, which makes overall geolocation more difficult.

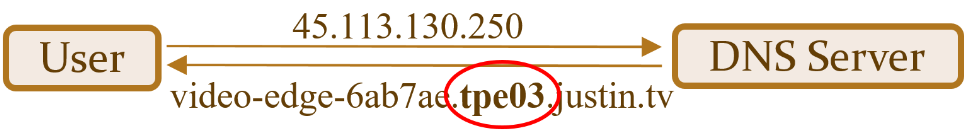


Figure 3. Geolocation hint provided by reverse DNS

**2.4 Measurement-based Geolocation**

Measurement-based geolocation methods focus on the measurement of network latency to estimate the geolocation of a machine on the Internet, but the traditional concept is that there is a weak correlation between network delay and geographic distance [10]. Nevertheless, in recent years, bandwidth and coverage of Internet grow rapidly. The capacity of high-speed links and the number of ISPs are increasing quickly as well (especially in well-connected regions such as North America and Europe). Thus it results in more complete connectivity on the Internet, which implies that there are less circuitous routes and more direct and shorter routes [5]. As a result, the correlation between network latency and geographic distance increases, so does the feasibility of using measurement-based geolocation methods. Therefore, the focus of this thesis is on evaluating measurement-based geolocation by measuring network latency from VPs to Ts.

**2.4.1 GeoPing**

GeoPing [5] based on the concept that hosts have similar network latency to other fixed hosts is likely to close to each other. They use 400 PlanetLab nodes as their VPs and use 25 universities locations as their Ts. GeoPing measure the delay time from each VP to the T to form a T fingerprint. In addition, each VP will also probe other LMs to establish multiple LM fingerprints shown in Figure 4. Calculate the Euclidean distance between the T fingerprint and each LM fingerprint. Finally place the T at the best matching LM. For the example in Figure 4, the fingerprint of LM 1 is more similar to the fingerprint of the T (smallest Euclidean distance between LM 1 fingerprint and T fingerprint), so the location of the T will be mapped to LM 1.

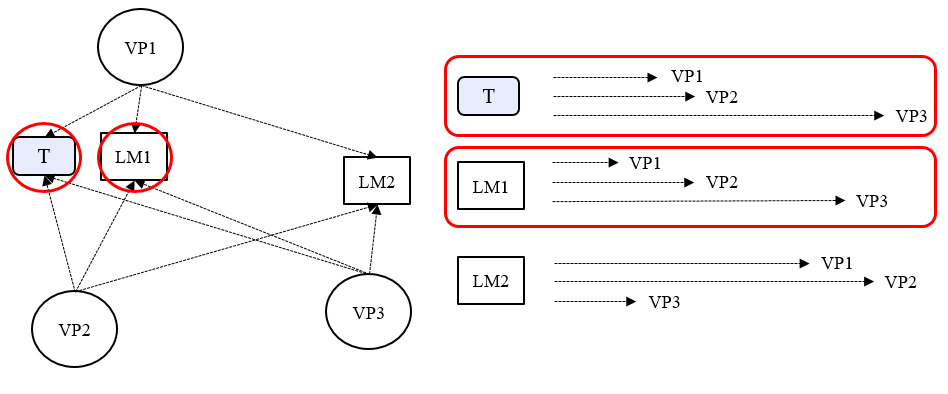


Figure 4. VP, T, and LM in GeoPing (left), with RTT vectors (fingerprints) for GeoPing (right)

**2.4.2 Shortest Ping**

More simply, researchers can also use shortest ping [4] to approximate GeoPing because shortest ping is a simplification of GeoPing. Shortest ping just needs to ping a T from the VPs which are near the T. After that, they select the one with the smallest RTT from 10 pings as the representative of this RTT between T and VPs, and then simply map T to the VP with the minimal RTT. For the example in Figure 5, The RTT between VP 1 and the T is minimum, so the location of T will be mapped to VP 1. Moreover, the researchers found that using just the VPs nearby the T achieves a similar accuracy to using all the VPs.

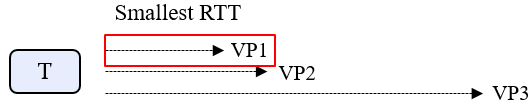


Figure 5. Map T to the VP with smallest RTT

**2.4.3 Constraint-Based Geolocation (CBG)**

CBG (Constraint-Based Geolocation) [6] uses multilateration. The idea is to draw a circle with each VP as the center and the estimated (bestline) distance to the T as the radius. The T is likely located in the overlap of all circles. They use 42 hosts located in Western Europe from RIPE data as the VPs and Ts and they also use 95 hosts located in continental United State from NLANR AMP dataset as the VPs and Ts. In addition, CBG points out that delay can be transformed to geographic distance constraints which can be used in multilateration. This potentially leads to more accurate location estimates of Internet hosts than just using delay for geolocalization. Moreover, this method establishes a dynamic relationship between network delay and geographic distance by periodically calculating the delay between LMs, resulting in a dynamic relationship between IP addresses and geolocations, allowing CBG to adapt itself to time-varying network conditions.

**2.4.4 Topology-based Geolocation (TBG)**

TBG (Topology-based Geolocation) [11] can be seen as an extension of CBG, more than CBG in that the topology of the servers is also taken into account. TBG learns the topology by simultaneously geolocalizing the Ts as well as routers encountered on paths from the LMs to other LMs or Ts. And it will map the network topology. And TBG claims to achieve a higher accuracy than CBG. However, it is hard to implement TBG on the current Internet, since sometimes routers are unreachable by traceroute or ping, because ICMP packets may be lost, filtered, or disabled by routers [12].

Consequently, this thesis will implement three of these measurement-based geolocation methods: Shortest Ping, GeoPing, and CBG. And evaluate the performance of these three geolocation methods with VPN servers as the VPs and LMs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Dynamic | Ping | Traceroute | Ref. points |
| DB | X | X | X | X |
| RDNS | X | X | X | X |
| GeoPing | O | O | X | O |
| Shortest Ping | O | O | X | O |
| CBG | O | O | X | O |
| TBG | O | X | O | O |

Table 1. Comparison of the geolocation methods

**Chapter 3**

**Our approach**

Our goal is to evaluate the following three measurement-based geolocation methods: Shortest Ping, GeoPing, and CBG. Before evaluation, how to implement these methods is a key point. Because most of measurement-based geolocation methods need three necessary elements: VPs, LMs, and T. We decide to use the Twitch ingest server as our T since the IATA airport code of each Twitch ingest server will be displayed on Twitch status [13] or on the OBS list [15], making these Twitch ingest servers suitable as the Ts of our experiments. After determining what the T is, the remaining two necessary element are VPs and LMs. The previous measurement approaches used PlanetLab nodes or other dataset as the VPs and LM. Since PlanetLab is no longer available since 2020 [7]. This thesis exploits the VPN servers provided by NordVPN [8] as the VPs and LMs. NordVPN specifies the longitude and latitude of its servers. We can regard each VPN server as a VP with known location. By connecting to a diverse set of VPN server working as the VPs, we can measure their delay to the LMs and Ts. Furthermore, the VPN servers also work well as the LMs because the VPN servers respond to ping from other machines for functional purposes.

After determining these three necessary elements, we use our laboratory NSLAB as the local host for this experiment and measure the following three delays:

(1) RTT between the local host and the VPN server the local host is connected to. (2) RTT between the local host across the VPN server the local host is connect to and the VPN servers functioning as LMs. (3) RTT between the local host across the VPN server the local host is connect to and the Ts. However, raw measurements don't give us the RTTs between the VP-LM and VP-T pairs that are required for the computation in the measurement-based geolocation methods under study. Therefore, we subtract (1) from (2) to get the RTT of VP-LM and subtract (1) from (3) to get the RTT of VP-T shown in Figure 6.

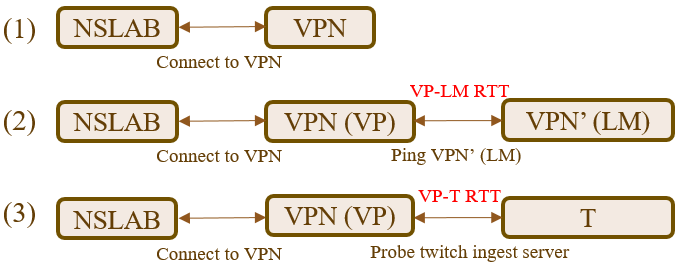


Figure 6. Required RTTs in measurement-based geolocation methods

After implementing these geolocation methods, we use two evaluation parameters: rank difference and error distance. About rank difference, we will come to a list of potential T. Each geolocation method takes different metric to rank the mapping priority. For example, Shortest Ping ranks the priority according to how small the RTT is. GeoPing ranks the priority according to how short the Euclidean distance between target fingerprint and the VPN fingerprint. CBG ranks the priority according to how many circle covers. If the rank of the ground truth location is k, then the rank difference will be k-1 because the ideal rank is 1. For example, in Table 2, T is Mexico, the rank of T’s ground truth location is 5. Hence the rank difference will be 5 - 1 = 4. As for error distance, it is the geo-distance between a mapped VPN geolocation and the T’s ground truth location. And, we will use the two metrics to evaluate the performance of each geolocation method.

|  |  |
| --- | --- |
| Rank | VPN |
| 1 | Dallas |
| 2 | Chicago |
| 3 | Phoenix |
| 4 | Manassas |
| **5** | **Mexico** |
| 6 | Kansas City |
| 7 | Charlotte |
| 8 | Toronto |
| 9 | Salt Lake City |
| 10 | Saint Louis |

Table 2. Rank of the geolocation mapping of Mexico ingest server

**Chapter 4**

**Measurement Methodology**

**4.1 Experiment Design**

Our goal is to compare the existing geolocation methods. Therefore, before evaluation, we have to implement these geolocation methods. We use known location VPNs as our VPs and LMs, and use Twitch ingest servers as our T for implementation. In addition, in order to use VPN more efficiently, we crawl the VPN information and format it into a VPN list (4.1.1). After establishing a VPN list, it is important to know which VPNs are needed, and we finally decided to select one VPN from each VPN server location provided by NordVPN as our reference point (4.1.2). Next, we need to think about how to exploit VPN to implement these geolocation methods (4.1.3). After realizing how to make good use of VPN, we find there are some necessary RTTs should be calculated in order to implement these geolocation methods (4.1.4). Therefore, in the chapter 4.2, we will talk more about how to get these necessary RTTs.

**4.1.1 Establish a VPN List**

Use the following command to get VPN information:

curl --silent "https://api.NordVPN.com/v1/servers?limit=50000" | jq

It can derive a json format VPN information. For example, the information below is one of the VPN servers in Japan.

"id": 963593,

"created\_at": "2020-06-09 10:37:53",

"updated\_at": "2022-04-13 17:44:24",

"name": "Japan #556",

"station": "89.187.161.144",

"ipv6\_station": "",

"hostname": "jp556.NordVPN.com",

"load": 21,

"status": "online",

"cpt": 0,

"locations": [

{

"id": 3984,

"created\_at": "2018-06-18 14:45:14",

"updated\_at": "2018-06-18 14:45:14",

"latitude": 35.685,

"longitude": 139.751389,

"country": {

"id": 108,

"name": "Japan",

"code": "JP",

"city": {

"id": 4633349,

"name": "Tokyo",

"latitude": 35.685,

"longitude": 139.751389,

"dns\_name": "tokyo",

"hub\_score": 0

}

}

]

After crawling the VPN information, we organize the data into a two-dimensional array to store VPN codenames with the same location in the same row. Then, we add some important VPN information to the front of the line. The format is below.

Location 1: [latitude1], [longitude1], [Country1], [City1], [numbers of Servers1], [domain1.1],

[domain1.2], [domain1.3], ...

Location 2: [latitude2], [longitude2], [Country2], [City2], [numbers of Servers2], [domain2.1], [domain2.2], [domain2.3], ...

Location 3: [latitude3], [longitude3], [Country3], [City3], [numbers of Servers3], [domain3.1], [domain3.2], [domain3.3], ...

Besides, after knowing the latitude and longitude of VPNs, VPN distribution can also be drawn on the map shown in Figure 7. We can see that these VPNs are more densely distributed in Europe and North America.

一張含有 地圖 的圖片

自動產生的描述

Figure 7. NordVPN distribution

**4.1.2 NordVPN Selection**

We select only one VPN server for one location. The reason one is no need of more than one server from the same location, because the perspectives of the two servers at the same location as VPs are the same. And the location of the servers as LMs are also the same. Reason two is selecting at least one VPN server from different locations is required. There can't be less than 1 server from the same location, because we are losing a certain perspective or a certain LM. One never knows if having these different VP/LM will make a difference, and we may lose important reference points. Hence, we select one server for each of the locations provided by NordVPN. We made some observations and statistics on VPN location in Figure 8. Total number of servers is 5332. Total number of locations is 82. Number of countries is 60. North America and Europe have the most servers, 2352, 2174 respectively. In addition, many countries have just one location, but some big countries have multiple locations. For example, United Stated (16), Australia (5), Canada (3), France (2), Germany (2). Furthermore, a big city location has more servers. Therefore, big countries tend to have more servers. For example, in Europe, UK has the most 432 servers. The second is Germany with 242 servers. The third is France with 215 servers. In North America, New York has the most 433 servers. The second is Los Angeles with 245 servers. The third is Chicago with 208 servers.

After establishing the VPN list, how to use VPN to implement these geolocation methods is a key point.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **VPN** | **Servers** | **VPN** | **Servers** | **VPN** | **Servers** | **VPN** | **Servers** |
| Taipei | 123 | Phoenix | 39 | Chisinau | 4 | Sofia | 17 |
| Tokyo | 83 | Seattle | 95 | Warsaw | 70 | Steinsel | 19 |
| Seoul | 17 | Salt Lake City | 35 | Vilnius | 5 | Tirana | 7 |
| Hong Kong | 83 | Vancouver | 123 | Marseille | 36 | Sarajevo | 3 |
| Kuala Lumpur | 18 | Denver | 70 | Milan | 68 | Tallinn | 14 |
| Hanoi | 14 | Kansas City | 28 | Paris | 179 | Skopje | 2 |
| Bangkok | 9 | Dallas | 178 | Riga | 17 | Reykjavik | 15 |
| Singapore | 67 | Saint Louis | 73 | Brussels | 56 | Oslo | 60 |
| Mumbai | 34 | Toronto | 161 | Frankfurt | 161 | Prague | 33 |
| Istanbul | 10 | Chicago | 208 | Dublin | 54 | Ukraine | 10 |
| Nicosia | 8 | Buffalo | 120 | London | 432 | Copenhagen | 62 |
| Tel Aviv | 19 | Manassas | 91 | Amsterdam | 172 | Budapest | 19 |
| Jakarta | 19 | New York | 433 | Lisbon | 32 | Athens | 15 |
| Perth | 30 | Charlotte | 24 | Berlin | 81 | Helsinki | 38 |
| Sydney | 70 | Montreal | 149 | Vienna | 51 | Stockholm | 163 |
| Adelaide | 26 | Atlanta | 84 | Bratislava | 21 | Bucharest | 18 |
| Melbourne | 39 | Miami | 105 | Madrid | 63 | Brazil | 30 |
| Brisbane | 35 | Mexico | 15 | Zagreb | 8 | Tbilisi | 3 |
| Auckland | 26 | San Jose | 11 | Ljubljana | 5 | Johannesburg | 22 |
| Los Angeles | 245 | Santiago | 7 | Belgrade | 20 |  |  |
| San Francisco | 91 | Buenos Aires | 21 | Zurich | 111 |  |  |

Figure 8. The number of VPN servers of each location

**4.1.3 Exploit VPN to Implement These Geolocation Methods**

**4.1.3.1 Shortest Ping**

Every time connecting a VPN server, calculate the RTT between the VPN and a Twitch ingest server as the T. There are 82 NordVPN servers, so there will be 82 RTTs, map ingest server to the VPN with the smallest RTT

**4.1.3.2 GeoPing**

The RTT between all VPNs (as VP) and one ingest server as T will be calculated as the T fingerprint, and the RTT between all VPNs (as VP) and each VPN (as LM) will be calculated as the LM fingerprints. There will be 82 VPN fingerprints as the LM fingerprint. Find the VPN with the most similar fingerprint to T fingerprint (with minimum Euclidean distance between LM fingerprint and T fingerprint). Finally locate T at the VPN location.

**4.1.3.3 CBG**

This method is produced in the process of implementing CBG method.

The steps are following:

For each VPN as VP,

1. Treat other VPNs as LMs, ping these LMs, and then calculate out the RTT between this VP and other LMs.
2. By knowing the latitude and longitude of all VPNs, the geo-distance between VPNs can also be calculated. Thus we can get the geo-distance between this VP and other LMs.
3. Plot the relationship between this VP and other LMs (marked by blue nodes):   
   X-axis is Distance (km) Y-axis is RTT (ms) shown in Figure 9.
4. Find bestline. This VP can find a bestline and satisfy the requirement that all points are above the bestline. The method is as follows:

# The number of LMs (nodes) is n

for each i = (1..n-1)

for each j=(i+1..n)

find a line through nodei and nodej

among all (m\_ij, b\_ij), this VP’s bestline will be 𝑦 = 𝑚𝑥 + 𝑏

where 𝑚, 𝑏 satisfy the following n conditions

1. m, b > 0

2. minimize ∑ 𝑦\_k − 𝑚𝑥\_k − 𝑏, k in (1..n)

3. 𝑦\_k − 𝑚𝑥\_k − 𝑏 >=0 for every k in (1..n)

where , >= 0 to minimize

, and this VP’s bestline will be

1. Calculate the RTT between this VP and a Twitch ingest server as T
2. Use the RTT between this VP and T as inputs and using to derive the estimated distance between this VP and T. For example in Figure 9, 100 ms T RTT will be converted to 5500 km as estimated T RTT.
3. Use the geolocation of this VP as the center of the circle and the estimated distance from this VP to T to draw a circle.
4. Because there are 82 VP in this experiment. So, there are 82 estimated T distance, and we can draw 82 circles.
5. Finally, for each LM, we compute the number of circles cover the LM location, and we locate the T at the LM location that covered by the most circles. If there are more than two LM locations covered by the most circles, we will locate the T at the LM location with smaller estimated distance. For example, in Figure 10, VP 1 to 3 generate bestline 1 to bestline 3 respectively. And estimated T distance 1 to distance 3 are generated by bestline 1 to bestline 3 respectively. Since the geolocation of LM “a” covered by the most circles (3 circles), thus CBG maps T to the location of LM “a”.

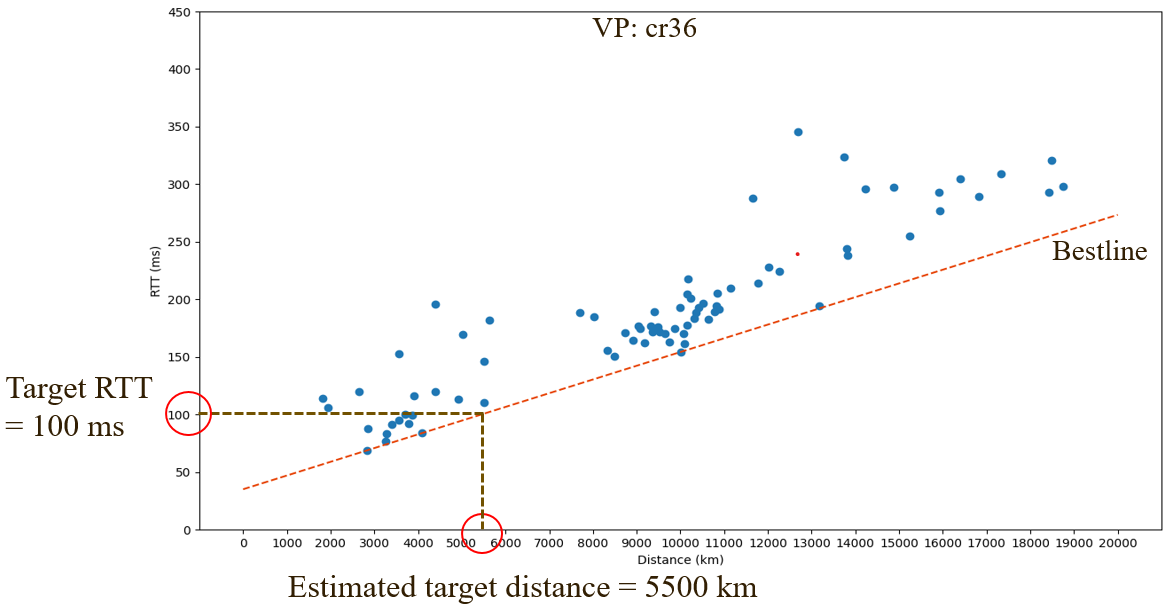
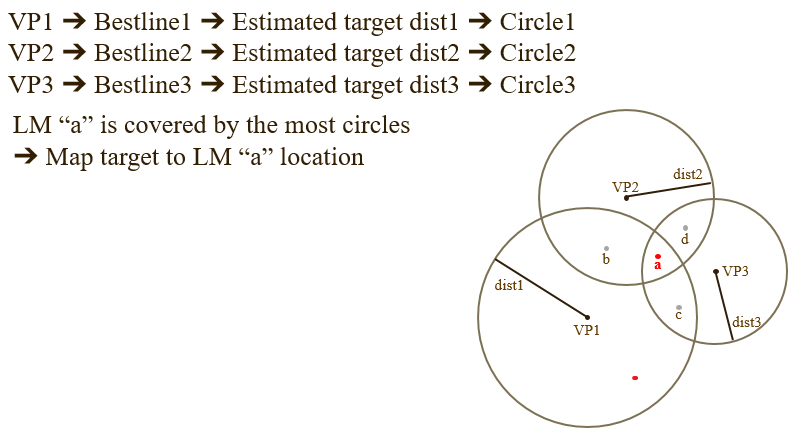


Figure 9. Sample scatter plot of geographic distance and network delay

  
Figure 10. Map T to the LM location according to number of covers

**4.1.3.4 Shortest Distance**

This geolocation method is generated when implementing CBG

The steps are following:

1. The previous steps are the same as step.1 to step.6 in CBG. We obtain the estimated distance between the ingest server and 82 VPNs.
2. Locate the Twitch ingest server at the closest VPN according to estimated T distance. For example in Figure 11, T will be located at VPN1’s location because the estimated distance between T and VPN1 is shortest.

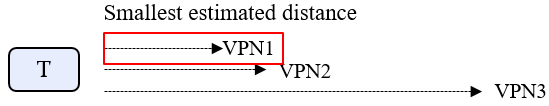


Figure 11. Map T to the VPN with smallest estimated distance

* + 1. **Three Necessary RTTs in These Geolocation Methods**
       1. **RTT1 between NSLAB and VPN**

Simply ping each VPN from NSLAB, use tcpdump to intercept the packet and calculate the RTT1 between NSLAB and each VPN.

**4.1.4.2 RTT2 between VPN and Twitch Ingest Server**

After connecting a VPN, we cannot directly ping any Twitch ingest server since Twitch ingest servers block ICMP packets for some reasons, such as preventing DDoS and other security issues. Therefore, it seems that RTT cannot be obtained directly by pinging ingest servers. Hence, we pretend that we are streamers and use OBS to perform live streaming. Furthermore, Twitch ingest server list can be found in OBS server list, we will locate all Twitch ingest servers on the list which is shown in Figure 12 and Table 3. After starting streaming, use tcpdump to intercept the packet transmission between VPN and the Twitch ingest server and calculate the RTT from the TCP data and acknowledgement packet pairs. It should be noted that this raw RTT actually contains two segments "RTT between NSLAB and VPN" plus "RTT between VPN and Twitch ingest server". As a result, we subtract RTT between NSLAB and VPN from this raw RTT to get the RTT2 between VPN and Twitch ingest server. We will calculate the RTT2 between 82 VPNs and 60 Twitch ingest servers, which are about 4920 RTT2 pairs.

* + - 1. **RTT3 between VPNs**

VPN can actively ping and passively be pinged. Therefore, VPNs ping each other to obtain the RTT between any two VPNs. Similarly, we subtract RTT between NSLAB and VPN from the raw RTT to get RTT3 between VPNs, which are about 6724 RTT3 pairs.

一張含有 文字, 監視器, 螢幕擷取畫面, 電子用品 的圖片

自動產生的描述

Figure 12. Twitch ingest server list in OBS

|  |  |
| --- | --- |
| Ashburn3 | Queretaro1 |
| Ashburn5 | Queretaro2 |
| Atlanta | Miami |
| Vienna2 | Amsterdam2 |
| Fortaleza1 | Amsterdam3 |
| Rio | New York |
| Toronto | Norway Oslo |
| Chicago IL2 | Phoenix AZ |
| Chicago IL3 | Warsaw |
| Prague | Portland OR |
| Copenhagen | Salt Lake City |
| Helsinki | San Francisco |
| Marseille | San Jose |
| Marseille2 | Singapore |
| Paris10 | Seoul1 |
| Berlin | Seoul3 |
| Düsseldorf1 | Stockholm |
| Frankfurt2 | Sydney1 |
| Frankfurt5 | Sydney2 |
| Hong Kong | Taipei1 |
| Houston | Taipei3 |
| Bangalore1 | Bangkok |
| Chennai | London3 |
| Hyderabad1 | London4 |
| Mumbai | Sao Paulo |
| Milan2 | Quebec |
| Tokyo1 | Dallas |
| Tokyo3 | Denver |
| Tokyo5 | Seattle |
| Los Angeles | Madrid |

Table 3. Full list of Twitch ingest server

* 1. **Packet Collection and RTT Measurement**

The packet collection architecture is shown in Figure 13. Whether we want to get RTT② or RTT③ we need to connect to a VPN and finally collect the traffic in NSLAB’s machine. The RTT① must be calculated here. Therefore, we must subtract the additional RTT① from the RTT①+② and RTT①+③

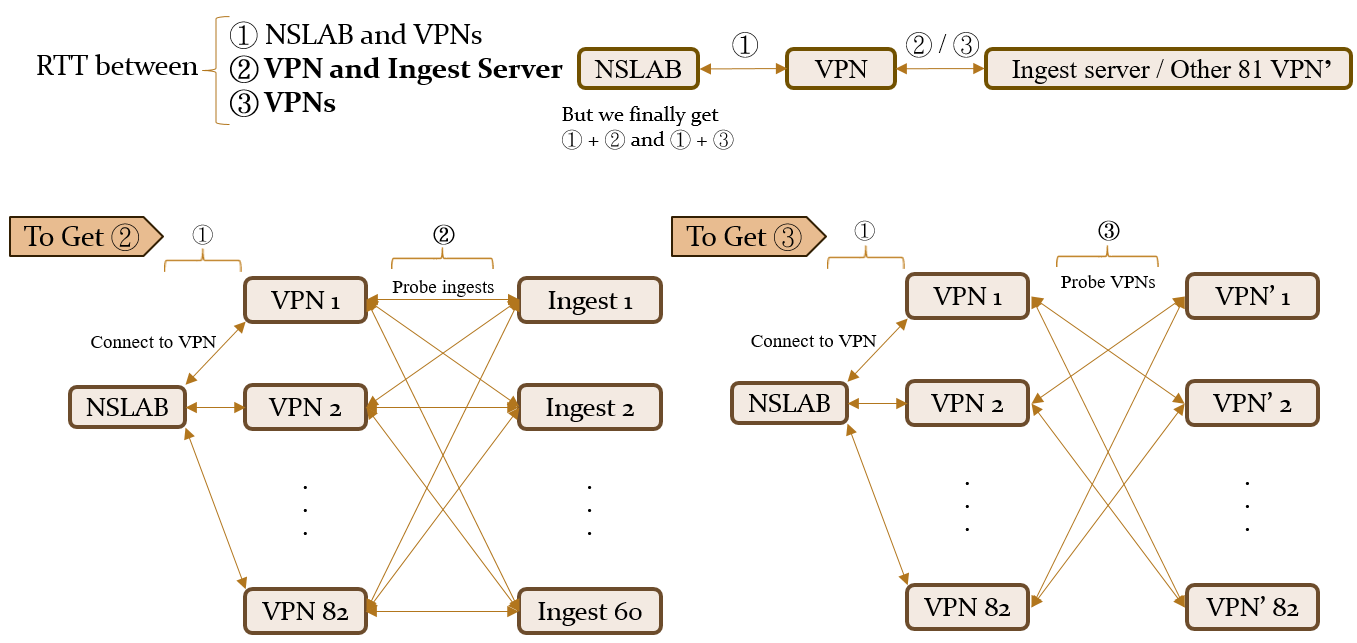
****

Figure 13. Packet collection architecture

**4.2.1 Get RTT1 between NSLAB and VPN**

**4.2.1.1 Packet Collection**

The steps are following:

1. Simply ping each VPN from Nslab for 30 seconds.
2. Use tcpdump to intercept the packet between NSLAB and each VPN  
   host ip in NSLAB is 140.112.42.xxx and the other side is the ip of VPN. For example, the packets captured by ping ca944.NordVPN.com is as follows.

22:07:27.219059 IP 140.112.42.155 > 172.83.40.219: ICMP echo request, id 3, seq 16 ...

22:07:27.373360 IP 172.83.40.219 > 140.112.42.155: ICMP echo reply, id 3, seq 16 ...

22:07:28.220435 IP 140.112.42.155 > 172.83.40.219: ICMP echo request, id 3, seq 17 ...

22:07:28.375005 IP 172.83.40.219 > 140.112.42.155: ICMP echo reply, id 3, seq 17 ...

* + - 1. **RTT Computation**

1. Treat the same request seq and reply seq as a pair with RTT equal to Time\_reply -Time\_request. For the same example, ping ca944.NordVPN.com.

**22:07:27.219059** IP 140.112.42.155 > 172.83.40.219: ICMP echo **request**, id 3, **seq 16** ...

**22:07:27.373360** IP 172.83.40.219 > 140.112.42.155: ICMP echo **reply**, id 3, **seq 16** ...

The RTT of the pair is 22:07:27.373360 - 22:07:27.219059 = 0.154301 (ms)

1. Take the minimum RTT from all RTT pairs as the RTT1 between NSLAB and VPN shown in Figure 14.

****

Figure 14. RTT1 between NSLAB and VPN

* + 1. **Get RTT2 between VPN and Twitch Ingest Server**

**4.2.2.1 Packet Collection**

**We do not use containers or VMs to prevent more overhead from running the application process because it may lead to more bias in the measured RTT. Therefore, we use three machines in NSLAB with Linux OS for simultaneous crawling to speed up the experiment. Each machine selects a different Twitch ingest server sequentially according to OBS server list order for probing and run the following script to intercept the packet transmission between VPN and Twitch ingest server.**

**# Packet Collection**

**for each Twitch ingest server:**

**for each VPN:**

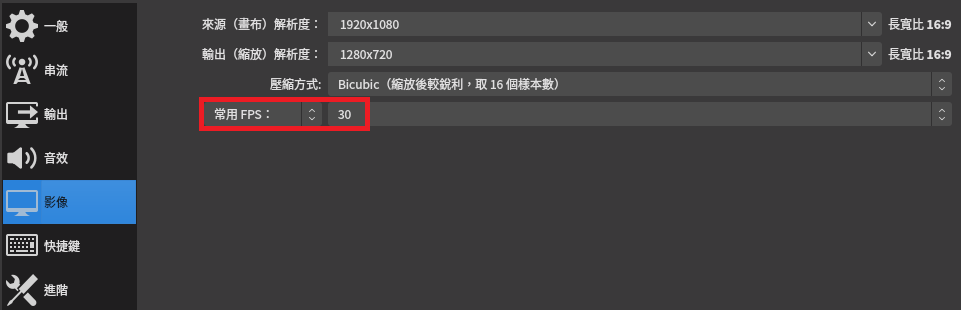
1. **Select a NordVPN for Connection**
   1. Sequentially connect a NordVPN from our NordVPN list. Besides, the locations of VPNs we choose will be unique.
   2. Check the VPN connection status. For example, if we use the command “nordvpn connect Japan”, and it will generate three possible result shown in Figure 15. If the VPN is hard to be connected (VPN Failure), it will change to the next VPN server in another location to crawl the same ingest server.
2. *You are connected to Japan #429 (jp429 .nordvpn.com)!* (Success)
3. *Whoops! Something went wrong. Please try again. If the problem persists, contact our customer support.* (VPN Failure)
4. *The specified server is not available at the moment or does not support your connection settings.* (VPN Failure)

Figure 15. VPN connection status

1. **OBS Connection Test**

**About OBS settings, we use 200 kbps video bitrate and 30 fps, and set nothing video source shown in Figure 16. These settings help increase the stability of the stream and reduce the possibility of congestion.**

****

****

****

**Figure 16. OBS settings**

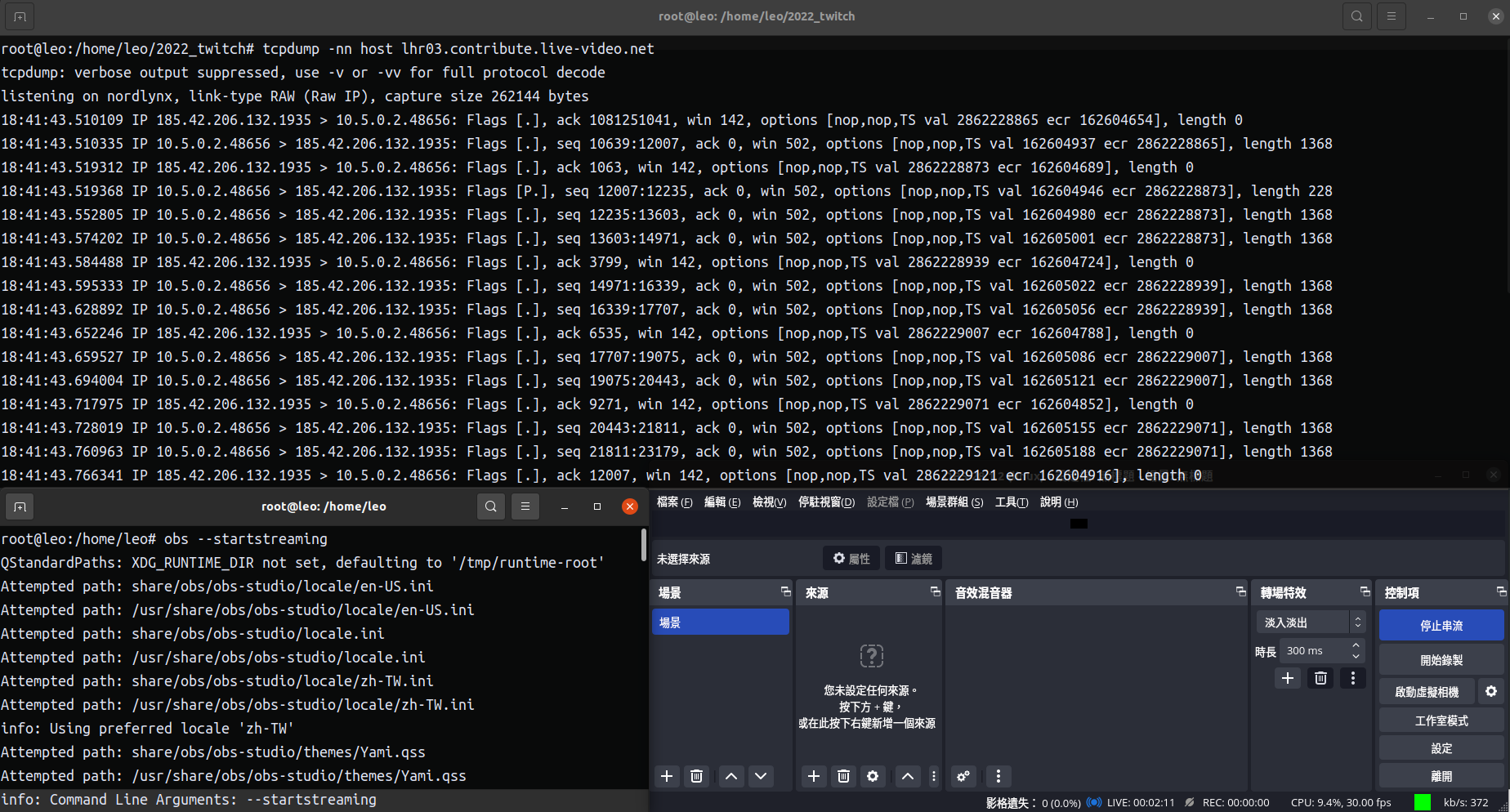
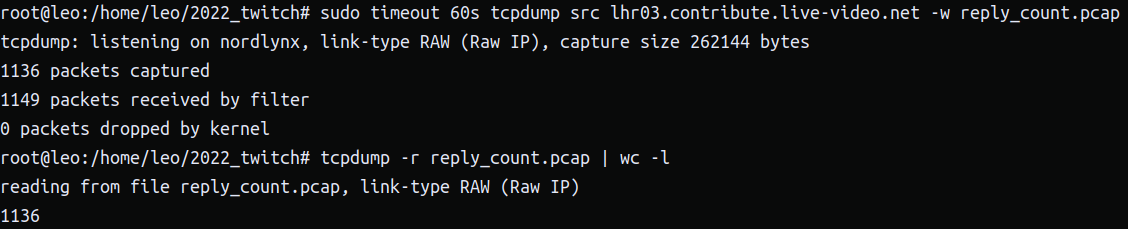
After successfully connecting the VPN, we open the OBS and start streaming. There will be packet transmission between VPN and Twitch ingest server. Figure 17 shows the packet transmission between Japan VPN (IP: 10.5.0.2) and London3 ingest server (IP: 185.42.206.132). And we use tcpdump **to** examine whether there is enough packet transmission between the VPN and Twitch ingest server or not. If the number of packets replied by the Twitch ingest server within one minute is more than 30 it will go to next step for formal interception, For example in Figure 18, it shows there are 1136 replied packets from London3 ingest server, and 1136 is larger than 30, so it will be considered an successful OBS connection, and then go to next step for formal interception. If the number of replied packets is less than 30, it will be regarded as OBS connection failure, and it will change to the next NordVPN at another location to crawl the same ingest server.

Figure 17. Interception of packet transmission between Japan VPN and London3 ingest server

Figure 18 Count the number of replied packets from London3 ingest server

**Besides, we show that how to use OBS and shell command to start streaming, end streaming, switch to another Twitch ingest sever, and reset the Twitch ingest server automatically in Figure 19, 20, 21, 22. It is worth noting that we cannot switch to another Twitch ingest server simply by OBS command and we should use shell command to operate the OBS GUI. Xdotool [16] is a shell command that can be used to operate our keyboard and mouse instruction.**

function start\_streaming ()

{

obs --startstreaming&

}

**Figure 19. Start streaming with OBS**

function end\_streaming ()

{

pidof obs | xargs kill

}

**Figure 20. End streaming and close the OBS**

function switch\_ingest()

{

obs& # open the obs

sleep 5

for ((i=1;i<=9;i++))

do

xdotool key Tab

done

xdotool key space

xdotool key Down

for ((i=1;i<=2;i++))

do

xdotool key Tab

done

**xdotool key Down # choose ingest**

for ((i=1;i<=6;i++))

do

xdotool key Tab

done

xdotool key space

pidof obs | xargs kill # obs disconnect

}

**Figure 21. Switch the ingest server to another one**

function reset\_ingest()

{

obs&

sleep 5

for ((i=1;i<=9;i++))

do

xdotool key Tab

done

xdotool key space

xdotool key Down

for ((i=1;i<=2;i++))

do

xdotool key Tab

done

**# reset to the first ingest**

**for ((re=1;re<=65;re++))**

**do**

**xdotool key Up**

**done**

**xdotool key Down**

for ((i=1;i<=6;i++))

do

xdotool key Tab

done

xdotool key space

pidof obs | xargs kill # obs disconnect

**Figure 22. Reset ingest server to the first one**

1. **Intercept Packets by Formal Tcpdump  
   OBS is still opened. Start tcpdump to intercept the packet transmission between the VPN and Twitch ingest server for 30 seconds.** 
   * + 1. **RTT Computation**
2. Record the timestamp and seqnum of all data packets sent from VPN to Twitch ingest server
3. Record the timestamp and acknum of all ack packets reply to the VPN from Twitch ingest server
4. Find all the seqnum(i) = acknum(j) pairs with RTT equal to Time\_ack(j) - Time\_seq(i). For the same example, the following is Hong Kong VPN to Tokyo ingest server.

**09:05:39.912347** IP 10.8.3.3.33882 > 52.223.218.164.1935: Flags [P.], seq 568718:**569481** ...

…

…

**09:05:39.986342** IP 52.223.218.164.1935 > 10.8.3.3.33882: Flags [.], ack **569481** ...

The RTT of the pair is 09:05:39.986342 - 09:05:39.912347 = 73.995 (ms)

1. If there is a data packet retransmission. Even if an ack packet is received, its RTT is still not calculated as shown in the log file below.

20:45:11.103634 IP 10.8.2.6.48458 > 45.113.128.13.1935: Flags [P.], seq 80233480:80234794

20:45:11.103678 IP 10.8.2.6.48458 > 45.113.128.13.1935: Flags [.], seq 80234794:80236108 20:45:48.745752 IP 10.8.2.6.48458 > 45.113.128.13.1935: Flags [.], seq 80233480:80234794

20:45:48.745794 IP 10.8.2.6.48458 > 45.113.128.13.1935: Flags [.], seq 80234794:80236108

20:45:49.257436 IP 45.113.128.13.1935 > 10.8.2.6.48458: Flags [.], ack 80234794

It can be seen that seq 80233480:80234794 has the phenomenon of retransmission

1. Take the minimum RTT from all RTT pairs as RTT’ = RTT1 + RTT2. Since RTT**’** we get is actually “RTT1 between NSLAB and VPN” plus “RTT2 between VPN and Twitch ingest server”. Therefore, we subtract RTT1 from the RTT**’** to get the RTT2 between VPN and Twitch ingest server shown in Figure 23.

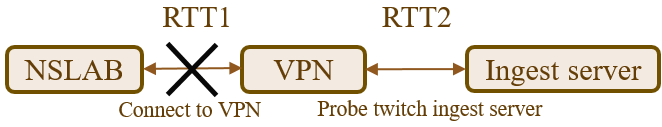
****

Figure 23. RTT2 between VPN and Twitch ingest server

* + 1. **Get RTT3 between VPNs**

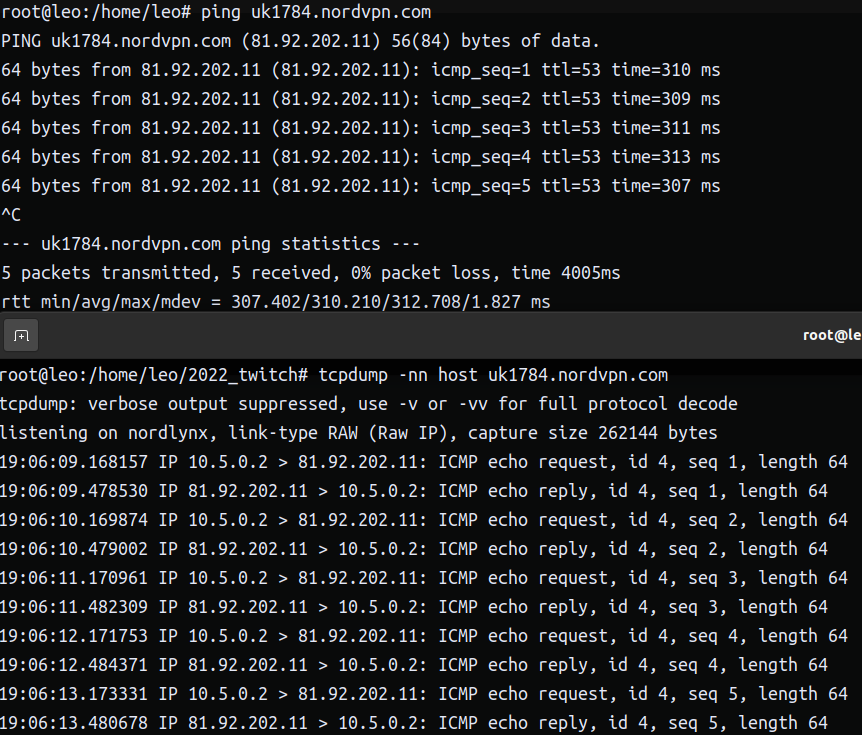
**4.2.3.1 Packet Collection**

**# Packet Collection**

for i in each VPN:

for j in each VPN:

1. Select a VPNi for Connection from the VPN list with unique location and check the connection status to determine VPN Failure or successful connection.
2. Ping each VPNj to generate the packet transmission between VPNi and VPNj. And Use tcpdump to intercept the traffic. For example, in Figure 24, VPNi we connect is Japan VPN, and VPNj we ping is UK VPN   
   (IP: 81.92.202.11). We can get the traffic between the two VPNs.



**Figure 24. Intercept the traffic between connected VPN and pinged VPN**

**4.2.3.2 RTT Computation**

1. Treat the same request seq and reply seq as a pair with RTT equal to Time\_reply - Time\_request. For the same example, connect VPN ca944 and then ping hk203.NordVPN.com.

22:49:05.787483 IP 10.8.2.6 > 84.17.37.226: ICMP echo request, id 4, seq 39, length 64

22:49:06.122541 IP 84.17.37.226 > 10.8.2.6: ICMP echo reply, id 4, seq 39, length 64

The RTT of the pair is 22:49:06.122541 - 22:49:05.787483 = 0.335058 (ms)

1. Take the minimum raw RTT from all RTT pairs as RTT**’**. This RTT**’** we get is actually RTT1 between NSLAB and VPN plus “RTT3 between VPN and other 81 VPN**’**. Therefore, we subtract RTT1 from RTT’ to get RTT3 between VPNs shown in Figure 25.



Figure 25. RTT3 between VPNs

**Chapter 5**

**Measurement Examination**

* 1. **RTT1 between NSLAB and VPN**

In general, the shorter the geographical distance between NSLAB and the VPN, the smaller the RTT. In Table 4, most VPNs in Asia such as Tokyo, Seoul, and HK have shorter network distances from NSLAB. Besides, The RTT between America and NSLAB is about 100~200 ms, and RTT between Europe and NSLAB is about 200~300 ms.

****

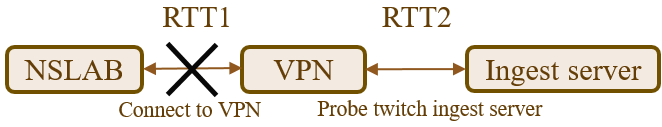
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **VPN** | **RTT(ms)** | **VPN** | **RTT(ms)** | **VPN** | **RTT(ms)** | **VPN** | **RTT(ms)** |
| Taipei | 27 | Phoenix | 143 | Chisinau | 227 | Sofia | 263 |
| Tokyo | 33 | Seattle | 151 | Warsaw | 229 | Steinsel | 263 |
| Seoul | 52 | Salt Lake City | 158 | Vilnius | 231 | Tirana | 265 |
| Hong Kong | 57 | Vancouver | 158 | Marseille | 232 | Sarajevo | 271 |
| Kuala Lumpur | 72 | Denver | 163 | Milan | 233 | Tallinn | 272 |
| Hanoi | 76 | Kansas City | 172 | Paris | 235 | Skopje | 272 |
| Bangkok | 77 | Dallas | 172 | Riga | 235 | Reykjavik | 274 |
| Singapore | 79 | Saint Louis | 186 | Brussels | 237 | Oslo | 276 |
| Mumbai | 132 | Toronto | 190 | Frankfurt | 240 | Prague | 283 |
| Istanbul | 249 | Chicago | 191 | Dublin | 243 | Ukraine | 283 |
| Nicosia | 315 | Buffalo | 197 | London | 244 | Copenhagen | 285 |
| Tel Aviv | 317 | Manassas | 198 | Amsterdam | 244 | Budapest | 286 |
| Jakarta | 72 | New York | 198 | Lisbon | 245 | Athens | 289 |
| Perth | 125 | Charlotte | 199 | Berlin | 246 | Helsinki | 290 |
| Sydney | 131 | Montreal | 200 | Vienna | 251 | Stockholm | 296 |
| Adelaide | 138 | Atlanta | 201 | Bratislava | 251 | Bucharest | 302 |
| Melbourne | 141 | Miami | 203 | Madrid | 252 | Brazil | 310 |
| Brisbane | 146 | Mexico | 195 | Zagreb | 254 | Tbilisi | 340 |
| Auckland | 157 | San Jose | 263 | Ljubljana | 254 | Johannesburg | 372 |
| Los Angeles | 130 | Santiago | 297 | Belgrade | 258 |  |  |
| San Francisco | 136 | Buenos Aires | 337 | Zurich | 263 |  |  |

Table 4. RTT between NSLAB and VPN

* 1. **RTT2 between VPN and Twitch Ingest Server**

**5.2.1 RTT between NSLAB across VPN and Twitch Ingest Server**

A part of our measurement RTT shown in Table 5 is RTT1 + RTT2 because we connect a VPN and then generate the traffic between the VPN and the Twitch ingest server. Therefore, this “RTT1 + RTT2” need to subtract RTT1 (between NSLAB and VPN showed in Table 4.) to get RTT2 between VPN and Twitch Ingest Server.

****

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| VPN Ingest | Sydney | HK | Tokyo | Singapore | Copenhagen | Helsinki | London | Toronto | New York | Atlanta |
| Sydney | 132.103 | 254.181 | 295.180 | 223.025 | 389.237 | 394.846 | 376.974 | 375.431 | 382.195 | 312.870 |
| HK | 218.907 | 23.535 | 73.560 | 55.603 | 275.262 | 290.616 | 216.386 | 209.839 | 225.084 | 220.447 |
| Tokyo | 238.654 | 79.097 | 32.411 | 112.291 | 279.971 | 294.323 | 253.646 | 186.823 | 194.565 | 188.146 |
| Singapore | 338.148 | 134.129 | 175.082 | 101.122 | 359.147 | 366.235 | 282.554 | 323.063 | 328.543 | 319.671 |
| Copenhagen | 596.341 | 493.471 | 539.332 | 462.212 | 287.089 | 296.306 | 305.503 | 385.260 | 375.167 | 391.109 |
| Helsinki | 616.653 | 529.269 | 536.325 | 454.795 | 282.180 | 274.170 | 298.439 | 367.873 | 357.720 | 374.585 |
| London | 641.080 | 525.425 | 524.739 | 523.938 | 291.460 | 302.197 | 273.422 | 352.468 | 354.676 | 361.098 |
| Toronto | 470.609 | 464.409 | 450.086 | 466.825 | 292.419 | 300.672 | 288.631 | 199.619 | 209.708 | 225.391 |
| New York | 495.777 | 419.674 | 343.896 | 424.664 | 289.366 | 297.646 | 270.480 | 211.532 | 201.864 | 218.981 |
| Atlanta | 479.777 | 458.404 | 435.509 | 448.140 | 307.378 | 317.794 | 296.766 | 227.145 | 221.980 | 201.424 |

Table 5. RTT between NSLAB across VPN and Twitch ingest server

**5.2.2 Unadjusted RTT2 between VPN and Twitch Ingest Server**

After subtracting RTT1 from the RTT between NSLAB across VPN and Twitch ingest server, we get the RTT2 and showed a part of the result in Table 6 as an example.

****

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| VPN Ingest | Sydney | HK | Tokyo | Singapore | Copenhagen | Helsinki | London | Toronto | New York | Atlanta |
| Sydney | 1.103 | 123.181 | 164.180 | 92.025 | 258.237 | 263.846 | 245.974 | 244.431 | 251.195 | 181.870 |
| HK | 161.907 | **-33.465** | 16.560 | **-1.397** | 218.262 | 233.616 | 159.386 | 152.839 | 168.084 | 163.447 |
| Tokyo | 205.654 | 46.097 | **-0.589** | 79.291 | 246.971 | 261.323 | 220.646 | 153.823 | 161.565 | 155.146 |
| Singapore | 259.148 | 55.129 | 96.082 | 22.122 | 280.147 | 287.235 | 203.554 | 244.063 | 249.543 | 240.671 |
| Copenhagen | 311.341 | 208.471 | 254.332 | 177.212 | 2.089 | 11.306 | 20.503 | 100.260 | 90.167 | 106.109 |
| Helsinki | 326.653 | 239.269 | 246.325 | 164.795 | **-7.820** | **-15.830** | 8.439 | 77.873 | 67.720 | 84.585 |
| London | 397.080 | 281.425 | 280.739 | 279.938 | 47.460 | 58.197 | 29.422 | 108.468 | 110.676 | 117.098 |
| Toronto | 280.609 | 274.409 | 260.086 | 276.825 | 102.419 | 110.672 | 98.631 | 9.619 | 19.708 | 35.391 |
| New York | 297.777 | 221.674 | 145.896 | 226.664 | 91.366 | 99.646 | 72.480 | 13.532 | 3.864 | 20.981 |
| Atlanta | 278.777 | 257.404 | 234.509 | 247.140 | 106.378 | 116.794 | 95.766 | 26.145 | 20.980 | 0.424 |

Table 6. Unadjusted RTT2 between VPN and Twitch ingest server

In Table 6, there are some RTTs < 0. It means that the RTT1 we get between NSLAB and VPN by “ping” is greater than the RTT1 between NSLAB and VPN by “VPN connection” plus RTT2 between VPN and ingest server. We infer that the reason may be the ICMP packets generated by ping are generally given a lower priority by NordVPN mechanism due to some information security issues. On the other hand, when we connect directly to the VPN, NordVPN mechanism may treat it to be a normal use and put the priority level higher. Therefore, RTT1 generated by the direct connection to the VPN is usually smaller than the RTT by pinging VPN from NSLAB. Furthermore, when the RTT2 between VPN and Twitch ingest server is small, RTT1 (by VPN connection) plus RTT2 basically depends on the size of the RTT1. As a result, it may lead to

And the condition is shown in Figure 26. In this case, since the RTT2 is small, the negative value can be regarded as the network delay between the VPN and the target Twitch ingest server is very small. But we still need to correct negative values to avoid making the analytical calculations more difficult.

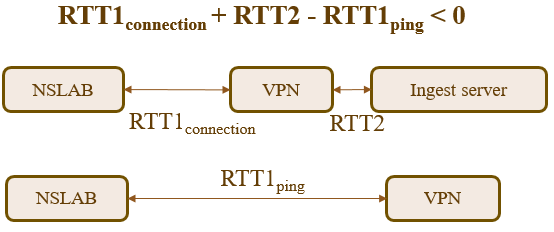


Figure 26. The condition of RTT2 in Table 6 less than zero

To solve this problem, we decided to adjust the RTT1 (by ping) between NSLAB and each VPN. Because we believe that the traffic generated by ping will produce additional overhead and increase the delay time compared to the traffic generated by connecting directly to VPN. Therefore, to each VPN, if there is any minimum RTT2 between this VPN and all ingest server is negative, all RTT2 between them should be subtracted by this negative RTT, and as a result the minimum RTT2 between them will be 0. The way to subtract the negative value from the RTT2 between the VPN and all the ingest server is equal to add the negative value to the RTT1 (by ping) to reduce RTT1 (by ping) because the value of RTT1 (by VPN connection) + RTT2 – RTT1 (by ping) are the same, and reducing RTT1 help us eliminate some extra overhead of RTT. For example, Figure 27 shows the distribution of minimum RTT2s (green curve) of all VPNs. We can discover that some minimum RTT2s are less than zero, such as the minimum RTT2 between Helsinki and all ingest servers is -15.830 ms. And the minimum RTT2 between HK and all ingest servers is -33.465 ms. In order to solve this problem, for example, Figure 28 shows the minimum RTT2 of (VPN Helsinki, Ingest Helsinki) is -15.830 ms. Thus, all RTT between VPN Helsinki and all ingest servers should be subtracted by -15.830 ms. Consequently, after adjusting all the negative RTT to nonnegative value, the adjusted RTT2 shown in Table 7.

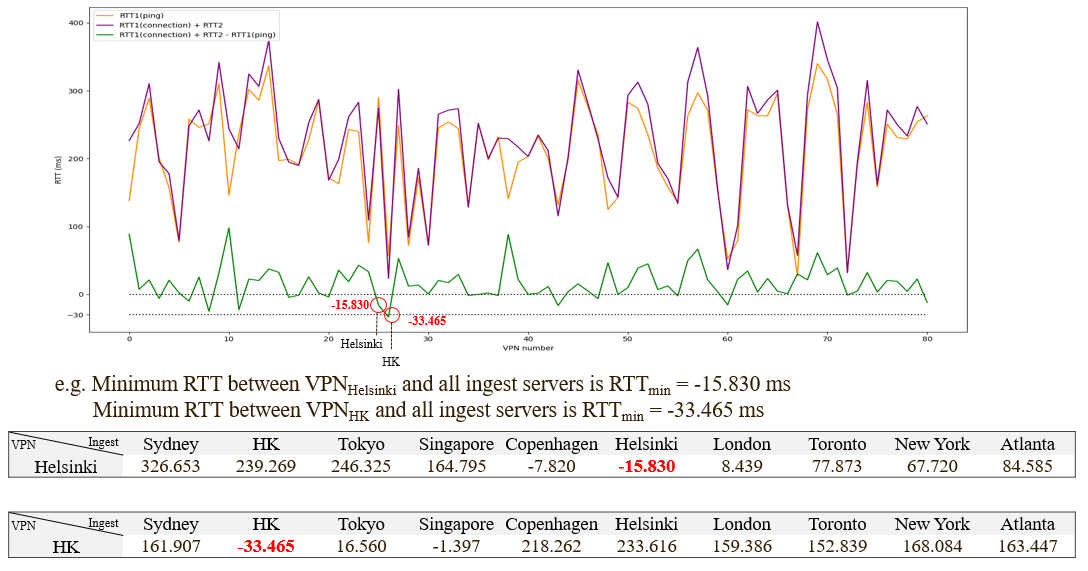


Figure 27. Distribution of minimum RTT

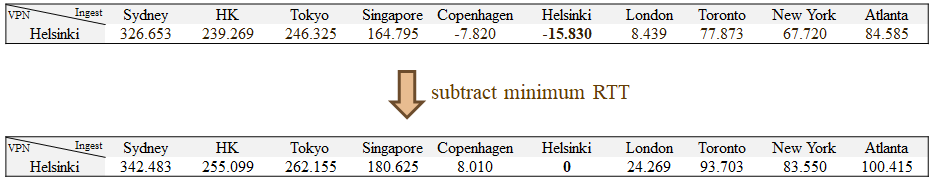


Figure 28. Adjust negative RTT by subtracting minimum RTT less than zero

In Table 7, it can be found that the diagonal value is small, because the VPN and the Twitch ingest server are located in the same city, thus the RTT will be relatively small. Besides, the value of (VPN\_i, T\_j) and the value of (VPN\_j, T\_i) are often similar as well, because one RTT value is generated from the VPN in "i" city probing the ingest server in "j", and the other RTT value is generated from a VPN in "j" city probing an ingest server in "i" city. Furthermore, there are three square frames in Table 7, RTT values in the same frame indicate the VPN and ingest server are on the same continent. And these RTTs are small because different cities on the same continent are not very far from each other.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| VPN Ingest | Sydney | HK | Tokyo | Singapore | Copenhagen | Helsinki | London | Toronto | New York | Atlanta |
| Sydney | 1.103 | 123.181 | 164.180 | 92.025 | 258.237 | 263.846 | 245.974 | 244.431 | 251.195 | 181.870 |
| HK | 195.372 | 0 | 50.025 | 32.068 | 251.727 | 267.081 | 192.851 | 186.304 | 201.549 | 196.912 |
| Tokyo | 206.633 | 47.076 | 0.390 | 80.270 | 247.950 | 262.302 | 221.625 | 154.802 | 162.544 | 156.125 |
| Singapore | 259.148 | 55.129 | 96.082 | 22.122 | 280.147 | 287.235 | 203.554 | 244.063 | 249.543 | 240.671 |
| Copenhagen | 311.341 | 208.471 | 254.332 | 177.212 | 2.089 | 11.306 | 20.503 | 100.260 | 90.167 | 106.109 |
| Helsinki | 342.483 | 255.099 | 262.155 | 180.625 | 8.010 | 0 | 24.269 | 93.703 | 83.550 | 100.415 |
| London | 397.080 | 281.425 | 280.739 | 279.938 | 47.460 | 58.197 | 29.422 | 108.468 | 110.676 | 117.098 |
| Toronto | 280.609 | 274.409 | 260.086 | 276.825 | 102.419 | 110.672 | 98.631 | 9.619 | 19.708 | 35.391 |
| New York | 297.777 | 221.674 | 145.896 | 226.664 | 91.366 | 99.646 | 72.480 | 13.532 | 3.864 | 20.981 |
| Atlanta | 284.676 | 263.303 | 240.408 | 253.039 | 112.277 | 122.693 | 101.665 | 32.044 | 26.879 | 6.323 |

Table 7. Adjusted RTT2 between VPN and Twitch ingest server

**5.3 RTT3 between VPNs**

A part of RTT3 measurement result between VPNs showed a part of measurement result in Table 8. These RTTs have been adjusted like chapter 5.2. It can be found that the diagonal value is zero because the VPN we connected and VPN which is pinged by the connected VPN are in the same location. Besides, VPNs in the same continent have small RTT as well.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| VPN VPN’ | Sydney | HK | Tokyo | Singapore | Copenhagen | Helsinki | London | Toronto | New York | Atlanta |
| Sydney | 0 | 122.204 | 99.093 | 92.21 | 256.181 | 322.814 | 253.441 | 188.256 | 194.397 | 184.645 |
| HK | 123.033 | 0 | 47.197 | 32.196 | 246.841 | 311.243 | 238.894 | 222.636 | 199.757 | 215.802 |
| Tokyo | 98.683 | 52.488 | 0 | 67.136 | 257.347 | 253.619 | 262.827 | 167.999 | 149.161 | 148.153 |
| Singapore | 89.111 | 27.441 | 64.278 | 0 | 160.006 | 296.646 | 158.658 | 225.663 | 223.335 | 213.66 |
| Copenhagen | 258.689 | 211.614 | 260.211 | 164.062 | 0 | 18.888 | 24.066 | 101.833 | 87.252 | 114.558 |
| Helsinki | 307.33 | 254.472 | 248.76 | 301.343 | 0.866 | 0 | 17.486 | 100.532 | 85.356 | 99.915 |
| London | 246.251 | 182.36 | 248.936 | 157.464 | 17.287 | 31.706 | 0 | 74.936 | 70.755 | 84.484 |
| Toronto | 188.125 | 210.895 | 168.035 | 227.785 | 99.421 | 115.375 | 81.785 | 0 | 20.04 | 35.252 |
| New York | 183.989 | 208.616 | 138.067 | 219.023 | 74.646 | 90.667 | 62.923 | 10.1 | 0 | 8.267 |
| Atlanta | 234.989 | 201.367 | 135.265 | 206.119 | 92.839 | 102.305 | 77.483 | 22.181 | 7.992 | 0 |

Table 8. RTT3 between VPNs

**Chapter 6**

**Evaluate Experimental Results**

**6.1 Evaluation Methodology**

After analyzing the three necessary RTTs for implementing the three measurement geolocation methods, next step is to evaluate the performance of these methods. First, the ground truth location of Twitch ingest server must be known to verify that the location we map to is correct or not. Table 9 shows the ground truth location of each Twitch ingest server. However, some few Twitch ingest servers do not have the same city in the VPN, so we do not always use the same city as the ground truth. If the VPN does not have a city corresponding to the ingest server, we will replace it with a nearby city in the VPN. For example, the ground truth of Düsseldorf’s ingest server will be Brussels or Amsterdam. If the geolocation method map target to this VPN location in nearby city (not the same city), rank difference and error distance will still be 0.

After getting the ground truth locations, there are two evaluation parameters available, rank difference and error distance. Table 10 takes Toronto's ingest server as an example. The table shows how each geolocation method generates the rank difference. For example, in the table, GeoPing maps Toronto's ingest server to VPN location (Chicago) which is not the same as the ground truth location, and we can find that the Toronto is the 7th rank of mapping in GeoPing. Thus the rank difference is 6. Besides, each rank is arranged by each method. Shortest Ping ranks the priority according to how small the RTT is. GeoPing ranks the priority according to how short the Euclidean distance between target fingerprint and the VPN fingerprint. CBG ranks the priority according to how many circle covers. Shortest Distance ranks the priority according to how short the estimated distance between target and the VPN. In addition, Table 11 shows the rank difference and error distance of each geolocation method. The error distance is the geo-distance between a mapped VPN geolocation and the Twitch ingest server’s ground truth location.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ingest server** | **Corresponding VPN** | **Ingest server** | **Corresponding VPN** |
| Ashburn3 | ***Manassas*** | Queretaro1 | Mexico |
| Ashburn5 | ***Manassas*** | Queretaro2 | Mexico |
| Atlanta | Atlanta | Miami | Miami |
| Vienna2 | Vienna | Amsterdam2 | Amsterdam |
| Fortaleza1 | Brazil | Amsterdam3 | Amsterdam |
| Rio | Brazil | New York | New York |
| Toronto | Toronto | Norway Oslo | Oslo |
| Chicago IL2 | Chicago | Phoenix AZ | Phoenix |
| Chicago IL3 | Chicago | Warsaw | Warsaw |
| Prague | Prague | Portland OR | ***Seattle*** |
| Copenhagen | Copenhagen | Salt Lake City | Salt Lake City |
| Helsinki | Helsinki | San Francisco | San Francisco |
| Marseille | Marseille | San Jose | ***San Francisco*** |
| Marseille2 | Marseille | Singapore | Singapore |
| Paris10 | Paris | Seoul1 | Seoul |
| Berlin | Berlin | Seoul3 | Seoul |
| Düsseldorf1 | ***Brussels / Amsterdam*** | Stockholm | Stockholm |
| Frankfurt2 | Frankfurt | Sydney1 | Sydney |
| Frankfurt5 | Frankfurt | Sydney2 | Sydney |
| Hong Kong | Hong Kong | Taipei1 | Taipei |
| Houston | ***Dallas*** | Taipei3 | Taipei |
| Bangalore1 | Mumbai | Bangkok | Bangkok |
| Chennai | Mumbai | London3 | London |
| Hyderabad1 | Mumbai | London4 | London |
| Mumbai | Mumbai | Sao Paulo | Brazil |
| Milan2 | Milan | Quebec | ***Toronto / Montreal*** |
| Tokyo1 | Tokyo | Dallas | Dallas |
| Tokyo3 | Tokyo | Denver | Denver |
| Tokyo5 | Tokyo | Seattle | Seattle |
| Los Angeles | Los Angeles | Madrid | Madrid |

Table 9. The ground truth of Twitch I ngest server locations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest: Toronto |  | Shortest Ping |  | GeoPing |  | CBG |  | Shortest Dist |
| Rank Diff. | VPN | RTT (ms) | VPN | Eu\_Dist | VPN | Covers (circles) | VPN | Estimated Dist­ (km) |
| 0 | ***Toronto*** | ***9.619*** | Chicago | 223.660 | ***Toronto*** | ***6*** | Chicago | 703.764 |
| 1 | Chicago | 9.901 | Saint-Louis | 233.070 | Chicago | 5 | ***Toronto*** | ***795.801*** |
| 2 | Montreal | 11.994 | Miami | 240.357 | Manassas | 5 | Montreal | 855.075 |
| 3 | New York | 13.532 | Manassas | 248.048 | Buffalo | 5 | New York | 1170.576 |
| 4 | Manassas | 16.844 | Montreal | 250.196 | Montreal | 4 | Saint-Louis | 1285.269 |
| 5 | Saint-Louis | 17.364 | Buffalo | 254.543 | New York | 4 | Manassas | 1393.594 |
| 6 | Charlotte | 27.803 | ***Toronto*** | ***260.993*** | Saint-Louis | 3 | Charlotte | 2233.570 |
| 7 | Atlanta | 32.044 | Charlotte | 270.211 | Charlotte | 3 | Kansas City | 2554.209 |
| 8 | Dallas | 32.24 | New York | 275.409 | Kansas City | 2 | Dallas | 2771.323 |
| 9 | Kansas City | 33.203 | Atlanta | 279.674 | Atlanta | 2 | Miami | 3125.620 |

Table 10. Evaluation of rank difference on Toronto’s ingest server in each methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Shortest Ping | GeoPing | CBG | Shortest Dist |
| Ground Truth Location | Toronto | Toronto | Toronto | Toronto |
| Mapped VPN Location | Toronto | Chicago | Toronto | Chicago |
| Rank difference | 0 | 6 | 0 | 1 |
| Error Distance | 0 | 701­ | 0 | 701 |

Table 11. Evaluation of rank difference and error distance on Toronto’s ingest server

**6.2 Evaluation Parameters in Geolocation Methods**

Table 12, 13, 14, 15 show each geolocation method’s mapped VPN and the two evaluation parameters: error distance and rank difference.

**6.2.1 Shortest Ping**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Dallas | 1502 | 14 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Dallas | 1502 | 14 |
| Atlanta | Charlotte | 364 | 1 | Miami | Miami | 0 | 0 |
| Vienna2 | Bratislava | 55 | 8 | Amsterdam2 | Brussels | 173 | 2 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Brussels | 173 | 1 |
| Rio | Brazil | 0 | 0 | New York | New York | 0 | 0 |
| Toronto | Toronto | 0 | 0 | Norway Oslo | Oslo | 0 | 0 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Warsaw | 0 | 0 |
| Prague | Belgrade | 738 | 8 | Portland OR | Seattle | 0 | 0 |
| Copenhagen | Copenhagen | 0 | 0 | Salt Lake City | Salt Lake City | 0 | 0 |
| Helsinki | Helsinki | 0 | 0 | San Francisco | Los Angeles | 561 | 1 |
| Marseille | Marseille | 0 | 0 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Marseille | 0 | 0 | Singapore | Kuala Lumpur | 317 | 2 |
| Paris10 | Paris | 0 | 0 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Belgrade | 999 | 24 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Brussels | 0 | 0 | Stockholm | Helsinki | 396 | 1 |
| Frankfurt2 | Belgrade | 1061 | 31 | Sydney1 | Sydney | 0 | 0 |
| Frankfurt5 | Belgrade | 1061 | 30 | Sydney2 | Sydney | 0 | 0 |
| Hong Kong | Hong Kong | 0 | 0 | Taipei1 | Hong Kong | 811 | 3 |
| Houston | Dallas | 0 | 0 | Taipei3 | Hong Kong | 811 | 3 |
| Bangalore1 | Hong Kong | 4303 | 4 | Bangkok | Bangkok | 0 | 0 |
| Chennai | Mumbai | 0 | 0 | London3 | Brussels | 317 | 15 |
| Hyderabad1 | Mumbai | 0 | 0 | London4 | Brussels | 317 | 17 |
| Mumbai | Mumbai | 0 | 0 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Toronto | 0 | 0 |
| Tokyo1 | Tokyo | 0 | 0 | Dallas | Dallas | 0 | 0 |
| Tokyo3 | Tokyo | 0 | 0 | Denver | Salt Lake City | 596 | 10 |
| Tokyo5 | Tokyo | 0 | 0 | Seattle | Vancouver | 192 | 1 |
| Los Angeles | Los Angeles | 0 | 0 | Madrid | Madrid | 0 | 0 |

Table 12. Shortest Ping

**6.2.2 GeoPing**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Mexico | 0 | 0 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Mexico | 0 | 0 |
| Atlanta | Chicago | 945 | 3 | Miami | Miami | 0 | 0 |
| Vienna2 | Milan | 623 | 7 | Amsterdam2 | Amsterdam | 0 | 0 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Amsterdam | 0 | 0 |
| Rio | Brazil | 0 | 0 | New York | Manassas | 368 | 4 |
| Toronto | Chicago | 701 | 6 | Norway Oslo | Stockholm | 415 | 4 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Madrid | 2289 | 10 |
| Prague | Amsterdam | 710 | 19 | Portland OR | San Francisco | 1093 | 1 |
| Copenhagen | Stockholm | 522 | 2 | Salt Lake City | San Francisco | 968 | 4 |
| Helsinki | Stockholm | 396 | 1 | San Francisco | San Francisco | 0 | 0 |
| Marseille | Madrid | 815 | 8 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Paris | 663 | 9 | Singapore | Bangkok | 1433 | 6 |
| Paris10 | Dublin | 777 | 2 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Riga | 844 | 23 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Dublin | 757 | 1 | Stockholm | Stockholm | 0 | 0 |
| Frankfurt2 | Amsterdam | 361 | 5 | Sydney1 | Auckland | 2156 | 2 |
| Frankfurt5 | Amsterdam | 361 | 12 | Sydney2 | Auckland | 2156 | 2 |
| Hong Kong | Bangkok | 1725 | 2 | Taipei1 | Hanoi | 1663 | 1 |
| Houston | Kansas City | 730 | 1 | Taipei3 | Taipei | 0 | 0 |
| Bangalore1 | Hanoi | 3452 | 16 | Bangkok | Hanoi | 988 | 2 |
| Chennai | Bangkok | 3005 | 7 | London3 | Dublin | 463 | 3 |
| Hyderabad1 | Bangkok | 3005 | 5 | London4 | Amsterdam | 355 | 1 |
| Mumbai | Bangkok | 3005 | 1 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Miami | 1990 | 3 |
| Tokyo1 | Taipei | 2105 | 1 | Dallas | Kansas City | 730 | 1 |
| Tokyo3 | Taipei | 2105 | 2 | Denver | Denver | 0 | 0 |
| Tokyo5 | Taipei | 2105 | 1 | Seattle | Seattle | 0 | 0 |
| Los Angeles | San Francisco | 561 | 1 | Madrid | Milan | 1188 | 12 |

Table 13. GeoPing

**6.2.3 CBG**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Dallas | 1502 | 4 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Dallas | 1502 | 6 |
| Atlanta | Atlanta | 0 | 0 | Miami | Miami | 0 | 0 |
| Vienna2 | Belgrade | 489 | 2 | Amsterdam2 | Brussels | 173 | 1 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Brussels | 173 | 5 |
| Rio | Brazil | 0 | 0 | New York | New York | 0 | 0 |
| Toronto | Toronto | 0 | 0 | Norway Oslo | Oslo | 0 | 0 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Warsaw | 0 | 0 |
| Prague | Steinsel | 599 | 5 | Portland OR | Seattle | 0 | 0 |
| Copenhagen | Copenhagen | 0 | 0 | Salt Lake City | Salt Lake City | 0 | 0 |
| Helsinki | Oslo | 786 | 9 | San Francisco | San Francisco | 0 | 0 |
| Marseille | Marseille | 0 | 0 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Marseille | 0 | 0 | Singapore | Kuala Lumpur | 317 | 2 |
| Paris10 | Zurich | 490 | 2 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Copenhagen | 354 | 3 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Amsterdam | 0 | 0 | Stockholm | Oslo | 415 | 1 |
| Frankfurt2 | Steinsel | 189 | 1 | Sydney1 | Sydney | 0 | 0 |
| Frankfurt5 | Berlin | 422 | 8 | Sydney2 | Sydney | 0 | 0 |
| Hong Kong | Hong Kong | 0 | 0 | Taipei1 | Hong Kong | 811 | 1 |
| Houston | Dallas | 0 | 0 | Taipei3 | Hong Kong | 811 | 1 |
| Bangalore1 | Mumbai | 0 | 0 | Bangkok | Bangkok | 0 | 0 |
| Chennai | Mumbai | 0 | 0 | London3 | Brussels | 317 | 9 |
| Hyderabad1 | Mumbai | 0 | 0 | London4 | Zurich | 775 | 11 |
| Mumbai | Mumbai | 0 | 0 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Toronto | 0 | 0 |
| Tokyo1 | Tokyo | 0 | 0 | Dallas | Dallas | 0 | 0 |
| Tokyo3 | Tokyo | 0 | 0 | Denver | Dallas | 1065 | 5 |
| Tokyo5 | Tokyo | 0 | 0 | Seattle | Vancouver | 192 | 1 |
| Los Angeles | Los Angeles | 0 | 0 | Madrid | Paris | 1053 | 1 |

Table 14. CBG

**6.2.4 Shortest Distance**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Dallas | 1502 | 6 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Dallas | 1502 | 6 |
| Atlanta | Charlotte | 364 | 1 | Miami | Miami | 0 | 0 |
| Vienna2 | Bratislava | 55 | 7 | Amsterdam2 | Brussels | 173 | 1 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Brussels | 173 | 1 |
| Rio | Brazil | 0 | 0 | New York | New York | 0 | 0 |
| Toronto | Chicago | 701 | 1 | Norway Oslo | Oslo | 0 | 0 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Warsaw | 0 | 0 |
| Prague | Belgrade | 738 | 13 | Portland OR | Seattle | 0 | 0 |
| Copenhagen | Copenhagen | 0 | 0 | Salt Lake City | Salt Lake City | 0 | 0 |
| Helsinki | Helsinki | 0 | 0 | San Francisco | San Francisco | 0 | 0 |
| Marseille | Marseille | 0 | 0 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Marseille | 0 | 0 | Singapore | Kuala Lumpur | 317 | 2 |
| Paris10 | Paris | 0 | 0 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Belgrade | 999 | 20 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Brussels | 0 | 0 | Stockholm | Helsinki | 396 | 2 |
| Frankfurt2 | Belgrade | 1061 | 34 | Sydney1 | Sydney | 0 | 0 |
| Frankfurt5 | Belgrade | 1061 | 33 | Sydney2 | Sydney | 0 | 0 |
| Hong Kong | Hong Kong | 0 | 0 | Taipei1 | Hong Kong | 811 | 1 |
| Houston | Dallas | 0 | 0 | Taipei3 | Hong Kong | 811 | 1 |
| Bangalore1 | Mumbai | 0 | 0 | Bangkok | Bangkok | 0 | 0 |
| Chennai | Mumbai | 0 | 0 | London3 | Brussels | 317 | 22 |
| Hyderabad1 | Mumbai | 0 | 0 | London4 | Brussels | 317 | 24 |
| Mumbai | Mumbai | 0 | 0 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Toronto | 0 | 0 |
| Tokyo1 | Tokyo | 0 | 0 | Dallas | Dallas | 0 | 0 |
| Tokyo3 | Tokyo | 0 | 0 | Denver | Salt Lake City | 596 | 10 |
| Tokyo5 | Tokyo | 0 | 0 | Seattle | Vancouver | 192 | 1 |
| Los Angeles | Los Angeles | 0 | 0 | Madrid | Paris | 1053 | 1 |

Table 15. Shortest Distance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rank Diff. | Shortest Ping | GeoPing | Shortest Dist | CBG |
| 0 | 40 | 22 | 40 | 40 |
| 1 | 5 | 11 | 8 | 7 |
| 2 | 2 | 7 | 2 | 3 |
| 3 | 2 | 3 | 0 | 1 |
| 4 | 1 | 3 | 0 | 1 |
| 5 | 0 | 2 | 0 | 3 |
| 6 | 0 | 2 | 2 | 1 |
| 7 | 0 | 2 | 1 | 0 |
| 8 | 2 | 1 | 0 | 1 |
| 9 | 0 | 1 | 0 | 2 |
| ≥ 10 | 8 | 6 | 7 | 1 |

Table 16. The number of mapping in different rank difference of each method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Error Distance | Shortest Ping | GeoPing | Shortest Dist | CBG |
| 0 km (right location) | 40 | 22 | 40 | 40 |
| 50~100 | 1 | 0 | 1 | 0 |
| 100~200 | 3 | 0 | 3 | 4 |
| 200~300 | 0 | 0 | 0 | 0 |
| 300~400 | 5 | 5 | 5 | 3 |
| 400~500 | 0 | 2 | 0 | 4 |
| 500~600 | 2 | 2 | 1 | 1 |
| 600~700 | 0 | 2 | 0 | 0 |
| 700~800 | 1 | 6 | 2 | 2 |
| 800~900 | 2 | 2 | 2 | 2 |
| 900~1000 | 1 | 3 | 1 | 0 |
| 1000~1500 | 2 | 3 | 3 | 2 |
| 1500~2000 | 2 | 3 | 2 | 2 |
| 2000~3000 | 0 | 6 | 0 | 0 |
| > 3000 km | 1 | 4 | 0 | 0 |

Table 17. The number of mapping in different error distance range of each method

Table 16 summarizes the rank difference of each method from table 12 to table 15.

We can observe that the GeoPing maps the least target to the correct locations (22 locations on rank difference 0), and other methods hit 40 correct locations. Besides, we can randomly give a threshold of rank difference, such as top 3 (rank diff. 0 to rank diff. 2). In top 3, there are 47, 40, 50, 50 mapping in Shortest Ping, GeoPing, Shortest Distance, CBG respectively. It represents that Shortest Distance and CBG are best in top 3 rank difference because they have the most the most ground truth in top 3 mapping priority.

Table 17 summarizes the error distance of each method from table 12 to table 15. We can randomly give a threshold of the error distance. For example, we take 500 km to be the threshold, if error distance larger than 500 km, then the mapping is not good. We can observe that CBG has 51 locations, and both Shortest Ping and Shortest Distance have 49 locations. It represents that CBG is the best in the region of 0 to 500 km, because there are the most mapped locations in the CBG are no more than 500 km. In addition, the error distance of one mapped location > 3000 km in Shortest Ping. It maps Bangalore to Hong Kong the distance is 4303 km. However, the error distance of the worst case in CBG and Shortest Distance no more than 2000 km.

**6.3 Evaluation Result**

**6.3.1 Evaluation on Rank Difference**

Table 18 shows the cumulative probability and Figure 29 shows the CDF of rank difference in each geolocation method. From the Table 18 and Figure 29, we can easily observe. Nonetheless, if we only pick 0 rank difference for mapping, the 66% accuracy is not reliable. Hence we should pick more possible VPN locations for mapping. For example, we can pick top 3, top 5, or even top 10 rank difference for evaluation. However, we should not take all the rank difference into consideration because a too large rank difference means a lot of VPNs locations have a higher ranking than the ground truth location. It leads to numerous and widely distributed VPN locations to be possible candidates for mapping and become useless because too many candidate are not helpful for filtering. Thus we still pay more attention to the cumulative probability value of smaller rank difference, for example, 0 rank difference is more important than 1 rank difference, and so on. To sum up, we have to make compromises between filtering capability and the proportion of ingest servers included since the stronger the filtering capability, the less ingest server proportion is included, and vice versa. For example, if we pick top 3 rank difference in CBG. There will be 83% ground truth locations can be matched to one of the VPN location in top 3 mapping candidates. Besides, if we pick top 10 rank difference in CBG, there will be 98% ground truth location can be matched to one of top 10 VPN candidates. In the future, from the experimental result, researchers can pick top 3, top 5, top 10, or other value as possible mapping candidates as preliminary filtering.

Next, we evaluate the proportion of each geolocation method in the top 1 top 10 rank difference. First, GeoPing’s performance is awful in top 0 to top 6 rank difference, while Shortest Ping only lose to GeoPing in top 8 to top 10 rank difference, but Shortest Ping beats GeoPing other rank difference, and it has the most proportion in 0 rank difference among these methods. Thus Shortest Ping can be considered as a better method than GeoPing on rank difference metric. On the other hand, Shortest Ping perform cannot beat Shortest Distance in all region. So, Shortest Distance is better than Shortest Ping on rank difference metric. Moreover, CBG slightly loses to Shortest Distance in top 1 rank difference, but it performs the best in other rank difference. In summary, the performance ranking on rank difference is CBG, Shortest Distance, Shortest Ping, and GeoPing.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Diff. | Shortest Ping | | GeoPing | | Shortest Dist | | CBG |
| 0 | 66 | | 36 | | 66 | | 66 |
| 1 | 75 | | 55 | | 80 | | 78 |
| 2 | 78 | | 66 | | 83 | | 83 |
| 3 | 81 | | 71 | | 83 | | 85 |
| 4 | 83 | | 76 | | 83 | | 86 |
| 5 | 83 | | 80 | | 83 | | 91 |
| 6 | 83 | | 83 | | 86 | | 93 |
| 7 | 83 | | 86 | | 88 | | 93 |
| 8 | 86 | | 88 | | 88 | | 95 |
| 9 | 86 | | 90 | | 88 | | 98 |
|  |  |  | |  | |  | |

Table 18. Cumulative probability of rank difference (in %)

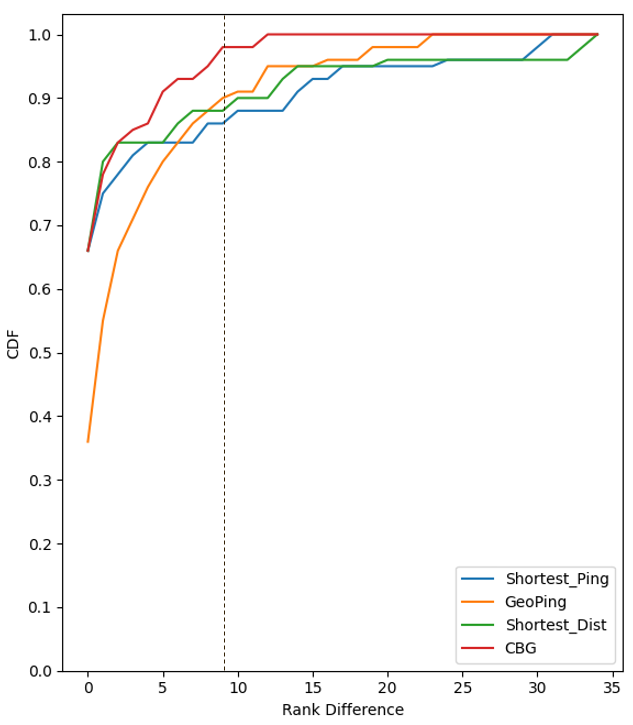


Figure 29. CDF of rank difference

**6.3.2 Evaluation on Error Distance**

Table 19 shows the cumulative probability and Figure 30 shows the CDF of error distance in each geolocation method. On error distance, unlike the rank difference, the large error distance are important, too. Because large error distance means distance between the mapped VPN location and the ingest server’s ground truth location is large. This is a very serious situation that cannot be ignored because mapping a target far away can cause many problems. Besides, about total error distance, if total error distance is small, it means the average error distance of each mapping is very small as well. And it also means the geolocation method has a higher error tolerance rate when doing mapping. Therefore, total error distance can be seen as the most important criterion for evaluating the performance of the measurement geolocation methods.

In Table 19 and Figure 30, we can find that GeoPing performs terribly in all situations. And total error distance in GeoPing is very large. Besides, there are only 36% error distance less than 300 km, up to 17% error distance are larger than 2000 km, and 7% error distance are larger than 3000 km. These reasons lead to a very low fault tolerance rate of GeoPing when mapping. On the other hand, we can easily observe that CBG performs best among them. There are 85% error distance less than 500 km. And there are only 4% mapped VPNs with the largest error distance in CBG and these error distance are less than 2000 km. Besides, the most important total error distance in CBG is 12435 km and average error distance is 207.25 km which is the smallest among all the geolocation methods. In addition, the performances of Shortest Ping and Shortest Distance seem to be similar, but in the maximum error distance between mapped VPN and ground truth location, Shortest Ping has an error distance larger than 3000km, while the maximum error distance in Shortest Distance is smaller than 2000 km. Furthermore, the total error distance in Shortest Distance is relatively small. In conclusion, the performance rankings on rank difference are CBG, Shortest Distance, Shortest Ping, and GeoPing.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Error Distance | Shortest Ping | | GeoPing | | Shortest Dist | | CBG |
| 0 km | 66 | | 36 | | 66 | | 66 |
| 50~100 | 68 | | 36 | | 68 | | 66 |
| 100~200 | 73 | | 36 | | 73 | | 73 |
| 200~300 | 73 | | 36 | | 73 | | 73 |
| 300~400 | 81 | | 45 | | 81 | | 78 |
| 400~500 | 81 | | 48 | | 81 | | 85 |
| 500~600 | 85 | | 51 | | 83 | | 86 |
| 600~700 | 85 | | 55 | | 83 | | 86 |
| 700~800 | 86 | | 65 | | 86 | | 90 |
| 800~900 | 90 | | 68 | | 90 | | 93 |
| 900~1000 | 91 | | 73 | | 91 | | 93 |
| 1000~1500 | 95 | | 78 | | 96 | | 96 |
| 1500~2000 | 98 | | 83 | | 100 | | 100 |
| 2000~3000 | 98 | | 93 | | 100 | | 100 |
| > 3000 km | 100 | | 100 | | 100 | | 100 |
| **Total err(km)** | 16249 | | 48528 | | 13139 | | **12435** |
|  |  |  | |  | |  | |

Table 19. Cumulative probability of error distance (in %)

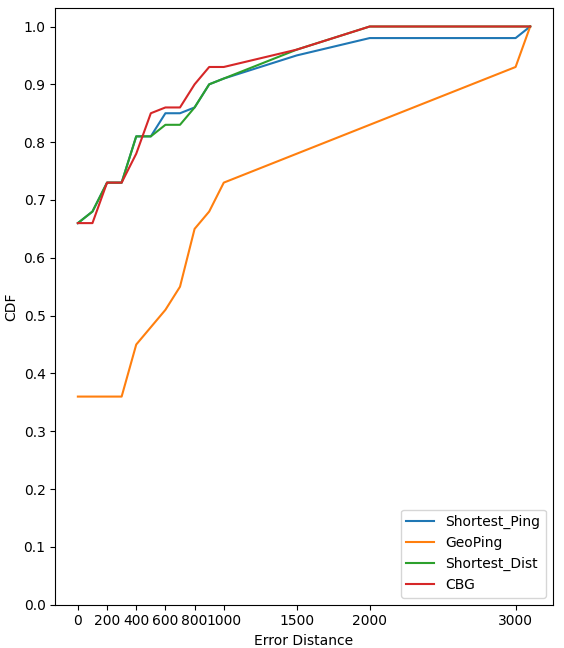
****

Figure 30. CDF of error distance

**6.3.3 Analysis of Evaluation Result**

After the evaluation on rank difference and error distance, we can roughly conclude that the performance rankings of these geolocation methods in our measurement are CBG, Shortest Distance, Shortest Ping, GeoPing. First, Shortest Ping is the most convenient to implement. But the performance is not very good. It only performs well in the top rank (rank #1). Nevertheless, if Shortest Ping fails to map the target to the correct ground truth location, there will be a risk that it may map the target to the VPN location with error distance larger than 3000 km. It still be a serious penalty on error distance metric. Besides, Shortest Distance performs the best when we pick top 3 VPN locations for candidates, while its performance is ordinary in other rank difference. In addition, Shortest Distance perform well on error distance less than 400 km, and its maximum error distance is not very large (less than 2000 km). Moreover, CBG perform the best among these methods in both metrics. It can pick from top3 to top 10 rank difference VPN locations as candidates, and the coverage of ground truth location will be 83% to 98% which is the best among these methods. Besides, CBG have the minimum total error distance and the maximum error distance will not be too high.

However, the performance of GeoPing is very terrible. The reason is that GeoPing uses Euclidean distance for mapping. In this case, as long as there is a measured RTT, whether this RTT is greater or less than a normal measured RTT, the RTT difference will be greatly enlarged by square. It leads to a large fluctuation in Euclidean distance. For example, Euclidean distance function is . Because we only compare relative sizes, thus we can ignore the SQRT, so the new function will be. As long as there is one RTT difference being large, it can cause a massive increase in the overall sum. For example in Figure 31, if the target fingerprint is [20, 30, 50, 100], LM1’s fingerprint is [22, 35, 60, 190] and LM2’s fingerprint is [60, 80, 10, 50]. And LM1 is the ground truth of target location. However, the value generated from new function will be 8229 in LM1 and 8200 in LM 2. Thus GeoPing will map the T to LM2 despite their extremely dissimilar fingerprints. It means that GeoPing is very low tolerance for unknown noise, waiting time or other known overhead with delay time which is hard to estimate. The above reasons may lead to confusion in GeoPing geolocalization even if the additional delay time is small.

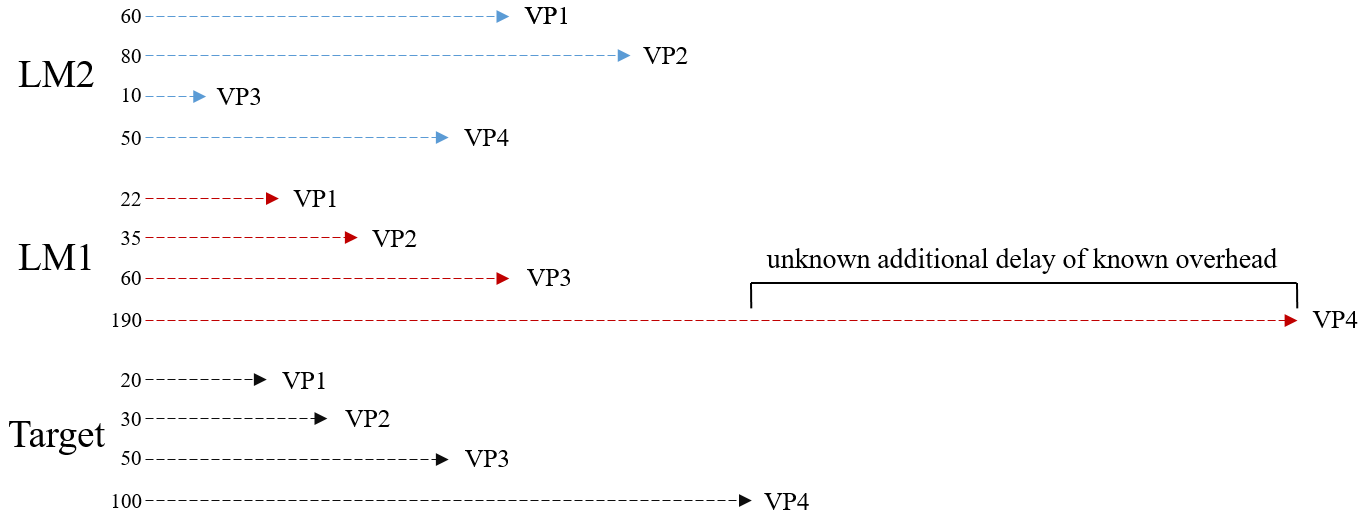


Figure 31. A common case that will make GeoPing wrong mapping

On the other hand, in the evaluation, CBG and Shortest Distance perform well among these geolocation methods. Therefore, we are curious about the performance of the Geolocalization if CBG and Shortest Distance are combined.

**Chapter 7**

**Evaluate CBG - Shortest Distance Method**

**7.1 Introduction of CBG - Shortest Distance Method**

The difference between CBG - Shortest Distance method and Pure CBG method is only when there are more than one VPN’s circle covers are the same and maximum, CBG checks which VPN’s estimated distance to the ingest server is smallest (by Shortest Distance method). It means only when CBG determines that multiple VPNs are equally good, they will be handed over to the Shortest Distance method to judge which VPN to map to. However, we can generalize this situation in CBG - Shortest Distance Method. We rank VPNs according to the number of covers from high to low. The higher the ranking, the higher the point\_cover. In a similar way, according to the estimated distance from small to large, and the VPN ranking is made from high to low. Equally, the higher the ranking, the higher the point\_dist. In order to measure the performance of CBG - Shortest Distance method, we randomly give two scaled scores forCBG - Shortest Distance 1 and CBG - Shortest Distance 2 for evaluation shown in Table 20. After that, we calculate the total point equal to point\_cover plus point\_dist for each VPN and locate ingest server at the VPN which has highest total point. If there are multiple VPNs with the highest point, then the CBG - Shortest Distance method will randomly map target to one of these VPNs. For example, Table 21 shows top 5 rank difference in CBG - Shortest Distance method when ingest server is Toronto. Toronto VPN and Chicago VPN have the same highest point. Therefore, this method map the Toronto ingest server to one of the two VPNs randomly, and it luckily maps the ingest server to Toronto VPN in this example.

**CBG - Shortest Distance 1 CBG - Shortest Distance 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rank | point\_cover | point\_dist |  | Rank | point\_cover | point\_dist |
| 1 | 100 | 100 |  | 1 | 100 | 100 |
| 2 | 90 | 90 |  | 2 | 95 | 90 |
| 3 | 80 | 80 |  | 3 | 90 | 80 |
| 4 | 77 | 77 |  | 4 | 87 | 77 |
| 5 | 74 | 74 |  | 5 | 84 | 74 |
| 6 | 71 | 71 |  | 6 | 81 | 71 |
| 7 | 69 | 69 |  | 7 | 79 | 69 |
| 8 | 67 | 67 |  | 8 | 77 | 67 |
| 9 | 65 | 65 |  | 9 | 75 | 65 |
| 10 | 62 | 62 |  | 10 | 72 | 62 |
| 11 | 61 | 61 |  | 11 | 71 | 61 |
| 12 | 60 | 60 |  | 12 | 70 | 60 |
| 13 | 59 | 59 |  | 13 | 69 | 59 |
| 14 | 58 | 58 |  | 14 | 68 | 58 |
| 15 | 57 | 57 |  | 15 | 67 | 57 |
| 16 | 56 | 56 |  | 16 | 66 | 56 |
| 17 | 55 | 55 |  | 17 | 65 | 55 |
| 18 | 54 | 54 |  | 18 | 64 | 54 |
| 19 | 53 | 53 |  | 19 | 63 | 53 |
| 20 | 52 | 52 |  | 20 | 62 | 52 |
| 21 | 51 | 51 |  | 21 | 61 | 51 |
| 22 | 50 | 50 |  | 22 | 60 | 50 |
| 23 | 49 | 49 |  | 23 | 59 | 49 |
| 24 | 48 | 48 |  | 24 | 58 | 48 |
| 25 | 47 | 47 |  | 25 | 57 | 47 |
| 26 | 46 | 46 |  | 26 | 56 | 46 |
| 27 | 45 | 45 |  | 27 | 55 | 45 |
| 28 | 44 | 44 |  | 28 | 54 | 44 |
| 29 | 43 | 43 |  | 29 | 53 | 43 |
| 30 | 42 | 42 |  | 30 | 52 | 42 |
| > 30 | 0 | 0 |  | > 30 | 0 | 0 |

Table 20. Scaled score of CBG - Shortest Distance method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Ingest:Toronto |  |  |  |  |  |
| Mapping Priority | VPN | Cover Rank | Short Dist rank | Point\_cover | Point\_dist | Total Point |
| 1 | ***Toronto*** | 1 | 2 | 100 | 90 | 190 |
| 2 | Chicago | 2 | 1 | 90 | 100 | 190 |
| 3 | Manassas | 2 | 6 | 90 | 71 | 161 |
| 4 | Montreal | 5 | 3 | 74 | 80 | 154 |
| 5 | New York | 5 | 4 | 74 | 77 | 151 |

Table 21. Rank difference on Toronto ingest serverin CBG - Shortest Distance1 method

**7.2 Evaluation Parameters in CBG - Shortest Distance Method**

In CBG - Shortest Distance Method,we map the Twitch ingest server to the VPN with the highest total score of point\_cover plus point\_dist according to the scaled score we randomly give in Table 20. Finally, Table 22 and Table 23 show the mapping results and the two evaluation parameters (error distance and rank difference) of CBG-Shortest Distance Method 1 and CBG Shortest Distance Method 2.

**CBG - Shortest Distance 1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Dallas | 1502 | 4 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Dallas | 1502 | 6 |
| Atlanta | Atlanta | 0 | 0 | Miami | Miami | 0 | 0 |
| Vienna2 | Belgrade | 489 | 3 | Amsterdam2 | Brussels | 173 | 1 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Brussels | 173 | 2 |
| Rio | Brazil | 0 | 0 | New York | New York | 0 | 0 |
| Toronto | Toronto | 0 | 0 | Norway Oslo | Oslo | 0 | 0 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Warsaw | 0 | 0 |
| Prague | Brussels | 721 | 6 | Portland OR | Seattle | 0 | 0 |
| Copenhagen | Copenhagen | 0 | 0 | Salt Lake City | Salt Lake City | 0 | 0 |
| Helsinki | Oslo | 786 | 5 | San Francisco | San Francisco | 0 | 0 |
| Marseille | Marseille | 0 | 0 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Marseille | 0 | 0 | Singapore | Kuala Lumpur | 317 | 2 |
| Paris10 | Paris | 0 | 0 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Copenhagen | 354 | 4 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Brussels | 0 | 0 | Stockholm | Oslo | 415 | 1 |
| Frankfurt2 | Zurich | 305 | 21 | Sydney1 | Sydney | 0 | 0 |
| Frankfurt5 | Zurich | 305 | 24 | Sydney2 | Sydney | 0 | 0 |
| Hong Kong | Hong Kong | 0 | 0 | Taipei1 | Hong Kong | 811 | 1 |
| Houston | Dallas | 0 | 0 | Taipei3 | Hong Kong | 811 | 1 |
| Bangalore1 | Mumbai | 0 | 0 | Bangkok | Bangkok | 0 | 0 |
| Chennai | Mumbai | 0 | 0 | London3 | Brussels | 317 | 12 |
| Hyderabad1 | Mumbai | 0 | 0 | London4 | Brussels | 317 | 13 |
| Mumbai | Mumbai | 0 | 0 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Toronto | 0 | 0 |
| Tokyo1 | Tokyo | 0 | 0 | Dallas | Dallas | 0 | 0 |
| Tokyo3 | Tokyo | 0 | 0 | Denver | Dallas | 1065 | 5 |
| Tokyo5 | Tokyo | 0 | 0 | Seattle | Vancouver | 192 | 1 |
| Los Angeles | Los Angeles | 0 | 0 | Madrid | Paris | 1053 | 1 |

Table 22. CBG - Shortest Distance 1

**CBG - Shortest Distance 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ingest Server | Map to VPN | Error Distance | Rank Diff. | Ingest Server | Map to VPN | Error Distance | Rank Diff. |
| Ashburn3 | Manassas | 0 | 0 | Queretaro1 | Dallas | 1502 | 4 |
| Ashburn5 | Manassas | 0 | 0 | Queretaro2 | Dallas | 1502 | 6 |
| Atlanta | Charlotte | 364 | 1 | Miami | Miami | 0 | 0 |
| Vienna2 | Bratislava | 55 | 3 | Amsterdam2 | Brussels | 173 | 1 |
| Fortaleza1 | Brazil | 0 | 0 | Amsterdam3 | Brussels | 173 | 1 |
| Rio | Brazil | 0 | 0 | New York | New York | 0 | 0 |
| Toronto | Chicago | 701 | 1 | Norway Oslo | Oslo | 0 | 0 |
| Chicago IL2 | Chicago | 0 | 0 | Phoenix AZ | Phoenix | 0 | 0 |
| Chicago IL3 | Chicago | 0 | 0 | Warsaw | Warsaw | 0 | 0 |
| Prague | Brussels | 721 | 6 | Portland OR | Seattle | 0 | 0 |
| Copenhagen | Copenhagen | 0 | 0 | Salt Lake City | Salt Lake City | 0 | 0 |
| Helsinki | Oslo | 786 | 3 | San Francisco | San Francisco | 0 | 0 |
| Marseille | Marseille | 0 | 0 | San Jose | San Francisco | 0 | 0 |
| Marseille2 | Marseille | 0 | 0 | Singapore | Kuala Lumpur | 317 | 2 |
| Paris10 | Paris | 0 | 0 | Seoul1 | Seoul | 0 | 0 |
| Berlin | Copenhagen | 354 | 4 | Seoul3 | Seoul | 0 | 0 |
| Düsseldorf1 | Brussels | 0 | 0 | Stockholm | Oslo | 415 | 1 |
| Frankfurt2 | Zurich | 305 | 23 | Sydney1 | Sydney | 0 | 0 |
| Frankfurt5 | Zurich | 305 | 24 | Sydney2 | Sydney | 0 | 0 |
| Hong Kong | Hong Kong | 0 | 0 | Taipei1 | Hong Kong | 811 | 1 |
| Houston | Dallas | 0 | 0 | Taipei3 | Hong Kong | 811 | 1 |
| Bangalore1 | Mumbai | 0 | 0 | Bangkok | Bangkok | 0 | 0 |
| Chennai | Mumbai | 0 | 0 | London3 | Brussels | 317 | 12 |
| Hyderabad1 | Mumbai | 0 | 0 | London4 | Brussels | 317 | 13 |
| Mumbai | Mumbai | 0 | 0 | Sao Paulo | Brazil | 0 | 0 |
| Milan2 | Milan | 0 | 0 | Quebec | Toronto | 0 | 0 |
| Tokyo1 | Tokyo | 0 | 0 | Dallas | Dallas | 0 | 0 |
| Tokyo3 | Tokyo | 0 | 0 | Denver | Dallas | 1065 | 6 |
| Tokyo5 | Tokyo | 0 | 0 | Seattle | Vancouver | 192 | 1 |
| Los Angeles | Los Angeles | 0 | 0 | Madrid | Paris | 1053 | 1 |

Table 23. CBG - Shortest Distance 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Error Distance | Shortest Ping | GeoPing | Shortest Dist | CBG | Hybrid 1 | Hybrid 2 |
| 0 km (right location) | 40 | 22 | 40 | 40 | 41 | 39 |
| 50~100 | 1 | 0 | 1 | 0 | 0 | 1 |
| 100~200 | 3 | 0 | 3 | 4 | 3 | 3 |
| 200~300 | 0 | 0 | 0 | 0 | 0 | 0 |
| 300~400 | 5 | 5 | 5 | 3 | 6 | 7 |
| 400~500 | 0 | 2 | 0 | 4 | 2 | 1 |
| 500~600 | 2 | 2 | 1 | 1 | 0 | 0 |
| 600~700 | 0 | 2 | 0 | 0 | 0 | 0 |
| 700~800 | 1 | 6 | 2 | 2 | 2 | 3 |
| 800~900 | 2 | 2 | 2 | 2 | 2 | 2 |
| 900~1000 | 1 | 3 | 1 | 0 | 0 | 0 |
| 1000~1500 | 2 | 3 | 3 | 2 | 2 | 2 |
| 1500~2000 | 2 | 3 | 2 | 2 | 2 | 2 |
| 2000~3000 | 0 | 6 | 0 | 0 | 0 | 0 |
| > 3000 km | 1 | 4 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rank Diff. | Shortest Ping | GeoPing | Shortest Dist | CBG | Hybrid 1 | Hybrid 2 |
| 0 | 40 | 22 | 40 | 40 | 41 | 39 |
| 1 | 5 | 11 | 8 | 7 | 6 | 9 |
| 2 | 2 | 7 | 2 | 3 | 2 | 1 |
| 3 | 2 | 3 | 0 | 1 | 1 | 2 |
| 4 | 1 | 3 | 0 | 1 | 2 | 2 |
| 5 | 0 | 2 | 0 | 3 | 2 | 0 |
| 6 | 0 | 2 | 2 | 1 | 2 | 3 |
| 7 | 0 | 2 | 1 | 0 | 0 | 0 |
| 8 | 2 | 1 | 0 | 1 | 0 | 0 |
| 9 | 0 | 1 | 0 | 2 | 0 | 0 |
| ≥ 10 | 8 | 6 | 7 | 1 | 4 | 4 |

Table 24. The number of mapping in different rank difference of each method

Table 25. The number of mapping in different error distance range of each method

Table 24 summarizes the rank difference of each method in table 22 and table 23. We call CBG-Shortest Distance1 to be Hybrid1 and call CBG-Shortest Distance1 to be Hybrid2. Hybrid1 hit the most ground truth (41 mappings) and Hybrid2 hit 39 ground truth. In addition, if we take the top 5 rank difference as the threshold, Hybrid 2 has the most 53 mappings in the top 5 rank difference.

Table 25 summarizes the error distance of each method in table 22 and table 23. If we set 500 km to be the threshold, Hybrid1 has the most 52 locations with error distance < 500km. It seems that the performance of hybrid method is not bad.

**7.3 Evaluation Result of CBG - Shortest Distance method**

Table 26 and Figure 32 shows the rank difference of these geolocation methods. Hybrid1 performs the best in 0 rank difference, but it is not better than CBG in other rank difference. Besides, Hybrid2 is better than CBG only in top 2 and top 5 rank difference but not better in others. Therefore, on rank difference, Hybrid method is not better than CBG.

However, on error distance metric, Hybrid1 performs the best in all region. And its total error distance is the smallest among these geolocation methods, too. While Hybrid2 is only better than CBG in 300 to 400 km error distance, but it wins by up to 5 percentage points. And most important, its total error distance is slightly smaller than CBG’s total error distance. In summary, on error distance metric, the performance of Hybrid method can beat all the other method.

Furthermore, because the rank difference metric in Hybrid method is based on the respective rank of the ground truth location in CBG and Shortest Distance. As a result, the calculated rank in CBG-Shortest Distance must fall between CBG and Shortest Distance. Thus, the performance of rank difference metric will fall between CBG and Shortest Distance. However, unlike rank difference metric, error distance metric only takes the first mapping priority location into consideration. This thesis maps each ingest server to a VPN location with the highest rank. And there will be a special case, for example, CBG map Vienna ingest server to VPN Steinsel in Luxembourg, and Shortest Distance map it to Belgrade, but the Hybrid method map it to Brussels in Belgium. Consequently, there are three different error distances which are different from each other. It means the error distance in Hybrid method may not be in between the error distance of CBG and error distance of Shortest Distance.

In addition, one advantage of Hybrid method is that if a rank difference is too low in one of the two methods of CBG or Shortest Distance, this problem can be eliminated after combining. Because the rank difference will fall between CBG and Shortest Distance. So, Hybrid method may be a more stable approach on rank difference metric. Furthermore, on error distance metric, unless the error in the number of covers in CBG and the error of estimated distance in Shortest Distance are both large, the two parameters (number of covers and estimated distance) can complement each other. Because the two parameters are related to the actual geolocation, rather than the network distance such as RTT. If the error in one of the two parameters is larger, the other one can give it a lower score to prevent it to be the highest rank. As a result, the final mapped VPN location will be highly rated by both the two parameters (a lot of covers and short estimated distance). Besides, the error distance metric is related to geolocation. Therefore, the error distance in the Hybrid method will be small due to the double appreciated by both number of covers and estimated distance as the evaluation result shown in Table 27 and Figure 33.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank Diff. | Shortest Ping | | GeoPing | | Shortest Dist | | CBG | | Hybrid 1 | Hybrid 2 |
| 0 | 66 | | 36 | | 66 | | 66 | | 68 | 65 |
| 1 | 75 | | 55 | | 80 | | 78 | | 78 | 80 |
| 2 | 78 | | 66 | | 83 | | 83 | | 81 | 81 |
| 3 | 81 | | 71 | | 83 | | 85 | | 83 | 85 |
| 4 | 83 | | 76 | | 83 | | 86 | | 86 | 88 |
| 5 | 83 | | 80 | | 83 | | 91 | | 90 | 88 |
| 6 | 83 | | 83 | | 86 | | 93 | | 93 | 93 |
| 7 | 83 | | 86 | | 88 | | 93 | | 93 | 93 |
| 8 | 86 | | 88 | | 88 | | 95 | | 93 | 93 |
| 9 | 86 | | 90 | | 88 | | 98 | | 93 | 93 |
|  |  |  | |  | |  | |

Table 26. Cumulative probability of rank difference (in %)

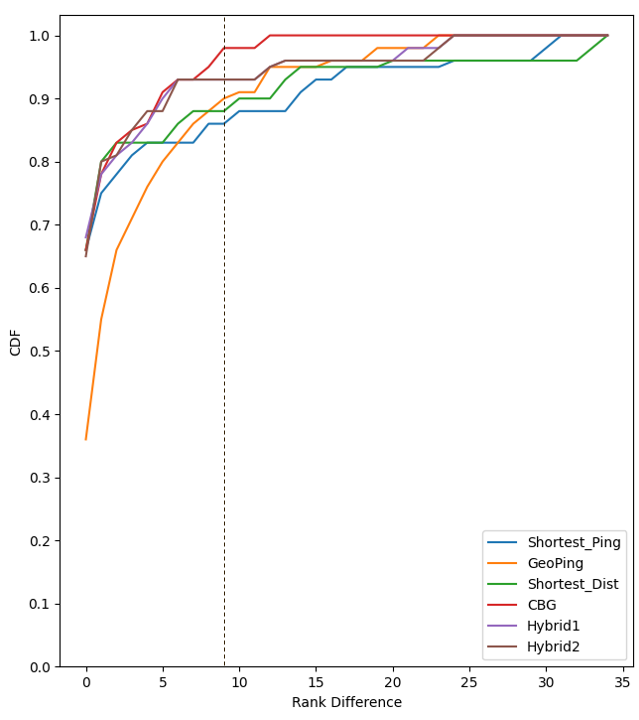
****

Figure 32. CDF of rank difference

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Error Distance | Shortest Ping | GeoPing | Shortest Dist | CBG | Hybrid 1 | Hybrid 2 |
| 0 km | 66 | 36 | 66 | 66 | 68 | 65 |
| 50~100 | 68 | 36 | 68 | 66 | 68 | 66 |
| 100~200 | 73 | 36 | 73 | 73 | 73 | 71 |
| 200~300 | 73 | 36 | 73 | 73 | 73 | 71 |
| 300~400 | 81 | 45 | 81 | 78 | 83 | 83 |
| 400~500 | 81 | 48 | 81 | 85 | 86 | 85 |
| 500~600 | 85 | 51 | 83 | 86 | 86 | 85 |
| 600~700 | 85 | 55 | 83 | 86 | 86 | 85 |
| 700~800 | 86 | 65 | 86 | 90 | 90 | 90 |
| 800~900 | 90 | 68 | 90 | 93 | 93 | 93 |
| 900~1000 | 91 | 73 | 91 | 93 | 93 | 93 |
| 1000~1500 | 95 | 78 | 96 | 96 | 96 | 96 |
| 1500~2000 | 98 | 83 | 100 | 100 | 100 | 100 |
| 2000~3000 | 98 | 93 | 100 | 100 | 100 | 100 |
| > 3000 km | 100 | 100 | 100 | 100 | 100 | 100 |
| **Total err(km)** | 16249 | 48528 | 13139 | 12435 | **11608** | 12239 |

Table 27. Cumulative probability of error distance (in %)

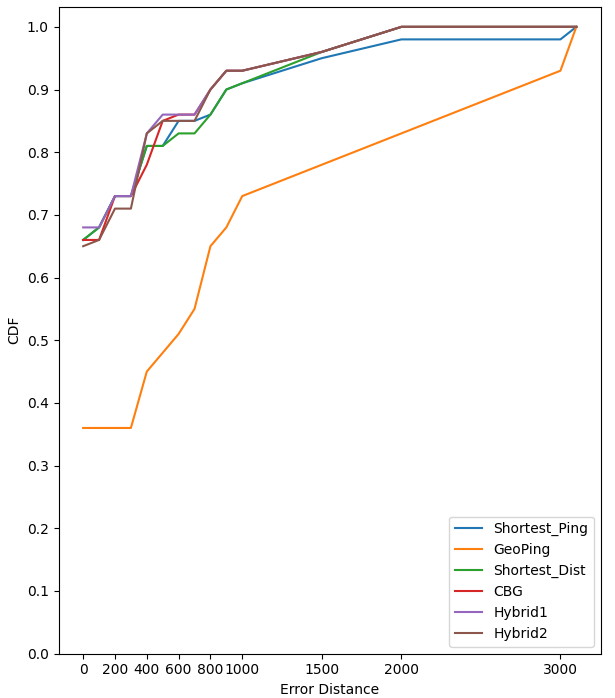
****

Figure 33. CDF of error distance

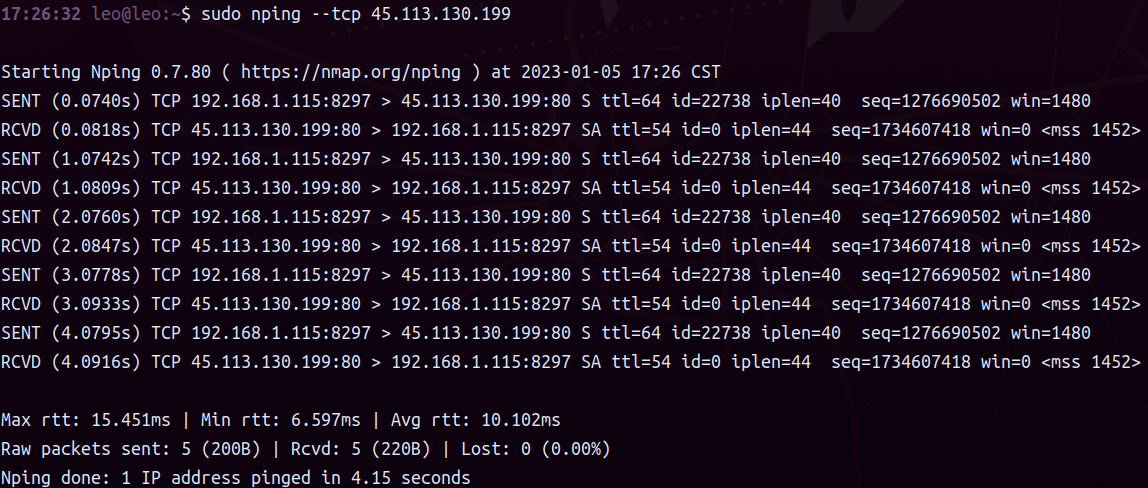
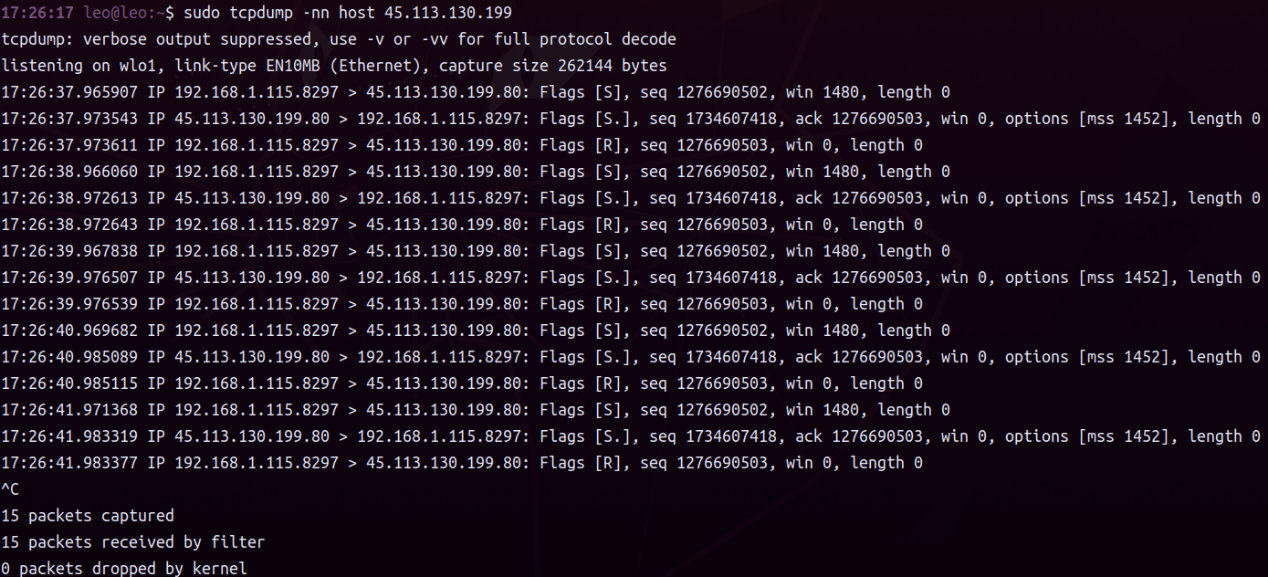
**Chapter 8**

**Conclusion**

With the popularity of the Internet, the demand for Geolocalizing network users and machines is also getting higher and higher. Evaluating the performance of existing geolocation methods helps choose appropriate geolocation methods on different platforms. In this work, we use VPNs as reference points to probe target location since these VPNs is distributed all over the world, it means that there are different angles to detect the target location more accurately. And we use Twitch ingest servers as the targets because they have known locations to help evaluation on geolocation methods. In the evaluation method, we use rank difference to evaluate how many VPN location candidates can hit the ground truth location, and propose error distance to evaluate the error of the VPN location we map the target to in the experimental results. The results show that CBG performs the best in most cases, followed by Shortest Distance and Shortest Ping, and GeoPing may cause poor performance owing to noise or some additional overhead. Finally, we propose a hybrid method that combines two geolocation methods, namely CBG and Shortest Distance. And the result shows that CBG-Shortest Distance method gives a lower error distance, as we see in the evaluation result, CBG-Shortest Distance method has the lowest error on the most important evaluation parameter (total error distance). And this work provides a hint for the development of future geolocation methods. After conducting this research, we give some advice to future researchers who are interested in the measurement-based geolocation methods:

1. About generating packet transmission between two machines, the most intuitive idea is to use ping. But some systems block ICMP packets, or the ICMP packets will be given lower priority. So, the first method to solve this problem is to use the “normal way” to obtain packet transmission. For example, if we want to get the RTT between the twitch ingest server and us, we can stream on twitch to get the packet transmission between twitch ingest server and us. As long as we get the packet transmission, calculating RTT becomes much easier. However, some machine doesn’t have the “normal way” to get the packet transmission. So we have to come up with other ways. I found an alternative, Nping [17] is an open source tool of Nmap [18], and it can be used to generate network packets with multiple range of protocols, so users can control the protocol headers. Not only ICMP packets in the IP layer, they can generate packet transmission of protocols including TCP/UDP/ARP. Therefore, when one of the protocols is blocked, we can use nping to find other available protocols to generate packet transmission.

For example, “ingest-proxy-edge-325976.tpe03.justin.tv” is one of twitch outgest server, and its ip is “45.113.130.199”.  
if we use ping 45.113.130.199, then it will not respond to us. However,   
if we use nping --tcp 45.113.130.199, then we can get the tcp packet transmission between 45.113.130.199 and us.



1. You can try some different hybrid methods and evaluate the performance of these methods. Since the hybrid method may be able to eliminate some measurement errors that lead to serious errors in geolocalization.
2. I suggest the future researchers can evaluate the geolocation methods periodically to observe whether the variation in the performance of these geolocation methods as the Internet changes.
3. If you want to use NordVPN as the reference points.
   1. The three necessary RTTs must be measured at similar times.

In my experiment, if the interval in the packet collection of the same pair of VPN and ingest server is too long (for several months). The RTT between them may have some variation and it may lead to lower accuracy in geolocalization. For example, RTT between Singapore VPN and Poland ingest server was 169.426 ms in the first measurement. But after 6 months, it changed to 293.58 ms. I infer that this is because the change on the Internet.

In addition, in my experiment, VPN servers in some areas may become unavailable for a certain period of time, or suddenly become available after a certain time. Therefore, we should not prolong the period of using VPN for measurement to avoid VPN servers’ downtime and losing some VPN’s perspectives.

Moreover, If you have enough hardware resources, I recommend measuring the three necessary RTT simultaneously to avoid VPN servers going out of service and RTT variation between endpoints or routers due to Internet changes.

* 1. There are some limits when using VPN as reference point,   
     Limit 1: Reference Point is not abundant.

There are not a lot of VPN we can use, we only have 82 VPN in different locations to be reference point. However, prior papers often use PlanetLab Nodes as reference point, which have up to 400 nodes in different locations. If the reference point is less, then the accuracy of geolocalization will decrease.

Limit 2: VPN distribution is mainly in certain regions.

The previous Figure 7 shows VPN distribution, the VPN servers are mainly distributed in Europe and North America. But less or none in Canada, South America, Africa, West Asia, Middle East, China, and Russia. If the target is in an area with little VPN coverage. Then, it hard to geolocalize this target with high accuracy.

To deal with the two issues. I suggest that you can try other VPN servers provided by other VPN companies to increase the coverage of reference point and increase the accuracy in geolocalization. If there are two servers provide by NordVPN and other VPN company respectively are close. You can test if the RTT fingerprints are similar when the two servers act as VPs and LMs to determine whether the VPN servers provided by other companies are suitable added to the VPN set to expand the reference point set.   
For example, there are two VPN servers A and B, A is provided by NordVPN, and B is provided by other VPN company. If A and B are close, pair A and B. Thus there will be several pairs. Next, we discuss a single example, in Figure 34, treat A and B as VPs and there are N LMs (N is the size of reference point set). And then compute the RTT between each VP and each LM to get).

In addition, In Figure 35, treat A and B as LMs and there are N VPs. And then compute the RTT between each VP and each LM to get  
).

If both RTTVP difference and RTTLM difference are small in all pairs, you can try to add all VPNs provided by other VPN company to the VPN set to expand the reference point set, because the VPN from different companies at the same location has similar characteristics. Hence we infer that adding the VPN server in the new location to the set will benefit the geolocation system.

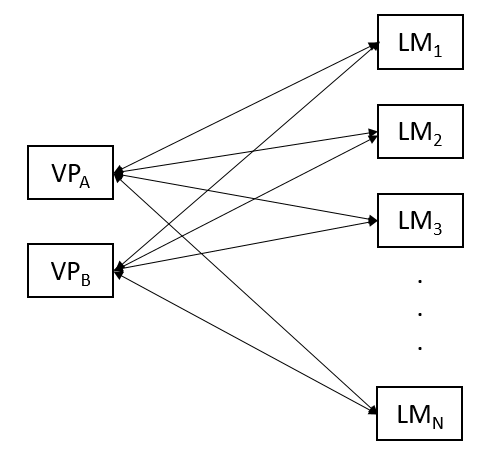


Figure 34. Fingerprint of VPA and VPB

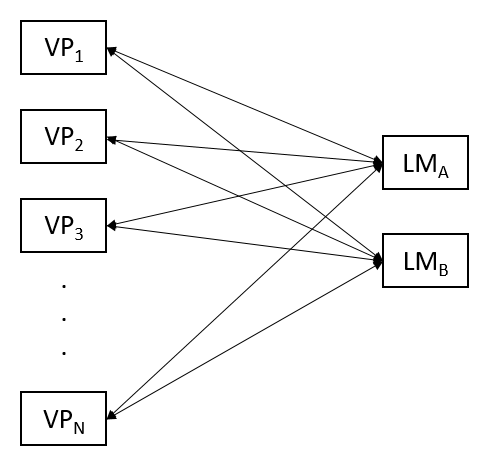


Figure 35. Fingerprint of LMA and LMB

Limit 3: Only one class of target.

This study only use twitch ingest servers as the target. And we evaluate the performance of these measurement-based geolocation methods including Shortest Ping, GeoPing, CBG, Shortest Distance, and Hybrid Method. However, if one method performs well when the target is twitch ingest servers, it doesn’t mean it will perform well in other target. Therefore, I suggest that we can find other known location target to expand the target set. And if there is a method perform well in most target set, then we can mainly use this method to geolocalize other unknown location target.

**Reference**

1. M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos. A Look at Router Geolocation in Public and Commercial Databases. In ACM IMC, 2017.
2. Y. Shavitt and N. Zilberman. A study of geolocation databases. Journal on Selected Areas of Communications, abs/1005.5674, 2011
3. Dan, Ovidiu, Vaibhav Parikh, and Brian D. Davison. "IP geolocation through reverse DNS." ACM Transactions on Internet Technology (TOIT) 22.1 (2021): 1-29.
4. Zi Hu and John Heidemann. Towards Geolocation of Millions of IP Addresses. Proceedings of the ACM Internet Measurement Conference (Boston, MA, USA, 2012), 123–130.
5. Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In Proceedings of the ACM SIGCOMM Conference, pages 173–185, San Diego, California, USA, August 2001. ACM.
6. B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation  
   of Internet hosts. IEEE/ACM Transactions on Networking, 14(6):1219--1232, 2006.
7. PlanetLab <https://en.wikipedia.org/wiki/PlanetLab>
8. NordVPN <https://nordvpn.com/zh-tw/>
9. Hsi Chen. Geolocating IP Adresses <https://hackmd.io/@gRXsWssJQIy-dR9p369JLQ/B1MhZ2cUu>
10. G. Ballintijn, M. van Steen, and A.S. Tanenbaum. Characterizing Internet   
     Performance to Support Wide-area Application Development. Operating Systems   
     Review, 34(4):41-47, October 2000.
11. Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall,  
     Thomas Anderson, and Yatin Chawathe. Towards IP geolocation using delay and   
     topology measurements. In Proceedings of the ACM Internet Measurement   
     Conference, pages 71–84, Rio de Janeiro, Brazil, October 2006. ACM.
12. F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush, “A Measurement Study on   
     the Impact of Routing Events on End-to-end Internet Path Performance,” in   
     Proceedings of ACM SIGCOMM, 2006.
13. Twitch. Twitch Status [https://Twitchstatus.com/](https://twitchstatus.com/)
14. Twitch. Streaming Platform [https://www.Twitch.tv/](https://www.twitch.tv/)
15. OBS. Streaming Tool <https://obsproject.com/>
16. Xdotool. <https://manpages.ubuntu.com/manpages/trusty/man1/xdotool.1.html>
17. Nping <https://nmap.org/book/nping-man.html>
18. Nmap <https://nmap.org/>
19. NordVPN. NordVPN API [https://sleeplessbeastie.eu/2019/02/18/how-to-use-public-NordVPN-api/](https://sleeplessbeastie.eu/2019/02/18/how-to-use-public-nordvpn-api/)
20. NordVPN. Ip-lookup [https://NordVPN.com/zh-tw/ip-lookup/](https://nordvpn.com/zh-tw/ip-lookup/)
21. Zi Hu, John Heidemann, and Yuri Pradkin. LANDER geolocation datasets.   
     <http://www.isi.edu/ant/traces/geolocation,> August 2012.