STOCHASTIC METHODS FOR GEOLOCATION OF INTERNET HOSTS

by

Inja Youn
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____________________________ Dr. Dana Richards, Dissertation Co-Director
_____________________________ Dr. Daniel Carr, Dissertation Co-Director
_____________________________ Dr. Kris Gaj, Committee Member
_____________________________ Dr. Brian Mark, Committee Member
_____________________________ Dr. Hassan Gomaa, Department Chair
_____________________________ Dr. Daniel Menascé, Senior Associate Dean

_____________________________ Dr. Lloyd J. Griffiths, Dean, The Volgenau
School of Information Technology and
Engineering

Date: _________________________ Fall Semester 2008
George Mason University
Fairfax, VA
STOCHASTIC METHODS FOR GEOLOCATION OF INTERNET HOSTS

A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at George Mason University

By

Inja Youn
Master of Science
George Mason University, 2003

Director: Dana Richards, Professor,
Department of Computer Science
Director: Daniel Carr, Professor
Department of Statistics

Spring Semester 2009
George Mason University
Fairfax, VA
TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION ............................................................................................ 1

CHAPTER 2. PROBLEM STATEMENT ........................................................................... 3

CHAPTER 3. RELATED WORK .................................................................................... 7
  3.1. TERMINOLOGY .................................................................................................... 7
  3.1.1. Active Landmarks ........................................................................................ 7
  3.1.2. Passive Landmarks ....................................................................................... 8
  3.2. GEO Ping ........................................................................................................... 8
  3.3. CONSTRAINT BASED GEOLOCATION ............................................................... 9
  3.4. TOPOLOGY-BASED GEOLOCATION ............................................................... 13
  3.5. CLIENT MEASUREMENTS AND MIDDLEBOX DETECTION ......................... 16

CHAPTER 4. RESEARCH METHODS ........................................................................... 18
  4.1. WEBSITE NETWORK ......................................................................................... 18
  4.2. SERVER-SIDE MEASUREMENTS ................................................................. 19
  4.3. CLIENT-SIDE MEASUREMENTS .................................................................... 21
  4.4. CLIENT DATA .................................................................................................. 23
    4.4.1. JavaScript and Java Applets ................................................................. 24
    4.4.2. Flash Objects .......................................................................................... 26
    4.4.3. TCP SYN Fingerprinting ...................................................................... 27
    4.4.4. Browser Timezone/Language .............................................................. 27
    4.4.5. Web Cookies and Flash Cookies ......................................................... 27
    4.4.6. Probabilistic Self-Calibrating Algorithm and Data Fusion ..................... 28

CHAPTER 5. PROBABILISTIC GEOLOCATION MODEL ........................................... 29
  5.1. ALGORITHM FOR MAXIMIZATION OF $F(\lambda, \phi)$ ..................................... 34

CHAPTER 6. IMPROVEMENTS OVER GEOPING .................................................... 37

CHAPTER 7. LOCALIZATION IN THE INTERNET ...................................................... 40
  7.1. GENERALIZED CRAMER-RAO Bound ............................................................ 42
LIST OF TABLES

Table 1. Examples of Kernels for Density Estimation .................................................... 32
LIST OF FIGURES

Figure 1. The “baseline”, as defined in Gueye et al. [3] .................................................... 10
Figure 2. “Baseline” and “bestline” in Gueye et al. [3] .................................................... 11
Figure 3. CBG Geolocation .......................................................................................... 13
Figure 4. Server side measurements architecture for web clients geolocation ............ 20
Figure 5. Client side measurements architecture for web clients geolocation .......... 22
Figure 6. Getting public IP address of clients behind transparent proxies ............. 24
Figure 7. Getting local IP address of clients behind anonymizing proxies .......... 25
Figure 7. Getting public IP address of clients behind anonymizing proxies ......... 26
Figure 8. Upper bound underestimation with CBG method \( g_t > \hat{g}_t \) ............ 30
ABSTRACT

"{CLICK HERE TO ADD TITLE IN ALL CAPS}"

"{Click here to add authors name and Degree being earned such as MA/MS or PhD}"

George Mason University, "{Click here to add year}"

Thesis/Dissertation Director: "{Click here to add directors name}"

"{Click here to add double spaced the text}"
"{Click here to add double spaced the text}"
Chapter 1. Introduction

As Internet is a global network, commercial companies have recently realized the importance of mapping an IP address to its geographical location. Two of the most important applications are geographically targeted content (distribution content and advertising) and prevention/reduction of internet frauds (such as credit card fraud, identity theft, spam and phishing). There are two main approaches to this problem. The first approach is used by commercial IP geolocation products and consists of building and maintaining a database which assigns to each block of IP addresses to its estimated location. This method has obvious drawbacks, such as obsolete and incomplete/inconsistent data, as well as exponential growth of the database with the advent of the IPv6. In addition, two IP addresses from the same block often have completely different geographical locations. The second approach is to use active measurements to estimate the location, using a distributed network of servers and a small number of nodes (aka landmarks) with known location. This second approach can be made self-calibrating and is able to detect when a target IP address has changed its geographical coordinates significantly as well as find its new location. However, it also the disadvantage of large errors due to the non-linear correspondence between geographical distance and internet distance, and its usefulness is limited to geolocation of
servers which can be pinged and/or tracerouted. Both methods described above have problems with identification and geolocation of clients behind firewalls and middleboxes, such as NATs and proxy servers.

This proposal improves over the second approach in several ways. First, it uses a novel probabilistic approach for the IP location problem, in contrast to the previous deterministic approaches. Second, it attempts to geolocate the clients connecting to a webserver, by using client information and client-based measurements performed by JavaScript and browser plugins, such as Flash Objects and Java Applets. These applications are able to initiate outgoing connections from the client, thus bypassing firewalls and eventually middleboxes and reveal information such as client originated measurements and local IP addresses. All the information will be assigned a weight (confidence factor) and will be fused with a probabilistic algorithm. Third, it investigates different variations and improvements of previous methods, such as GeoPing. In particular, it will study the effect of using different distance and proximity measures for the geolocation process.
Chapter 2. Problem Statement

The geolocation of IP addresses is an important problem. Many sites nowadays rely on more or less accurate databases to determine the location of a customer. Customer geolocation has important application, such as determining regional distribution of the clients, local news delivery, targeted advertising, restriction of content delivery based on regional policies, etc. In addition, IP geolocation can be applied in intrusion detection, to find the approximate geographical position of an intruder. However, such databases are often proprietary and manually updated. Therefore, their consistency and accuracy are questionable at best. In addition, with the advent and adoption of IPv6, such databases become harder and harder to update and maintain. As an alternative approach, RFC 1876 proposes to incorporate geographical information into DNS records (DNS LOC). However, the implementation of this approach is not widespread nowadays, since it requires changes in the DNS records.

There have been a number of efforts to automate the geolocation of IP addresses. Padmanabhan and Subramanian [8] have investigated three techniques – inferring the geographic location of an Internet host based on the DNS name of the host or another nearby node (GeoTrack); clustering the IP address space into likely collocated clusters (GeoCluster), and pinging the host, with geolocation of the IP addresses by correlation of
the ping delays (GeoPing). Gueye et al. [5] has improved upon GeoPing using an idea borrowed from sensor networks. His Constrained Based Geolocation (CBG) algorithm uses a triangulation-like algorithm to determine the probable location of the targets. Katz-Basset et al. [6] further improved upon the idea by finding the hosts along the Internet route using *traceroute* utility and geolocating the hosts simultaneously using CBG. The approach of [6] is called Topology Based Geolocation (TBG). None of these approaches use measurements originated at the client side to improve the location process, and all use deterministic algorithms, which often have unacceptable errors of more than 1000 km in geolocation process.

Casado and Freedman [1] enumerated several ways to perform measurements originating at targets. Some of them are used in this proposal, such as JavaScript, Java Applets, SYN fingerprinting, browser time zone/language and cookies. However, rather than using the measurements for geolocation, the authors use the measurements together with a database-type geolocation system (Quova) to generate inferences and statistics of clients behind middleboxes (proxies and NATs).

Dabek et al. [10] and Shavitt and Tankel [12] attempted to model the Internet network distance by embedding the Internet in a higher dimensional space. These adaptive distributed algorithms have the property of providing decent estimates of latency among hosts in the long run. While network distance and geographical distance are two different things, the two algorithms can be used to automatically find the nearest landmarks by probing the most probable landmarks from the target machines in real-time.

The novelty of our proposed geolocation system consists of:
1. For the Server-Based geolocation, this proposal introduces a new probabilistic algorithm for client geolocation called Stochastic Based Geolocation (SBG). The algorithm follows a Bayesian approach. The prior distribution of delay-distance pairs is estimated from the measurements between landmarks with known locations, using bivariate kernel density estimation. Using the delay to each active target as input, the SBG algorithm calculates the a posteriori distribution of the distance to each target. In the next phase, the SBG algorithm calculates an approximation of the Maximum a posteriori estimator (MAP), using a force-directed algorithm, or an exhaustive search over a discretization of the domain.

2. Introduces a new concept, named Client-Based Geolocation. Client-Based Geolocation collects information from the client and performs measurement to the active and passive landmarks. The advantages of this method are:
   
   a. Originates client-side measurements from clients to targets. This is done by embedding JavaScript/Flash and Java applets in the webpage requested by the client. Thus, all targets can act as active landmarks when they request a webpage. In JavaScript, the measurement is achieved with the help of a JavaScript XMLHttpRequest. The “same-site” restriction is bypassed by embedding inline frames (iframes) in the page with content from different servers.

   b. Collects additional information from clients allows detection of clients behind proxies and aids in geolocation. The content from clients includes: HTTP headers, SYN fingerprinting, client time zone/language, identifying
the client by web and flash cookies. This information can be used for identifying the clients and improving geolocation information when the same client connects again (cookies). It is also useful in figuring out proxies (SYN fingerprinting),

c. Detects and attempts to geolocate clients hiding behind proxies, NATs and VPN connections. When a target is detected as using a middlebox, and finding the real location of the target fails, the geolocation algorithm will assign a low confidence to the estimated location of the target.

d. Combines the information gathered from client and server measurements, as well as the additional information, which aids geolocation.

The disadvantage of Client-Based Geolocation is that the targets have to load a specially crafted page in their browsers, in order to perform the measurements.

3. Investigates variations of the GeoPing algorithm, by replacing the Euclidean distance in delay space with other distances and proximity measures, such as the p-Norm and Mahalanobis distance.
Chapter 3. Related Work

While client-based measurement techniques have not been explored until now for geolocation purposes, there are several important papers concerned with geolocation using server-based measurements. After a brief explanation of the terminology, this chapter presents the most important techniques: GeoPing, Constrained Based Geolocation and Topology Based Geolocation.

3.1. Terminology

3.1.1. Active Landmarks

In order to perform location estimation of a host, some ground truth information is needed to start. We use a distributed network of reference hosts called active landmarks, with well-known geographical locations. The active landmarks can measure the round trip time (RTT) delay both between each other and to other targets. On their web servers, the active landmarks contain JavaScript, Java Applet and Flash code as web beacons to be included in hidden inline frames on the servers of the website network. In addition, the active landmarks are able to probe the passive landmarks and clients using traceroute and ping utilities.
3.1.2. Passive Landmarks

Unlike the active landmarks, the passive ones do not carry on measurements by themselves. Rather they are measured by other landmarks and clients. Some of the passive landmarks are found using the `traceroute` application, and are carefully chosen to improve the geolocation capability.

3.2. Geo Ping

GeoPing was one of the earliest attempts to use the Internet measurement to geolocate a target. The algorithm is very simple and quite efficient. A small number of hosts (active landmarks) perform network delay measurements both between themselves and to so-called passive landmarks. When a new target host is encountered, the active landmarks calculate the distance between the active landmarks and the target. These delays are compared to the existing measurements. Finally, the target is mapped to the landmark (active or passive) having the most similar delays using a minimum residual sum of squares technique (Padmanabhan and Subramanian, [8]).

In other words, let \(L_1, L_2, \ldots, L_N, L_{M}\) be the landmarks, where the first \(N\) are active landmarks and the last \(M - N\) are passive landmarks. The location of the landmark \(L_j\) denoted by \((\lambda_j, \varphi_j)\) is known for \(1 \leq j \leq M\). We can measure the delay from landmark \(L_i\) to landmark \(L_j\), which we call \(d_{ij}\). Here, \(L_i\) must be an active landmark, however, \(L_j\) can be either active or passive. Thus, we have \(1 \leq i \leq N\) and \(1 \leq j \leq M\). Also, we can measure the delay \(d_{it}\) from each active landmarks to the target \(\tau\). Let \(L_k\) be the landmark which is the closest to \(\tau\) in the delay space, which is:
\[
k = \arg \min_{l \neq j \neq M} \left\{ \sum_{i=1}^{N} (d_{ij} - d_{ir})^2 \right\}
\]  

(1)

We then assign the location of the landmark \( L_k \), which is \((\lambda_k, \varphi_k)\), to the target.

This algorithm has the drawback that the distance in the delay space is sensitive to errors in the delay measurement, which are inherent, especially when the distance between the landmark and target is large. Techniques to reduce this sensitivity will be explored in Chapter 6.

### 3.3. Constraint Based Geolocation

Gueye et al. [5] used an approach based on sensor network called constraint-based geolocation (CBG). They noticed that the packet propagation speed on the Internet is at most the speed of light through optical fiber cable, which in turn is about 2/3 of the speed of light. This restriction induces circle-like bounds on the location of the target. If we denote the round trip delay between two hosts by \( d \), an upper bound for the geographical distance between the two hosts is given by:

\[
\hat{g} = \tilde{c} \cdot d
\]

(2)

Where \( \tilde{c} = \frac{2}{3} c \) and \( c \) is the speed of light \((3 \times 10^8 \text{ m/s})\). When the round trip is measured in ms and the geographical distance is measured in km, \( \tilde{c} \) is approximately 100 km/ms, thus the equation can be written as:

\[
d = \frac{1}{100} \hat{g}
\]

(3)

This line is called “baseline” by the authors, and is illustrated in Figure 1. All the distance-delay measurements are situated above the baseline.
However, the distance upper bound provided by the baseline is too loose to be of any use. To tighten this upper bound, Gueye et al. [5] fit a so-called “bestline” $d = m_i \cdot g + b_i$, to each landmark. This “bestline” is the tightest bound below the distance-delay pairs.
The constraints are: 1) all the distance-delay pairs should lay above the “bestline”, 2) the slope of the “bestline” should be at least as large as the one of the “baseline”, and 3) the intercept of the “bestline” should be positive. Thus, the problem becomes a linear programming (LP) problem:
\[
\min_{m_j,b_j} \sum_{j=1}^{N} (d_{ij} - m_j \cdot g_{ij} - b_j) \quad \text{subject to:}
\]

\[
d_{ij} - m_j \cdot g_{ij} - b_i \geq 0 \quad \forall j \neq i
\]

\[
m_i \geq m \quad (=1/100)
\]

\[
b_i \geq 0
\]

For a given target, the algorithm calculates the upper bounds based on the delays \(d_{ir}\) from each active landmark to the target. Thus the “bestline” upper bound is:

\[
\hat{g}_{ir} = \frac{d_{ir} - b_i}{m_i}
\]

The algorithm draws a circle with radius \(\hat{g}_{ij}\) around each landmark \(L_i\). By intersecting these circles, the authors obtain a region where the target should be located. The target is located in the center of this region.
The area of the region is taken as a measure of confidence of geolocation. The authors claim their method is an improvement over the GeoPing method. However, this method also has some problems, outlined in Chapter 5.

3.4. Topology-based Geolocation

Katz-Bassett et al. [6] suggested potential improvements of geolocation by taking into account the network topology.

Using `traceroute` utility, the authors measured the delay between routers on the path from the active landmarks to the targets. After measurements are completed, the
algorithm of [6] tries to geolocate both the target and the intermediate routers simultaneously.

A summary of the techniques used by Topology Based Geolocation is:

1. Using traceroute utility to map the topology. The hop latency is estimated as the difference between the round-trip-time to adjacent routers. This technique is not accurate when the routing is not symmetric. To determine whether the routing is symmetric the authors designed three techniques: 1) observing the reverse TTL values, 2) measuring the paths in both directions between pairs of landmarks, and 3) probing both ends of the links from all active landmarks.

2. Clustering network interfaces. Two techniques are used to cluster network interfaces. The first one is named “Mercator” after the Mercator Project [4], which first employed it, and sends UDP packets to high-numbered ports on a set of interfaces. Routers will answer with an ICMP “port unreachable” message containing a source address. If two interfaces reply with the same IP address they are aliases for each other. The second technique is called “Ally” [10], and is used by Rocketfuel Project. It acts in a similar way to “Mercator” technique; however, instead of looking to the IP source address of the ICMP message, it sends probes pairs of interfaces and looks at the IP-ID instead. In most routers, IP-ID is a counter which is incremented with each generated packet. If the counters are close to each other, the two addresses are aliases.
3. Validating the location hints from DNS names. Most routers have location hints in their DNS names. For example \textit{L100.VFTTP-38.WASHDC.verizon-gni.net} is most likely located in Washington DC. However, some of these names are incorrect, due to reconfiguration and repairs. Thus, the location hints need to be validated using the bounds on RTT measurements, clustering and hop latencies.

4. Solving the constrained optimization problem. To formulate the problem, we denote by \( L_1, L_2 \ldots L_N \) the set of targets, and by \( x_1, x_2 \ldots x_M \) the set of targets. The set of targets is formed by the host to be geolocated (\( \tau \)), and by the intermediate routers on the path from eachs \( L_i \) to \( \tau \). The Topology Based Geolocation considers two kind of constraints:

a. Hard delay constraints, similar to the constraints of Gueye et al. \cite{5}, but with the same bestline for all targets, corresponding to the propagation speed of internet \( \tilde{c} = \gamma \cdot c \), which is less than the speed of light in optical fiber, however, it is considered by the authors to hold for 99.96% of the links. If we denote by \( d(\cdot, \cdot) \) and \( g(\cdot, \cdot) \) the delay respectively the geographical distance between a landmark/target and another landmark/target, the constraints can be formulated as:

\[
g(L_i, x_j) \leq \tilde{c} \cdot d(L_i, x_j) \quad \forall i = 1, 2 \ldots N, \quad \forall j = 1, 2 \ldots M
\]

Let \( C_d \) be the set of hard delay constraints.

b. Soft link latencies constraints. Let \( d(x_i, x_j) \) be the estimated delay between the adjacent nodes \( x_i \) and \( x_j \). From these delays, the authors
obtain an estimation of the distance between the two nodes, which they
denote by \( h(x_i, x_j) \). The authors are not clear about how to estimate
\( h(x_i, x_j) \). The soft delay constraints can be formulated as:

\[
d(x_i, x_j) = h(x_i, x_j) + e_{ij} \quad \forall i, j = 1, 2 \ldots M
\]

where \( e_{ij} \) is some error which account for the noise of the measurements,
and which should be minimized during geolocation.

Let \( C_i \) denote this set of soft link constraints. The optimization problem can now
be formulated as:

\[
\min \sum_{i, j \in C_i} |e_{ij}|
\]

subject to: \( C_h, C_l \)

This is not a linear programming problem; however, it can be reformulated as a
semidefinite program and solved using fast solvers, such as SeDuMi [7] or
Vivaldi [11] to solve for the optimal location of the targets on the map.

The authors claim their algorithm is an improvement over Constraint Based
Geolocation (CBG); however, they are not specific enough to enable the replication of
their results.

3.5. Client Measurements and Middlebox Detection

All previous methods use measurements originating in active landmarks. Casado
and Freedman [1] performed measurements originating from the client. We use most of
their methods here, together with some newer methods recently discovered and used for
de-anonymyzing clients of the Tor networks (such as Flash objects). However, the intent of Casado and Freedman [1] was to detect whether their client was hiding behind a middlebox. We use the methods of this paper to improve the accuracy of the geographical location of web clients by originating measurements from the target machines, as well as to attempt geolocation of the clients behind middleboxes by tricking their computers into revealing their true IP addresses.
Chapter 4. Research Methods

Our research enhances previous approaches with several novelties. In addition to the server-side measurements performed in the previous work, our approach used client-side measurements for geolocation through JavaScript, Java applets and Flash objects. Clever analysis of the measurements performed by these browser extensions enables us to geographically locate clients behind middleboxes, which was not possible in the previous work. The additional information provided by the HTTP headers and Javascript, Java and Flash plugins (operating system, browser timezone/language, local IP address, public IP address behind proxy) are used to further restrict the geographical area of the target, providing an improvement in accuracy. In addition, user tracking through cookies and Flash local objects can enable the system to accumulate information about users behind middleboxes over time. The learning algorithm should be able to discard outdated data and improve the accuracy of the user geolocation over time.

Our proposed system architecture is described in detail below:

4.1. Website network

The first thing that we need in order to start our experiment is to get a representative sample of measurements using two methods. The first method consists of
including web beacons in IFrames on different websites, and getting the measurements while the users view the pages. The second method is based on getting voluntary websites to temporarily redirect a fraction of their web traffic to our sites for performing measurements; after performing measurements the traffic is redirected back to the original web sites. Casado and Freedman [1] use the second method in two contexts. First, they build a community of content providers who agree to temporarily redirect their pages to perform measurements using one of the methods above. Second, they redirect a fraction of the CoralCDN web content distribution network through their measurement servers.

This thesis will use PlanetLab as a platform measurement network. The PlanetLab platform has 68 nodes at different sites in the US and Canada, which will act as active landmarks. As passive landmarks and targets, we can use different sets of hosts with known location, such as university websites, which can be readily located geographically, and have well-known web addresses.

### 4.2. Server-Side measurements

Server-side measurements originate in active landmarks, and have as destination the target, a set of hosts with known location (passive landmarks) or intermediate routers. If the target is a server, which responds to ping/traceroute, this measurement technique can be employed statically, with a predefined set of targets. However, for geolocating web clients, we have to modify our architecture as in Figure 4:
When a client requests a web page from the web server, the web server collects its IP address and broadcasts it to the active landmarks (aka PlanetLab nodes). The active landmarks perform the measurements using `ping` and `traceroute` utilities, and report the results of the measurements either back to the server or to an analytics server.

The technique works if the web client responds to `ping/traceroute`, which most of the workstations do not. In this case, this server-side technique uses the location of the last router which responds to `ping` for the location of the client. If the client is behind a

![Figure 4. Server side measurements architecture for web clients geolocation](image)
proxy, thus geographically distant from the last router, this geolocation technique produces large errors.

**4.3. Client-Side measurements**

One of the novelties of this proposal is the integration of the client and server measurements. While the previous geolocation approaches and experiments have collected and used measurements originating only in the active landmarks, our approach to improve the methodology uses both client-side and server-side measurements. Unlike the server-side measurements, which are prone to errors when users are located behind proxies or NATs, client-side measurements can estimate more accurately the network distance between client nodes and landmarks.

The platform architecture for client-based measurements is presented in Figure 5.
JavaScript can perform a measurement by requesting to load an image from each site. This method does not restrict JavaScript to the same origin policy. However, the script might be slow and the results unreliable. Another way is to use the XMLHttpRequest API from JavaScript. Being asynchronous, this Ajax method is more reliable and does not block the browser. Thus XMLHttpRequest has less impact on the client who loads the page. However, this method, as well as Java Applets and Flash objects, is subject to the same origin policy. This policy can be bypassed by loading the

Figure 5. Client side measurements architecture for web clients geolocation.
objects from different sites in inline frames. Since we have only a limited number of measurements available (too many flash objects and/or applets would slow down the target system and produce unreliable measurements), we have to select only the active and passive landmarks that our system estimates to be closest to the target in terms of network distance. This will maximize the measurement accuracy.

By performing measurements from client to landmarks and from landmarks to client’s public address, we can statistically determine whether the client is behind a proxy. In this case, the client’s proxy address should not be used for geolocation.

4.4. Client Data

As stated in the section 3.4, client-side measurements are crucial for an accurate estimate of geographical location, especially for the clients behind middleboxes.

Another advantage of client-side measurements is that the web beacons contained in client page can reveal additional information about the target. Java applets can reveal the local IP address of the target. Both Flash and Java plugins can attempt to bypass the proxies and reveal the true public IP address of the target. HTTP headers contain, among other information, the operating system, timezone and language of the client. If the client is connected through a transparent (non-anonymizing) proxy, the web server will get the real IP address of the client from the X-Forwarded-For HTTP header, as in Figure 6.
4.4.1. JavaScript and Java Applets

Unlike JavaScript or Flash objects, Java Applets have access to the local IP address by using the `getLocalAddress()` method of the `Socket()` class, as detailed in Figure 7.
In addition, unless the Java Control Panel is configured to use the browser’s proxy settings, the applet would bypass the proxy and reveal the computer’s public address behind the proxy. This method is illustrated in Figure 8.
There are two methods to detect the public IP address of a client behind an anonymizing proxy. The first method applies only to Java applets and

4.4.2. Flash Objects

Flash objects can be used in the same way as JavaScript and Java applets. While Flash does not reveal the local address, a Flash object can also open a socket (XMLSocket object), which bypasses the browser’s proxy settings.

Figure 8. Getting public IP address of clients behind anonymizing proxies
4.4.3. TCP SYN Fingerprinting

TCP SYN Fingerprinting is useful for estimating the client’s operating system and finding whether the client is behind a proxy or not. The client is detected as being behind a proxy if the SYN fingerprint is a known proxy type (e.g. CISCO or NetApp).

4.4.4. Browser Timezone/Language

The client’s timezone and language impose useful additional constraints on the user location. While the client’s timezone is most likely to be accurate and imposes restrictions on the longitude, the client language is not always relevant (especially for North America case).

4.4.5. Web Cookies and Flash Cookies

Web cookies and Adobe Flash locally stored as objects (Flash cookies) can be used to identify a user. When the client connects again to the server, the client will be identified and the estimation of the geographical location can be improved based on previous collected information. While web cookies can be easily deleted with a browser menu option, Flash cookies can only be deleted manually. Most users will not delete them. Thus, Flash cookies can be more useful than web cookies.
4.4.6. Probabilistic Self-Calibrating Algorithm and Data Fusion

The algorithm should be trained on a representative sample of hosts with known geographical location and will have to satisfy the following properties:

1) The algorithm should be able to calibrate the client measurements versus server measurements;

2) The algorithm should be able to detect the host behind middleboxes

3) The algorithm should be able to choose a relevant number of landmarks to be probed by target machines;

4) The algorithm should be able to fuse the information supplied by measurements server and client as well as additional knowledge provided by the target machines, with proper weights;

5) The algorithm should be robust with respect to geolocation errors, and should correct previous errors when updated data becomes available (in terms of quantity and/or quality).
Chapter 5. Probabilistic Geolocation Model

The Gueye et al. [5] model has several obvious shortcomings. The first is that their “bestline” upper bound for distance is calculated from a limited set of empirical data, and it is not guaranteed to hold for all targets. This deficiency is best illustrated when there is a direct high speed (optical cable) connection between a landmark (say $L_i$) and target, but no high speed/direct connection between $L_i$ and the rest of the landmarks. Obviously, the bestline would be situated much higher than the baseline, while the distance-delay plot of the target would fall between baseline and bestline. Thus, if the upper bound distance between any landmark $L_i$ and target will be underestimated, the CBG algorithm may result in either an incorrect geographical location of the target, or completely fail. The situation is illustrated in Figure 9.
If we update the bestline to be below this spurious point, the new bestline will be too close to the baseline, thus the upper bound would become too large and practically useless for geolocation purposes.

The main reason behind these shortcomings is that Gueye et al. [5] use only the limited information provided by the baseline and bestline in their location estimation of the target host. The distribution of the landmark-delay pairs is not used at all in CBG estimation.

Figure 9. Upper bound underestimation with CBG method \((g_{i\tau} > \hat{g}_{i\tau})\).
My algorithm proposes a probabilistic method for an Internet host geolocation, which takes advantage of the full distribution of the landmarks and delays.

**Step 1.** The first step of the algorithm is identical to that of Gueye at al. and builds a scatterplot for each active landmark $L_i$ by measuring the distance-delay pairs $(g_{ij}, d_{ij})$ for each active or passive landmark $L_j, j = 1, 2 \ldots N$.

**Step 2.** The second step of the algorithm estimates the bivariate distributions of the random variables $(G_i, D_i)$, for $i = 1, 2 \ldots n$. Here $G_i$ represents the geographical distance between landmark $L_i$ and the target $\tau$ and $D_i$ is the measured delay between $L_i$ and $\tau$.

The data sample is the one collected at step 1:

$$\{(g_{i1},d_{i1}), (g_{i2},d_{i2}) \ldots (g_{in},d_{in})\}$$

(9)

The sample data is transformed into a valid probability density function using kernel density estimation:
\[ f_{G_i,D_j}(g_i,d) = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{h_g h_d} K\left( \frac{g - g_{ij}}{h_g}, \frac{d - d_{ij}}{h_d} \right) \] (10)

In the last equation, \( h_g \) and \( h_d \) are the bandwidth parameters of the geographical distance, respectively the delay and should be chosen to minimize the geolocation error on a training set. The kernel \( K(x,y) \) is a valid probability density function and is symmetric with respect to both \( x \) and \( y \).

\[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K(x,y) dx dy = 1, \text{ and } K(-x,y) = K(x,-y) = K(x,y) \] (11)

Kernel density estimators for bivariate distributions are usually built upon two kinds of kernels – product kernels and radial kernels. The product kernels are constructed by multiplying two one-dimensional kernels:

\[ K(x,y) = K_1(x) \cdot K_2(y) \] (12)

Radial basis kernels are rotation invariant and depend on the arguments \((x,y)\) only through the magnitude of \((x,y)\), which is \( \sqrt{x^2 + y^2} \).

Popular kernel choices are presented in Table 1.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Form</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular</td>
<td>( \frac{1}{2} )</td>
<td>(-1 \leq x \leq 1)</td>
</tr>
<tr>
<td>Triangular</td>
<td>( 1 -</td>
<td>x</td>
</tr>
<tr>
<td>Epanechnikov</td>
<td>( \frac{3}{4} (1 - x^2) )</td>
<td>(-1 \leq x \leq 1)</td>
</tr>
</tbody>
</table>
Biweight \[ \frac{15}{16} (1 - x^2)^2 \quad -1 \leq x \leq 1 \]

Triweight \[ \frac{35}{32} (1 - x^2)^3 \quad -1 \leq x \leq 1 \]

Gaussian \[ \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{1}{2} x^2 \right) \quad -\infty \leq x \leq \infty \]

In practice, good bandwidth selection has a much greater impact on the density estimation than kernel choice.

We limit the support of \( f_{G_i,D_i}(g,d) \) to the range of possible measurements (positive and above the baseline). That is:

\[ \text{supp}(f_{G_i,D_i}) \subseteq \{(g,d): g \geq 0 \text{ and } d \geq mg\} \quad (13) \]

Where \( m = 1/100 \) is the slope of the baseline.

By increasing the number of active and/or passive landmarks the accuracy of density estimation increases.

**Step 3.** Calculate the conditional probability of the distance \( G_i \) given the delay \( D_i \).

\[ f_{G_i|D_i}(g \mid d) = \frac{f_{G_i,D_i}(g,d)}{f_{D_i}(d)}, \quad (14) \]

where \( f_{D_i}(d) \) is the marginal distribution of \( D_i \):

\[ f_{D_i}(d) = \int_0^{d/m} f_{G_i,D_i}(g,d) dg \quad (15) \]

**Step 4.** For a measured target \( \tau \), let \( d_{i_\tau} \) be the delay measured from landmark \( L_i \) to the target \( \tau \). Thus we have a set of probability density functions:
Our objective is to find a geographical location for \( \tau \), which maximizes most of these conditional probability density functions or likelihoods. For a given longitude and latitude \((\lambda, \phi)\) of the target it is easy to calculate the geographical distances \(g_{i\tau}\) to each landmark \(L_i\). The only remaining problem is how to combine the evidence given by the conditional distribution of distance to landmarks given the delay. There are several ways these probability density functions can be combined to estimate the target location:

- By maximizing the product of the probability density functions:

\[
F(\lambda, \phi) = f_{G_i|D_i}(g_{i\tau} | d_{i\tau}) \cdot f_{G_2|D_2}(g_{2\tau} | d_{2\tau}) \cdot \cdots \cdot f_{G_n|D_n}(g_{n\tau} | d_{n\tau})
\]

This approach suffers from a lack of robustness: one misestimated zero probability density function would nullify the whole product.

If we assume that \(G_i|D_i\) are independent for \(i = 1, 2 \ldots n\), finding values of \(\lambda\) and \(\phi\) that maximize \(F\) gives the maximum likelihood estimation of target location.

- By maximizing the weighted sum of the probability density functions

\[
F(\lambda, \phi) = f_{G_1|D_1}(g_{1\tau} | d_{1\tau}) + f_{G_2|D_2}(g_{2\tau} | d_{2\tau}) + \cdots + f_{G_n|D_n}(g_{n\tau} | d_{n\tau})
\]

By maximizing a fuzzy measure of the probability density functions (such as fuzzy union or fuzzy intersection)

5.1. **Algorithm for maximization of \(F(\lambda, \phi)\)**

\(F(\lambda, \phi)\) can be maximized by doing an exhaustive search. For example, the area of interest can be divided into a grid \(\lambda_1 < \lambda_2 < \ldots < \lambda_m\) and \(\phi_1 < \phi_2 < \ldots < \phi_n\), and \(F(\lambda_i, \phi_j)\) calculated in each point of the grid.
Alternatively, we can use an optimization algorithm to search for a local maximum. An example of such an algorithm is a force-directed algorithm. The following algorithm combines gradient ascent and force-directed ideas. Unlike the genuine force-directed algorithms, it optimizes the position of the target only. The ascent step, $\eta_i$, is a decreasing sequence converging to zero.

Let $(\lambda_i, \phi_i)$ be the longitude and latitude of the active landmarks $L_i, i = 1,2 \ldots n$. The algorithm proceeds as follows:

**Step 1.** Start with a guess of the longitude and latitude of the target $(\lambda_\tau^{(0)}, \phi_\tau^{(0)})$.

Educated guesses for target location can be given by a previous method with low computational complexity, such as Shortest Ping or GeoPing.

Initialize $k = 0$

**Step 2.** Calculate the distance from the target to landmarks:

$$g_i^k = \sqrt{(\lambda_i - \lambda_\tau^{(k)})^2 + (\phi_i - \phi_\tau^{(k)})^2}, \text{ for } i = 1,2\ldots n$$

(19)

**Step 3.** Execute one step of gradient ascent:

$$h_i^{(k)} = g_i^{(k)} + \eta_k f_{G_i|D_i} (g_i^{(k)} | d_i), \text{ for } i = 1,2\ldots n$$

(20)

**Step 4.** For each $i=1,2 \ldots n$

If $f_{G_i|D_i} (h_i^{(k)} | d_i) > f_{G_i|D_i} (g_i^{(k)} | d_i)$ (bigger likelihood) then apply force $F_i$ of magnitude $h_i^{(k)} - g_i^{(k)}$ on the direction $L_i \rightarrow \tau$

Else $F_i=0$

**Step 5.** Calculate the resultant force (using parallelogram’s rule)
$$\mathbf{F} = \sum_{i=1}^{n} \mathbf{F}_i$$  \hspace{1cm} (21)

**Step 6.** Move the target $\tau$ on the direction of the resultant force

$$(\lambda_{\tau}^{(k+1)}, \varphi_{\tau}^{(k+1)}) = (\lambda_{\tau}^{(k)}, \varphi_{\tau}^{(k)}) + \mathbf{F}$$ \hspace{1cm} (22)

If the target was moved with more than $\varepsilon$, which is

$$\sqrt{(\lambda_{\tau}^{(k+1)} - \lambda_{\tau}^{(k)})^2 + (\varphi_{\tau}^{(k+1)} - \varphi_{\tau}^{(k)})^2} > \varepsilon$$

Then $k = k + 1$ and go back to step 2.

Else STOP
Chapter 6. Improvements over GeoPing

Padmanabhan and Subramanian [8] attempt to geolocate the Internet hosts by calculating the minimum Euclidean distance in the delay space. Katz-Basset takes a different approach, by considering the shortest ping measure, which is the shortest round trip time. One direction of this thesis to improve the geolocation efficiency by investigating a larger class of distances or other proximity measures in the delay space, instead of Euclidian distance, including:

- P-norms, which are defined by:

\[
\|x\|_p = \left( |x_1|^p + |x_2|^p + \cdots + |x_n|^p \right)^{1/p}
\]

(23)

With the corresponding distance measure:

\[
d(x, y) = \|x - y\|_p
\]

(24)

For \( p=1 \) we get the so-called “taxicab” or “Manhattan” distance. For \( p=2 \) we get the Euclidean distance, used by the GeoPing authors. For \( p \to \infty \) we get the infinity or the maximum norm:
\[ \| \mathbf{x} \|_e = \max \{ |x_1|, |x_2|, \ldots, |x_n| \} \]  

(25)

With the corresponding distance measure:

\[ d(\mathbf{x}, \mathbf{y}) = \max \{ |x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n| \} \]  

(26)

The Mahalanobis norm, a different generalization of Euclidean distance, defined by:

\[ \| \mathbf{x} \|_d = \mathbf{x}^T \mathbf{A} \mathbf{x} \]  

(27)

With \( \mathbf{A} \) being a positive defined matrix. The corresponding Mahalanobis distance is:

\[ d(\mathbf{x}, \mathbf{y}) = \| \mathbf{x} - \mathbf{y} \|_d = (\mathbf{x} - \mathbf{y})^T \mathbf{A} (\mathbf{x} - \mathbf{y}) \]  

(28)

If \( \mathbf{A} \) is a diagonal matrix

\[
\mathbf{A} = \begin{bmatrix}
  w_1 & 0 & \cdots & 0 \\
  0 & w_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & w_n
\end{bmatrix}
\]  

(29)

Then the Mahalanobis distance reduces to a weighted Euclidean distance:

\[ d(\mathbf{x}, \mathbf{y}) = \sqrt{w_1 (x_1 - y_1)^2 + w_2 (x_2 - y_2)^2 + \ldots + w_n (x_n - y_n)^2} \]  

(30)

The matrix \( \mathbf{A} \) is usually taken to be the inverse covariance matrix \( \mathbf{\Sigma} \) calculated from the empirical data. Thus the Mahalanobis distance formula is:

\[ d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{y}) \]  

(31)

This formula accounts for the variance and covariance of the different components of the distance.

Robust proximity measures
Using the Euclidean distance, or any other distance above (as in Geoping) has the obvious drawback where even one unreliable measurement can cause the distance in the delay space to become very large. Two very common causes of unreliable measurements are congestion and re-routing. Thus, we might want to keep only $m$ out of the total of $n$ components weight down, or even ignore the largest $m-n$. Thus, if $c_1 = |x_1-y_1|$, $c_2 = |x_2-y_2| \ldots c_n = |x_n-y_n|$, and $c_{(1)} < c_{(2)} < \ldots < c_{(n)}$ are the ordered values of $c_1$, $c_2 \ldots c_n$, we can take the proximity measure between $x$ and $y$ as:

$$d(x,y) = \sqrt{c_{(1)}^2 + c_{(2)}^2 + \ldots + c_{(m)}^2}$$

(32)

The components $c_{(m+1)}$, $c_{(m+2)} \ldots c_{(n)}$ have been ignored. This proximity measure is not a distance, as it does not satisfy the triangle inequality. However, it is more accurate in unreliable environments.
Chapter 7. Localization in the Internet

Let \( \{{L_1, L_2 \ldots L_n}\} \) be the set of landmarks whose location \( \mathbf{p}_i = \{ (\lambda_i, \varphi_i) \}^T | i = 1, 2 \ldots n \) are known. We are interested in estimating the unknown position \( \mathbf{p}_t = (\lambda_t, \varphi_t)^T \) of a particular Internet host by means of time delay measurements for each landmark to the target host. The time \( d_{it} \) taken by a packet to be transmitted from the landmark \( L_i \) to the target is modeled as follows:

\[
d_{it} = \frac{1}{\tilde{c}} g_{it} + q_i
\]  

(33)

Where \( d_{it} \) is a distance function, which provides a measure of the distance between landmark \( L_i \) and the target \( \tau \) and where \( \tilde{c} \) is taken as an approximation of the propagation speed of light through optical fiber:

\[
\tilde{c} = \frac{2}{3} c
\]  

(34)

And \( c = 3 \times 10^8 \) m/s is the speed of light. In the absence of prior knowledge of the network topology, Euclidean distance can be used to define the distance function:
\[ g_{ir} = \sqrt{(\lambda_i - \lambda_r)^2 + (\varphi_i - \varphi_r)^2} \] (35)

In the remainder of this section, we shall assume the distance function based on Euclidean distance, as given in (35).

The first term in (33) represents the propagation time of a packet from the landmark \( L_i \) on a great-circle path through an optical fiber link. The second term, \( q_i \), accounts for addition delays incurred by processing delay, queuing delay, and path deviations from the great circle path. We model \( q_i \) as a random parameter with known prior distribution. The observed or measured time delay from the target host to landmark \( L_i \) is modeled as:

\[ r_{ir} = d_{ir} + n_{ir}, \quad i = 1, 2, \ldots n \] (36)

Where \( n_{ir} \) is a zero-mean Gaussian random variable, \( n_{ir} \sim N(0, \sigma_i^2) \), which represents measurement noise. Let \( \mathbf{r} = (r_{ir} \mid i = 1, 2, \ldots n)^T \) denote the vector of observations and let \( \mathbf{q} = (q_i \mid i = 1, 2, \ldots n)^T \) denote the vector of delay parameters. Next, define the vector of unknown parameters:

\[ \mathbf{\theta} = \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} \] (37)

The pdf of the observation vector \( \mathbf{r} \) conditioned on \( \mathbf{\theta} \) is denoted by \( f_\theta(\mathbf{r}) \).

The prior distribution of \( \mathbf{q} \) is denoted by \( p_\mathbf{q}(\mathbf{q}) \). Let \((G_i, D_i)\) represent the distance \( G_i \) and the time delay \( D_i \) from landmark \( L_i \) to a target \( r_i \). By collecting empirical delay measurements between the landmarks, whose locations are known, the joint density
$f_{g,d}(g,d)$ can be estimated using kernel density estimation techniques, as described in Chapter 5. From $f_{g,d}(g,d)$, the prior distribution $p_q(q)$ can be estimated.

### 7.1. Generalized Cramer-Rao Bound

Let $\hat{\theta}$ denote an unbiased estimate of the parameter $\theta$. The generalized Cramer-Rao Bound (CRB) sets a lower limit for the covariance matrix of the error vector $\hat{\theta} - \theta$ as follows:

$$E_{\theta}[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T] \geq J^{-1},$$

where the matrix $J$ is called the “generalized Fisher information matrix (G-FIM)” and consists of two components

$$J = J_D + J_P$$

Here, the statement $A \geq B$ should be interpreted as saying that the matrix $A - B$ is nonnegative definite. The subscripts “D” and “P” stand for “data” and “prior” information respectively. The component $J_D$ is given by:

$$J_D = E\left[\frac{\partial}{\partial \theta} \log f_\theta(r) \cdot \left(\frac{\partial}{\partial \theta} \log f_\theta(r)\right)^T\right],$$

where the expectation is taken over both $r$ and $q$. The component $J_P$ is defined by
\[
J_p = E \left[ \frac{\partial}{\partial \theta} \log p_\theta(\theta) \cdot \left( \frac{\partial}{\partial \theta} \log p_\theta(\theta) \right)^T \right],
\]  
(41)

where \( p_\theta(\theta) \) is the prior distribution of \( \theta \). Since \( \theta = (p, q)^T \) and \( p \) is an unknown constant, the random component of \( \theta \) is given by \( q \). Hence we can write:

\[
p_\theta(\theta) = p_q(q)
\]  
(42)
Chapter 8. Future Experiments and Model Validation

For the probabilistic algorithm, we intent to use the PlanetLab testbed. First,
APPENDIX

"{Click here to add appendix}"
REFERENCES


CURRICULUM VITAE

"{Click here to add single spaced text}"