RandPing: A Randomized Algorithm for IP Mapping

Yuhan Cai
Department of Computer Science & Engineering
University of Washington
Seattle, WA, USA
yuhancai@cs.washington.edu

Jing Liu
Department of Computer Science & Engineering
University of Washington
Seattle, WA, USA
liujing@cs.washington.edu

ABSTRACT

Mapping IP addresses to geographic locations is a useful and important operation that can benefit a broad range of large-scale location-aware applications for both research and industrial organizations. In this paper, we develop a computational framework called *RandPing* that automates IP mapping based on a randomized algorithm using network delay measurements made from sparsely geographically located probing machines. Indeed, there exists previous work that is also based on the knowledge of network delay. However, the algorithms they use are deterministic in nature, which means that, the Round Trip Time (RTT) between the target host and any single probing machine has to be measured. On the other hand, in our framework, we choose non-deterministically (or randomly) a relatively small set of probing machines from a few "well-defined" clusters, program them to ping the query IP address, and repeat the process for a limited number of times. This will effectively reduce the time spent on pinging and therefore is cost-effective. After that, a dynamic programming technique is used to obtain the group of circles with the maximum likelihood to contain the target, and then we formulate and solve the final point estimation as a non-linear programming problem. We also hypothesize that there is no statistically significant difference between the accuracies of the deterministic and non-deterministic approaches. Furthermore, our claim is strengthened by our experimental results in terms of both efficiency and effectiveness.

1. INTRODUCTION

Where in the world is 18.31.0.1? How many people in Seattle have visited my homepage? And how can I limit the traffic to my site from Europe? All these questions entail the concept of geolocation. Geolocation (or IP mapping) refers to the problem of finding the geographic location of an internet host given only its IP address. Formally defined,

DEFINITION 1. **IP Mapping** *is the problem that, given an IP address p, find the geographic location (longitude/latitude pair) of the internet host with IP address p. The address p is called a query IP address.*

Mapping IP addresses to physical locations is not only an interesting problem in its own right, but also playing an important role in many large-scale applications. For example, a Web server could send personalized information (such as targeted advertisement, local weather, and webpages in the regional language) to the clients based on their locations. For another example, we can also restrict the availability of data and authority of transactions according to the locations of users. For yet another example, visitors to a website can be classfied with respect to where they physically are, and this piece of information will be useful when designers try to improve their sites.

On the other hand, however, IP mapping is a challenging problem due to the following reasons:

- IP addresses and even host names are values in a logical hierarchy, so they do not inherently contain any geographic information.
- There is no authorative database that maps IP addresses to physical locations.
- An automated technique is desired, as the large size of the typical datasets used in the search has made a manual mapping impractical and imprecise.

In this paper, our goal is to provide an efficient and effective methodology that is able to predict the physical location corresponding to any IP address within a reasonable accuracy. The location estimation is built on top of the assumption that the network delay experienced by packets traveling between a pair of hosts is a function of the geographic distance between the hosts. Existing work based on network delays relies on a large number of probing machines with given locations as the source of information. In contrast, we need only a small subset of those machines to provide fast derivations of physical locations from IP addresses with an acceptable accuracy under the help of clustering and outlier detection techniques.

As a preview, we make the following contributions in this paper:

- We present a randomized algorithm that avoids exhaustive employment of probing machines and guarantees an minimal reduction in accuracy.
- We use a dynamic programming technique to deduce the intersection region of circles that has the maximum

probability to contain the query point. Additionally, the final point estimation is formulated as a non-linear programming problem in which a Steiner Tree is constructed with an attempt to minimize the error distance.

- We use geometric identities to prune out invalid delay measurements so that the delay-to-distance relationship is as precise as possible. For instance, if the difference between the delay measures (not distances) from two probing machines to the target host is smaller than the delay measures between the two machines themselves, obviously one or both of those measures must be wrong.

- The quality of the randomized framework is further enhanced by data mining techniques such as clustering/classification and outlier detection.

The rest of this paper is organized as follows. In Section 2, we go over related work on IP mapping. In Section 3, we review background knowledge (clustering, outlier detection and the scriptroute system) that is central to the development of this paper. After that, in Section 4, we present our core algorithm that deduces geographic information from IP address information. Next, we present our experimental setup and results in Section 5. And finally, Section 6 concludes this paper.

2. RELATED WORK

Existing work on IP mapping can be classified into the following three categories:

- The first approach to this problem is based on Domain Name Servers (DNS). To obtain the DNS of a host given an IP address, there is a bunch of tools using DNS records either from internal databases or querying external databases like Whois. Sample tools are IP2LL [10] and NetGeo [4]. There are several limitations with this approach. First, the DNS records could be stale or inaccurate in the databases. Second, if hosts are behind a proxy, it might be a single entry of DNS record for the whole block of IP addresses. Third, the registered domain names in the databases may or may not be related to the physical locations of the machines. For example, large companies whose machines are distributed across many states or even countries typically register their domain names with the address of their corporate headquarters, regardless of the actual locations of the machines.

  Some researchers have proposed a way of retrieving the DNS name of a host or a nearby node by tracing DNS names of routers along a path from the source to the target IP address. This technique is used in GeoTrack [6], VisualRoute [11], and GTrace [1]. In this way, either the DNS name of the target host or the DNS name of the last recognizable router in the path toward the host will be retrieved. This solution avoids using the stale and inaccurate DNS records from databases. However, there are still some drawbacks with this solution. Since the position of the last recognizable router in the path toward the host to be located is used to estimate the position of the host and this router might not be geographically close to the host, a lack of accuracy is expected.

- An alternative approach is based on exploiting the possible correlation between geographic distance and network delay. Examples include GeoPing [6] and Constraint-Based Geolocation (CBG) [2]. In GeoPing, some fixed probe machines with known geographic locations are used to estimate the distances to the target machine by network delays to the target machine. CBG improves GeoPing in two ways. In CBG, a new technique called multilateration is used. Multilateration refers to the process of estimating a position using a sufficient number of distances to some fixed points. In effect, CBG has to use a larger number of probe machines. Furthermore, CBG transforms delay measurements into distance constraints. By Combining both multilateration and distance constraints, CBG is able to establish a geographic region that contains the target host. Experiments have shown that CBG delivers much better accuracy than GeoPing does, however, by using multilateration, more probe machines are required, which has made CBG a non-cost-effective method.

- The third approach splits the IP address space into clusters. It is based on the assumption that all hosts with IP addresses within a cluster are likely to be colocated. Thus, knowing the location corresponding to a few hosts in a cluster, the location of the entire cluster can be deduced. GeoCluster [6] is such a technique. There are two disadvantages with this approach, which leads to inaccuracy. First, the approach relies very much on the IP-to-location mapping information of some of the hosts in the clusters. The information may be inaccurate since the data in the databases are provided by the users, which may be unreliable. Second, the location information is from a relatively small subset of the IP address space. The inaccuracy is unavoidable.

In addition, there exist two approaches to associate geographic distances with network delays:

- **Nearest neighbor in delay space (NNDS)**: NNDS is motivated by the observation that hosts with similar network delays with respect to other fixed hosts tend to be located near each other. Thus, the first step is to construct a delay map that records the relationship between delay and location. Each entry of the delay map contains the coordinates of a host at a known location and a delay vector, $DV = (d_1, \ldots, d_N)$, containing the measured minimum delay to the host from N probing machines at known locations. Given a new target host T, a delay vector DV' with delays to N probing machines will be obtained through experiments, where $DV' = (d'_1, \ldots, d'_N)$. Eucledian distance will be used as the measure of distance in delay space. Once the nearest neighbor in delay space has been found, the corresponding location recorded in the delay map is the estimate of the location of the target host T.

- **Best Line Bound (BLB)**: BLB gives a mathematic model on the correlation between network delay and geographic distance. In the ideal case, the delay is only propagation delay and digital information travels along fiber optic cables at almost exactly 2/3 the speed of light in vacuum. Thus, we should have a straight line comprising this relationship that is given by the slope-intercept form $y = mx + b$, where $b = 0$ since there are no localized delays. However, in practice, there is delay distortion from queuing delay and lack of great-circle paths between hosts. Thus, a best-line is defined for each landmark L_i, which is a probing machine. The line $y = m_i x + b_i$ is closest to, but below, all data points (x, y) and has non-negative intercept. The finding of the best line is formulated as a linear programming problem. For a given landmark L_i, there are the network delay d_{ij}, and the geographic distance g_{ij} toward each landmark L_j, which is a different probing machine from L_i. Thus, d_{ij} and g_{ij} could be used by linear programming to obtain m_i and b_i. When the best line for L_i is found, the distance between target machine and the probing machine L_i could be obtained. We will address this issue in details later.

3. BACKGROUND

In this Section, we review the concepts of clustering (or classification) and outlier detection, and the scriptroute system. These are central to the development of the following sections.

3.1 Clustering/Classification

In the past few decades, cluster analysis has been intensely studied and widely applied to many areas such as data mining, medicine, chemistry and social studies [5]. The main goal is to identify structures or *clusters* present in the dataset, or in other words, *classify* data points into categories. Existing clustering algorithms can be classified into two main categories: hierarchical methods and partitioning methods. We choose a $k - medoid$ partitioning method as the basis for our algorithm for the following reasons:

- It is robust to outliers.
- Clusters found by a k-medoid method do not depend on the order in which the objects are examined.
- Experiments have shown that it can handle large datasets quite efficiently.

Formally speaking, the clustering problem is that, given a set S of N objects, a distance metric d and a positive number k, find the optimal partition of S into k groups with respect to a cluster quality metric. The algorithm we focus on is called Partitioning Around Medoids (PAM). To find k clusters, PAM attempts to determine a representative object for each cluster. This object, called a *medoid*, is supposed to be the most centrally located object in each cluster. Once the medoids are selected, every other object is grouped with the medoid to which it is most similar. If O_j is a non-selected object and O_m is a medoid, we say that O_j belongs to the cluster represented by O_m if $d(O_j, O_m) = min_{O_e} d(O_j, O_e)$, where min_{O_e} is the minimum over all medoids O_e and $d(O_1, O_2)$ is the distance between objects O_1 and O_2. Finally, the *quality* of a *clustering* is measured by the average distance between an object and the medoid of its cluster. To find the k medoids, PAM starts with an arbitrary selection of k objects, and swaps a selected object O_m and a non-selected object O_p in each step, as long as such a swap would result in an improvement of the quality of the clustering.

3.2 Outlier Detection

While there is no single formal definition of an outlier, Hawkins' definition capture the spirit: "an outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [3]. The notation studied in this paper is defined as follows:

DEFINITION 2. *An object O in a dataset T is a $DB(p, D)$-***outlier***, if at least fraction p of the objects in T lies greater than distance D from O.*

Here, the term $DB(p, D)$-outlier represents a *D*istance-*B*ased outlier detected using parameters p and D. This definition is intuitive and consistent with Hawkins' definition. It is suitable for situations where the observed distribution does not fit any standard distribution.

3.3 Scriptroute System

Scriptroute is a system that allows ordinary Internet users to conduct network measurements from remote vantage points. To use Scriptroute, clients use DNS to discover measurement servers and then submit a measurement script for execution in a sandboxed, resource-limited environment. An API is included in the sandbox for sending and receiving measurement packets and the results will also be reported back to the client by the API. Scriptroute code is publicly available and has been deployed on the PlanetLab testbed of 42 sites. There is a variety of measurements supported by Scriptroute, including hop-by-hop bandwidth, bottleneck bandwidth, round trip time and congestion. While Scriptroute is a flexible measurement tool, it also protects the servers from exposing their host to unwanted attacks. To isolate measurements from the host system, servers execute measurement scripts in a very strong sandbox that provides only a narrow interface for sending and receiving packets and communicating results to the client. Moreover, to ensure that the measurement scripts do not consume enough resources to cause denial-of-service to the host, the Scriptroute server limits all aspects of measurement execution. Servers limit the duration, traffic rate, memory footprint, processor time, and number of concurrent measurements, reclaiming their resources as scripts terminate. Scriptroute system includes the following components: system management, server front-end, script interpreter and network guardian. System management maintains a dynamically updated DNS database of server list. This allows clients to find Scriptroute servers using descriptive host names and servers to publish their feature set. Server front-end maintains three main "pages": job submission, traceback, and informational. Script interpreter runs the scripts. Sendtrain API is used by interpreter to send probes and receive responses. Network guardian is the only component which can access the network, after first checking that a destination filter does not block requested probes.

4. ALGORITHM OVERVIEW

In this Section, we present an overview of our randomized algorithm. Figure 1 summarizes all the symbols used in this paper, and Figure 2 gives an overview of our algorithm *RandPing*.

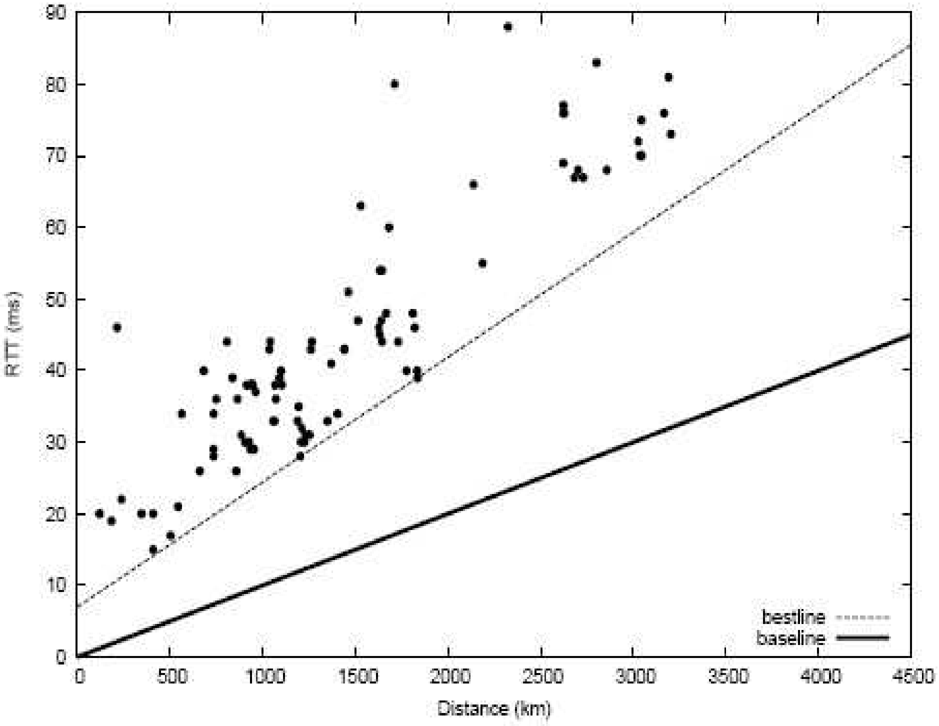

Figure 3: RTT VS. Physical Distance

Notations	Meanings
p	Query IP address
T	Set of probing machines
N	Number of probing machines
k	Number of clusters
l	Number of clusters picked
Cl_i	The i^{th} cluster picked
m	Number of machines picked from each cluster
r	Number of randomized runs
ED	Error Distance

Figure 1: Summary of Notations

Our algorithm can be ordered into 3 stages:

4.1 Preprocessing Stage

Prior to processing any queries, we do some preliminary setup work. Note that, this job is done once-and-for-all and therefore should not be included in the computation of query costs.

First of all, we want to establish a relationship between network delays (or RTTs) and geographic distances. Recall that in Section 2 we outline two existing methods to derive geographic distances from network delays. However, the first method, NNDS, has a drawback in that, the accuracy depends heavily on the delay space. In the case that the target host is not close to the location of any known hosts in the delay space, this method will lose accuracy. In our experiments, we will not restrict the target host to be close to any of location known hosts, and therefore will use the second method (BLB) instead.

Ideally, the relationship between RTT and geographic distances should be determined solely by the speed of light in fiber optic cables, which is equal to 2/3 of that in vacuum. Thus, we should have a straight line $y = mx + b$ where y is the RTT variable, x is the distance variable, $m = 1ms/100km$, and $b = 0$. This line is a strict lower bound for those cases in the real world, as there are always noises and extra sources of delays in practice. It is shown as the "base line" in Figure 3.

For each probing machine M_i in the set T, we use it to ping every other machine $M_j (i \neq j)$ in T. We will then have a RTT table $D = \{d_{ij}\}_{i \neq j}$ that contains the RTT measurement between every pair of machines. Additionally, as these are machines with known geographic locations, based on their longitude and latitude information, we can construct

Algorithm RandPing(p, T, k) {
/* input: a collection T of probing machines
with known IPs and locations */
/* input: number of clusters, k */
/* input: target IP address p */
/* output: location of p */
(1) Initial setup: run PreProcessing (T, k) in Figure 4
to perform clustering and compute the best line
of each probing machine.
(2) Run Pinging ($\{Cl_i\}_{i=1}^{k}, p$) in Figure 5 for r trails.
(3) Use algorithms in Figure 6 and Figure 8
to compute r point estimates.
(4) Prune outliers and return the centroid of the rest of points.
} /* end algorithm */

Figure 2: ***RandPing*** **Overview**

```
Algorithm Preprocessing(T, k) {
  /* input: a collection T of probing machines
     with known IPs and locations */
  /* input: number of clusters, k */
  /* output: best line of each probing machine */
  /* output: T is divided into k well-defined clusters */
  (1)  Construct the RTT table D and distance table G
       by programming each machine to ping every other
       machine.
  (2)  Compute the best line using linear programming
       based on Equations 1, 2, 3 and 4.
  (3)  Apply PAM (as described in Section 3.1) to T.
} /* end algorithm */
```

Figure 4: Algorithm for the preprocessing stage

```
Algorithm Pinging ({Cl_i}_{i=1}^k, p) {
  /* input: k clusters {Cl_i}_{i=1}^k of probing machines
  /* input: target IP address p
  /* output: a set of circles representing the estimated regions */
  (1)  Select l clusters randomly from the k clusters.
  (2)  Pick randomly m points from each cluster.
  (3)  Let them ping the target IP p.
  (4)  Compute the estimated distances based on
       the best line of each machine and the RTT measurements.
  (5)  Return the ml circles centred at the locations of
       those machines with radii being the distance estimates.
} /* end algorithm */
```

Figure 5: Algorithm for the pinging stage

a distance table $G = \{g_{ij}\}_{i \neq j}$ that contains the geographic distance between every pair of machines.

As in BLB, we assume that the combination of all sources of extra delays incurs only a pure geometric distortion factor of the RTT. Therefore, for each probing machine M_i, we define its "best line" as $y = m_i x + b_i$ that is closest to, but below, all data points (g_{ij}, d_{ij}) and has a non-negative intercept. The distance between each data point and the best line represents the existence of some additive distortion with respect to the best-observed case. And the region between the best line and the base line corresponds to the observed gap between the current relationship and the ideal case.

Our goal is to find the best line for each machine M_i, that is, we want to find the pair (m_i, b_i) for each M_i and this can be easily formatted into a linear programming problem.

The linear function we want to minimize is

$$\Sigma_{i \neq j}(d_{ij} - m_i g_{ij} - b_i), \tag{1}$$

subject to the constraints:

$$b_i \geq 0, \tag{2}$$

and

$$m_i \geq m, \tag{3}$$

and

$$d_{ij} - m_i g_{ij} - b_i \geq 0. \tag{4}$$

For each machine M_i, based on the above three equations, we can compute its best line $y = m_i x + b_i$. Now, given a RTT measurement d_i from machine M_i, the estimated geographic distance g_i is:

$$g_i = \frac{d_i - b_i}{m_i}. \tag{5}$$

Additionally, we perform PAM on the collection T of probing machines and divide them into k clusters. This step guarantees that, in the next step when we pick random machines, they are not going to be very close to each other, which can potentially improve the accuracy of our algorithm.

The preprocessing stage is summarized in Figure 4.

4.2 Randomized Pinging

The current subsection and the next one deal with the actual processing of queries. Figure 5 gives a sketch of our randomized pinging stage. The input is the N probing machines clustered into k clusters, and the query IP address p. Our goal is to locate the target in terms of a longitude/latitude pair.

First of all, we transform the geographic system to a Cartesian coordinate system, that is, we convert each longitude/latitude pair into a (x, y) coordinate with respect to a point with the smallest longitude and smallest latitude as the origin. This step can be easily accomplished using spherical coordinates as follows.

For a point with longitude G, and latitude T, its coordinate (x, y) is given by:

$$x = \frac{2\pi R \; cos T_0 \; (G - G_0)}{360} \tag{6}$$

$$y = \frac{2\pi R \; (T - T_0)}{360} \tag{7}$$

where $G_0 = -130^o$ and $T_0 = 25^o$ are the longtitude and latitude of the most southwest point in the United States.

Next, we select l clusters randomly from the k clusters ($l < k$) constructed from Section 4.1. Thirdly, we pick randomly m points (probing machines) from each of the l picked clusters, and program them to ping the target IP address p. Those l clusters are expected to distribute relatively "sparsely" over the investigated geograhic area, and therefore the ml points selected can not be too "close" to each other. We will then have a collection of ml RTT measurements. For the RTT measurement d_i from the i_{th} machine M_i, based on the best line of M_i (computed in the previous section), we can estimate the physical distance g_i from M_i to p, and thus a circular region C_i with radius $r_i = g_i$ centered at (x_i, y_i), where (x_i, y_i) is the coordinate of M_i. Ideally, those circles would intersect at one and only one point, which gives the exact location of the target machine. Of course, in the real case, as r_i is always expected to be an overestimate of the real distance between (x_i, y_i) and p, p must reside in each C_i, and we will have a region of possibilities for the target instead of one single point. This scenario is illustrated in Figure 7.

4.3 Location Estimation

Given this irregular intersection region of circles, where could the target host be? The key intuition is that, the largest cluster of circles in which every pair intersects would have the maximum likelihood to contain the target, as the number of outlying circles is assumed to be minority. Figure 6 summarizes our algorithm to find the best group of circles using dynamic programming. Basically, we start with groups of size 2 and build them up incrementally. In the i_{th} iteration, for each group b in the candidate list BG, we bring in another circle c from the "unprocessed" list PC, if c intersects all circles in group b, we add c in b. Note that, if a circle c is not added to any group in the i_{th} iteration,

```
Algorithm BestGroup (GC) {
  /* input: a collection GC of circles
     with known radii and centers */
  /* output: the largest group of circles
     in which every pair of circles intersects */
  (1) Let BG = {{c_i, c_j}|i ≠ j and c_i ∈ GC and c_j ∈ GC intersect}
      Let PC = {c_i|∃g ∈ BG such that c_i ∈ g}
  (2) while PC ≠ φ and foreach l ∈ PC {
        (3) flag = true;
        (4) foreach b ∈ BG {
              (5) if (l ∉ b and ∀c ∈ b, c intersects l) {
                    (6) flag = false;
                    (7) set b = b ∪ {l};
                  } /* end if */
            } /* end for-loop */
        (8) if (flag) PC = PC − {l};
      } /* end while */
  (9) Return the group in BG that has the largest size.
} /* end algorithm */
```

Figure 6: Algorithm for Selecting the Best Group of Circles

```
Algorithm BestPoint(BG) {
  /* input: the "best" group of circles BG */
  /* output: a point estimate for the target machine */
  (1) Let [point, EXITFLAG] = Minimize Equation 8
      subject to the constraints in Equation 9.
      (EXITFLAG = 0 iff there is no solution.)
  (2) while (EXITFLAG == 0) {
        (3) Remove randomly a circle from BG.
        (4) Let [point, EXITFLAG] = Minimize Equation 8
            subject to the constraints in Equation 9.
      } /* end while */
  (5) return point.
} /* end algorithm */
```

Figure 8: Algorithm for finding the best point

it will not possibly contribute to the $(i+1)_{st}$ iteration, and therefore can be removed from PC.

After we find the "best" group BG of circles, we drill down and aim for a single point estimation. In essence, we want this point to be within each of the circles in the group and compute a Euclidean Steiner tree [8]. That is, we seek to minimize the sum of the Euclidean distances between the point and the center of each circle. We formulate this as a non-linear programming problem as follows:

The object function to minimize is:

$$\Sigma_i[(x - x_i)^2 + (y - y_i)^2], \tag{8}$$

subject to the constraints:

$$\forall i, (x - x_i)^2 + (y - y_i)^2 \leq r_i^2. \tag{9}$$

One thing to note is that, among the circles in BG, they do not necessarily share a common region, although they mutually intersect with each other. The way in which we cope with this problem is that, we iteratively remove a randomly chosen circle from BG and try to solve the modified optimization Equation 8 together with a modified set of constraints (Equation 9), until we finally find a solution. This process is guaranteed to terminate if $|BG| = 2$ at the latest. See Figure 8 for details.

Up till now, we have only dealt with one randomized trail. We shall repeat this process for r times, and therefore obtain r point estimates for the target host p. Most of the estimates are supposed to be clustered around one area, while a few of them are outlying in the sense that too much error is involved in the process of estimation. We use the definitions in Section 3.2 to prune out those distance-based outliers (using sequential search) and obtain a set of points S. Suppose $S = \{s_1, s_2, \ldots, s_l\}$, then we define:

$$E(p).x = \frac{1}{l}\Sigma_{i=1}^{l} s_i.x \tag{10}$$

and

$$E(p).y = \frac{1}{l}\Sigma_{i=1}^{l} s_i.y \tag{11}$$

where $E(p)$ is our final point estimate for the target host p, which is simply taken as the centroid of S.

One point to note is that, we can use triangular inequality to disqualify distance estimations. Suppose two probing machines M_1 and M_2 have a known distance of d_1 and their estimated distances to a target host is d_2 and d_3, respectively. If $|d_2 - d_3| \geq d_1$, then one or both of those estimates must be invalid, because triangular inequality is violated.

Finally, as a simple example of how much saving we could have using our randomized algorithm, suppose there are 500 probing machines available, and we pick 10 clusters, 2 points from each cluster, and repeat the process for 5 times. In total, 100 delay measurements are going to be made in our non-deterministic algorithm, while 500 measurements are to be made for the deterministic one. Therefore, a performance upgrade of 5 times is to be expected.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

Because of the geolocation nature of our work, we need datasets with machines whose geographic locations are well known. This would allow us to compare the location estimations provided by *RandPing* with the real locations of target machines. We use PlanetLab machines in North America as both target machines and probing machines, and we have their exact physical locations and IP addresses. The main reason why we do this is that, all the datasets we have consist of mostly host machines located in North America and we think that they are representative enough in that the America is holding a large proportion of the Internet infrastructure.

Since we only have the machines on PlanetLab that can accept our pinging scripts, we choose a small subset of machines located in the United States as our targets, whose locations will be estimated by our algorithm and then compared with their real locations, while we keep the rest of them as probing machines, which will run our pinging scripts to collect all the necessary data for the computations. With an attempt for a fair evaluation, we pick target machines distributed all over the States and switch the roles of a machine in adjacent trials of our experiment.

In order to evaluate the effectiveness of our algorithm, we define the accuracy of geolocation as follows:

Definition 3. **Error Distance** *ED of an geographic estimation is the physical distance between the real location of a host and the estimated location of it.*

The smaller the Error Distance of an estimation, the better accuracy it has.

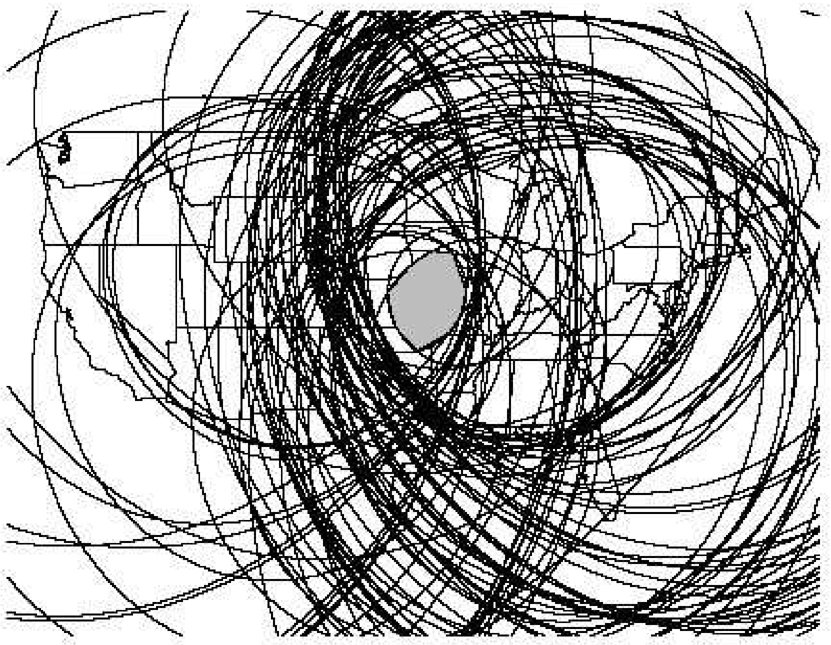

Figure 7: Intersection of Circles

5.2 Experimental Results

First of all, as *RandPing* always selects a relatively small subset of the probing machines that CBG would select, the running time of the former would always be much smaller than that of the latter, provided they are compared without implementation bias. Hereafer, we shall focus only on the accuracy perspective.

Figure 9 displays a sample of our experimental results. The first column of the table is the name of the location from where the target machine is. The second column is the real location of the target machine express by a (longitude, latitude) pair. The third column is the estimated location computed by *RandPing* and the fourth column is the error distance of the estimation. Additionally, Figure 10 sketches the cumulative probability as a function of error distance over all of our estimations. We did not include CBG in this graph, as it was based on a different set of landmarks.

From the two Figures, we can make two observations. First of all, most of the estimations have an error distance within 400km, which is acceptable. Moreover, if compared to the the performance Figure in CBG [2], our results are not as good, but close. In the next Section, we will analyze the possible reasons for the deficiency of our approach.

5.3 Experimental Analysis

In this Section, we will analyze our results and propose possible improvements. Basically, there are a few limitations in our experiments, in terms of both data sources and methodology.

- Due to the limitation of resources, the number of host machines available for us is much smaller than those for CBG and GeoPing. Therefore, the fact that *RandPing* does not deliver higher accuracy is not surprise.

Target	Real Location	Estimated Location	ED (km)
Cornell	(-76.47, 42.44)	(-72.37, 43.26)	345.9
Duke	(-78.94, 36.00)	(-73.97, 39.69)	633
HP	(-122.15, 37.44)	(-108.70, 38.22)	1147
Intel	(-122.31, 47.66)	(-122.20, 45.50)	250.1
Northwestern	(-87.69, 42.05)	(-89.94, 40.27)	272.2
Stanford	(-122.17, 37.42)	(-114.57, 35.89)	663
UNB	(-66.65, 45.96)	(-81.50, 40.57)	1338
Dartmouth	(-70.96, 41.61)	(-77.34, 40.93)	496.3
UCSC	(-122.06, 37.0)	(-119.10, 37.42)	270.2
UCSD	(-117.23, 32.87)	(-104.18, 36.52)	1246
UGA	(-83.36, 33.98)	(-76.74, 33.51)	591.1
UIUC	(-88.22, 40.11)	(-79.70, 39.50)	699.1
UMASS	(-72.52, 42.38)	(-68.57, 41.53)	333.7
UOregon	(-123.06, 44.04)	(-111.48, 39.17)	1075
Uvirginia	(-78.47, 38.06)	(-72.86, 39.94)	536.2
UW	(-122.31, 47.66)	(-106.18, 48.10)	1199
Winnipeg	(-97.16, 49.88)	(-98.25, 40.92)	1291
CalTech	(-118.15, 34.13)	(-114.43, 35.57)	373.3
Pittsburg	(-79.94, 40.44)	(-80.24, 39.76)	53.88
Rutgers	(-70.43, 40.52)	(-74.42, 40.54)	336.7
Umich	(-83.71, 42.29)	(-82.45, 42.61)	131.2
Wisc	(-89.38, 43.07)	(-88.20, 42.10)	150.6

Figure 9: Sample of Experimental Results

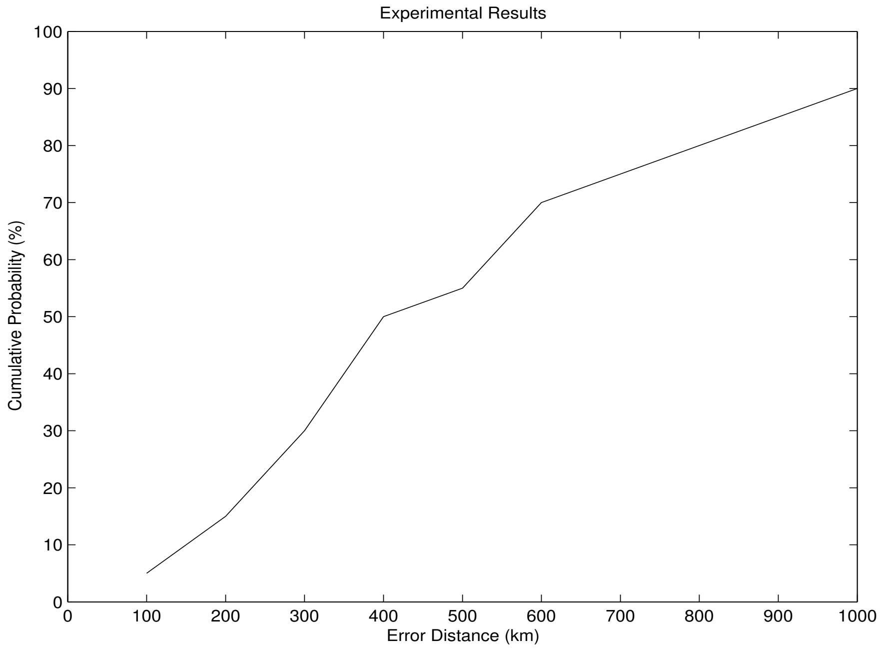

Figure 10: Cumulative Probabilities of Error Distances

- As there are not many probing machines, during the clustering process, we can divide them into only a small number of clusters, and thus, the effects of clustering and randomization are not obvious.
- The nature of our algorithm, that is, using only a small set of randomly selected machines, causes more inaccuracy. This is unavoidable, however, we hypothesize that there is no statistical difference between the accuracy of the deterministic approach and that of our randomized approach. We leave this as one direction of future work.
- Recall that, in the calculation of the best line for each landmark, we assume that the best line must be below each $(distance, RTT)$ data pair, that is, the RTT computed by the line must be smaller than the real RTT. In other words, the calculated distance will always be larger than the real distance. It turns out that this assumption is too conservative and induces even more inaccruacy when we try to find the locations of the target machine by the estimated distance and the location of the landmarks.

6. CONCLUSIONS

In this paper, we propose a randomized algorithm $RandPing$ for mapping IP addresses to physical locations. The framework is based on network delay measurements from randomly selected probing machines. Dynamic programming and non-linear programming techniques are employed to estimate the location of the target IP address. Our experimental results show that $RandPing$ has an acceptable accuracy but is not as good as CBG. In future work, our goals are threefolds. First, we seek to prove that the difference between the accuracy of the deterministic appraoch and that of the randomized one is upper bounded by a small counstant. Second, we plan to investigate better methods to estimate the best line bound for each probing machine in order to provide higher overall accuracy. Finally, another direction is to exploit the possiblity of improvements if we adjust the parameters of our randomized algorithm, such as the number of clusters (k), the number of trails (r), and the number of picked machines (l) in each cluster.

7. REFERENCES

[1] CAIDA, GTrace, http://www.caida.org/tools/visulization/gtrace/.

[2] Bamda Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based Geolocation of Internet Hosts. In Proc. of ACM Internet Measurement Conference 2004.

[3] Edwin M. Knorr and Raymond T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In Proceedings of the 24rd International Conference on Very Large Data Bases 1998, pp. 392–403.

[4] David Moore, ram Periakaruppan, Jim Donohoe, And Kimberly Claffy, Where in the world is netgeo.caida.org? In Proc. Of the INET'2000, Yokohama, Japan, July 2000.

[5] Raymond Ng and Jiawei Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. IEEE Transactions on Knowledge and Data Engineering 2002, pp. 1003–1016.

[6] Venkata N. Padmanabhan and Lakshminarayana Subramanian. An investigation of geographic mapping techniques for Internet hosts. In Proc. Of the ACM SIGCOMM'2001, San Diego, CA, USA, Aug. 2001.

[7] T. K. Sellis, N. Roussopoulos and C. Faloutsos. The

R^+-Tree: A Dynamic Index for Multi-dimensional Objects. *Proc. 1987 VLDB*, pp. 507–518.

[8] http://mathworld.wolfram.com/SteinerTree.html

[9] Anthony K. H. Tung, Jiawei Han, V. S. Lakshmanan and Raymond T. Ng. Constraint-Based Clustering in Large Databases. Lecture Notes in Computer Science, 2001.

[10] University of Illinois at Urbana-Champaign, IP Address to Latitude/Longitude, http://cello.cs.uiuc.edu/cgi-gin/slamm/ip2ll/

[11] Visulaware Inc., VisulaRoute, http://www.visualware.com/visularoute/.

[12] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In Proc. of ACM SIGMOD 1996, pp. 103–114.