Determining the Geographic Location of Internet Hosts

Venkata N. Padmanabhan
Microsoft Research

Lakshminarayanan Subramanian
University of California at Berkeley

November 2000

Technical Report
MSR-TR-2000-110

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Determining the Geographic Location of Internet Hosts

Venkata N. Padmanabhan[*]
Microsoft Research

Lakshminarayanan Subramanian[†]
University of California at Berkeley

Abstract

Knowing the physical location of users is a prerequisite for location-aware computing. Much work has gone into the user location problem in the context of wireless networks and mobile hosts. In this paper, we argue that location-aware computing is also relevant for stationary, wireline Internet hosts. We present and evaluate three distinct techniques — *GeoTrack*, *GeoPing*, and *GeoCluster* for determining the geographic location of Internet hosts.

GeoTrack infers location based on the DNS names of the host of interest or other nearby network nodes. GeoPing uses network delay measurements from geographically distributed locations to triangulate the coordinates of the host. GeoCluster couples partial host-to-location mapping information obtained (indirectly) from Web sites with BGP routing information to infer location of the host of interest. Using extensive and varied data sets, we evaluate the performance of these techniques. We also discuss the strengths and weaknesses of each technique.

1 Introduction

Location-aware computing [8, 7] is a powerful paradigm that captures the essence *user-centric* computing. If computing is to enable users to interact effectively with their environment, then computing should be a function of the user's location among other factors. This is the philosophy underlying location-aware computing. Both the behavior and the user-interface of applications are moulded by the user's location. A simple example is a printing service where the user's print jobs are routed based on which printer is located nearest the user's current location. Another is a restaurant location service that preferentially picks restaurants that are close to the user's location.

Knowing the physical location of the user is a prerequisite for location-aware computing. The granularity of location information needed may vary depending on the application. Much work has gone into the user location problem in the context of wireless networks and mobile hosts. Examples of such scenarios include a laptop user walking about in a building and a cell phone user driving around a city. A variety of approaches have been used for determining user/host location in a wireless setting: inference based on wireless signal timing [8] and strength [1], this mobile host's point of attachement in a cellular network, and of course the Global Positioning System (GPS) [4].

In this paper, we argue that location-aware computing is also relevant for the more traditional Internet hosts, such as users' desktop machines, that are mostly stationary and are typically connected via a fixed wireline network. Consider a user browsing information on a news Web site. There are many ways in which the information purveyed to the user can be moulded based on his or her physical location. The user could be sent information on *local* events, weather, etc. Advertisements can be targetted based on location. Access to content can be controlled based on location (*territorial rights management*) in a manner akin to TV broadcast rights that are often restricted to specific geographic areas.

It is possible for a Web site to determine a user's location by requiring the user to register with the site and then "log in" each time he or she visits the site. While such a mechanism may be appropriate for services with high security requirements (such as banking and email), it is probably too heavyweight for the vast majority of Web sites (such as news sites) that users browse casually. An alternative to requiring users to log in is to store location information (among other things) in a client-based *cookie* at the time of registration and then include the cookie in future requests. Such an approach does not require the user to log in on each visit, but it still imposes the burden of registration. Moreover, the cookie information may be unavailable when the user connects from a host other than the one he or she registered from. Finally, the location information manually input by an individual user may be inaccurate or errorneous, so depending on is undesirable.

Our model is one where Internet servers try to deduce the location of clients without depending on explicit information from the human user or the client's ISP. So the goal is to determine the geographic location of the user knowing only the IP address of the client he or she is connecting from. This is a challenging problem because the IP address does not inherently include any indication of geographic location. We have developed several novel techniques that approach this problem from different angles. These techniques exploit different properties of the

[*]http://www.research.microsoft.com/~padmanab/

[†]http://www.cs.berkeley.edu/~lakme/

Internet such as hierarchical addressing and correlation between delay and distance. We analyze a variety of data sets both to refine these techniques and evaluate their performance. To the best of our knowledge, ours is the first research effort in the open literature that studies this problem in detail.

The first technique, *GeoTrack*, tries to infer location based on the DNS names of the host of interest or other nearby network nodes. The DNS name of an Internet host sometimes contains clues about the host's location. Such a clue, when present, could indicate location at different levels of granularity such as city (e.g., *corerouter1.SanFrancisco.cw.net* indicates the city of San Francisco, USA), state/region (e.g., *www.state.ca.us* indicates the state of California, USA), or country (e.g., *www.un.cm* indicates the country of Cameroon). Even when present, the clue could be misleading (e.g., the host with the DNS name *www.newyork.com* is actually located in the city of New Orleans). So such clues need to be used with care.

The second technique, *GeoPing*, uses network delay measurements made from geographically distributed locations to triangulate the coordinates of the host. It is based on the premise that the delay experienced by packets travelling between a pair of hosts in the network is, to first order, a function of the geographic separation between the hosts (akin to the relationship between signal strength and distance exploited by certain user positioning systems in wireless networks). This is, of course, only an approximation. So our delay-based technique relies heavily on empirical measurements of network delay.

The third technique, *GeoCluster*, couples partial host-to-location mapping information obtained (indirectly) from Web sites with BGP routing information to infer the location of the host of interest. For our research, we obtained the host-to-location mapping information from a variety of sources, including a popular Web-based email site, a business Web hosting site, and a TV listing site. (User identity information was protected through anonymization.) The data thus obtained is *partial* in the sense that it only includes a relatively small number of IP addresses. We use BGP routing information to expand the coverage of this data by identifying clusters of IP addresses that are likely to be located in the same geographic area.

None of these techniques is perfect. Each has its strengths and weaknesses. Nevertheless the performance of our techniques is encouraging. The median error in our estimate of location varies from 28 km to several hundred kilometers depending on the technique used and the nature of the hosts being located (whether they are in a well-defined geographic clusters such as university campuses or whether they are dispersed such as AOL clients). The accuracy needed is a function of the application. We discuss how some of our techniques are self-calibrating in that they can offer an indication of how accurate a specific location estimate is likely to be.

The rest of this paper is organized as follows. In Section 2 we survey related work. In Section 3 we describe our experimental setup and methodology. We present the details of the three techniques we have developed and an analysis of their performance in Sections 4, 5, and 6. Finally, we present our conclusions in Section 7 and outline ongoing and future work in Section 8.

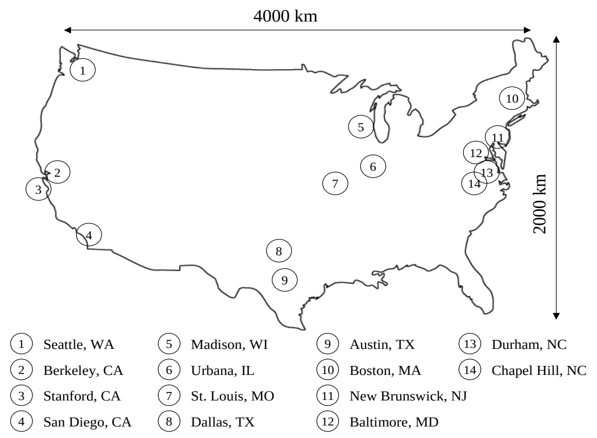

Figure 1: Outline map of the U.S. showing locations of our probe machines.

2 Related Work

There has been an increasing interest in location-aware computing and services in wireless environments (e.g., [7],[2]). As a result there has been been much work on the problem of locating hosts in such environments. The most well-known among these is the Global Positioning System (GPS) [4]. However, GPS is ineffective indoors. There have been several systems targeted specifically at indoor environments, including Active Badge [7], Active Bat [8], and RADAR [1]. As we discuss later, our GeoPing technique uses a variant of RADAR's NNSS algorithm. However, in general these techniques are specific to wireless networks and do not readily extend to the Internet.

There are many commercially and publicly available products that attempt to solve the host location problem in the Internet. These include VisualRoute [36], WhatRoute [37], GeoBoy [22], Neotrace [30], Traceware [33], NetGeo [12], IP2LL [34], GeoPoint [23] and Gtrace [13]. A list of existing approaches and their shortcomings is discussed in section 2.1.

Some of these tools are used by companies and other agencies for enabling different services on the Internet. Neotrace and VisualRoute have been used successfully to pinpoint causes of network problems, identify the intruder in the network, explore Internet geography and hunt down spammers. The FBI, NATO, US Customs and many ISPs are customers of these tools [30]. Digital Island announced a release of its tool called Traceware [33] to enable creation and deployment of applications that either require a geographic location service of clients or deliver geographically relevant customer experience [33]. Akamai's EdgeScape [16] offers the service of determining the location of networks and their corresponding type. Akamai [16] also has the capability of supporting customized content based on the geographic region of a host. Since many these commercial services use proprietary algorithms, it is hard to compare them with our research effort.

2.1 Existing Approaches and their Problems

There are multiple ways of building a IP address to geographic location mapping service. Many existing approaches and proposals for solving the problem can be broadly classified into the following categories:

1. Incorporating the latitude and longitude information in the DNS.
2. Using the *whois* [6] database to determine the location of the organization to which an IP address was allocated.
3. Performing traceroute to an IP address and mapping the router label to the geographic location using airport codes [15], city codes and country codes [20].

Uri Raz [31] presents a list of possible ways of solving the IP-geography problem. The first approach mentioned above was proposed in RFC 1876 [14]. This work defines the format of a new Resource Record(RR) for the Domain Name System(DNS) and reserves a corresponding DNS type mnemonic (LOC) and numerical code (29).The problems with the DNS based approach include:

1. A modification of the record structure of DNS records.
2. Requiring different administrations to enter the LOC records into the DNS record database.
3. No easy way of verifying whether the location entered by a user or administrator is correct and trustworthy.

A popular approach that has been adopted widely in many tools is querying *whois* servers. Tools such as IP2LL [34], and Allwhois [17] and NetGeo [12] use the *whois* database to determine the geographic location of a host. A list of publicly available whois servers is available at [26].

There are several problems with *whois* based approaches (many of these are listed in [12]). The main problems are:

1. The *whois* database is highly unreliable. The domain name maintainers do not insist on keeping the database accurate and current. Records corresponding to an IP address block may be present in multiple registries, but these records may not be consistent.
2. A large block of IP addresses may be allocated to a single entity. For any IP address in that block, the *whois* server will return only the headquarters or the address registered by the organization. For example, the 8.0.0.0/8 IP address block is allocated to BBN Planet and a query to the ARIN *whois* database will only return "Cambridge, MA" for any IP address within this range.
3. Due to web-hosting and domain name transfers, the location registered in the *whois* database may be very different from the actual location of host server. For example, a *whois* query on *www.desktop.com* would return the location as Colorado though the servers are based in San Francisco.

Several commercial products use *traceroute*[9] as the basic tool for tracing the geographic path to a given IP address. These solutions include VisualRoute [36], Neotrace [30], GeoBoy [22], WhatRoute [37] and Gtrace [13]. Our GeoTrack tool is motivated by many of these tools and attempts to remove some of their shortcomings. GTrace uses traceroute and ping measurements while Neotrace, VisualRoute and WhatRoute use traceroute, ping and *whois* queries for determining location. GeoBoy keeps track of the geographic location from a series of cache files which can be updated and customized by the user [22]. The details of how GeoBoy works is proprietary.

The basic idea in any traceroute-based tool is to perform a traceroute from a source to a given IP address and look at the *router labels* (i.e., the DNS names associated with a router's network interfaces) along the path. The router labels may have the geographic location information hidden in terms of city codes, airport codes and country codes. Although traceroute-based tools avoid many of the shortcomings of the *whois*-based approaches, they still face quite a few problems. These include:

1. Many Web clients are behind proxies which drop traceroute packets. So geographic locations are normally traceable only to the proxy's location.
2. Each ISP has its own naming scheme for cities and it is difficult to capture the different naming schemes. For example, every ISP has its own naming scheme for San Francisco. Some of the commonly used names include *sfo, sffca, sanfrancisco, sanfranciscosfd, snfr, snfrca.*
3. There are a lot of cities that have the same name or same city code. Most of these codes do not give

out any information regarding the state or the country. In such cases, it will be difficult to determine which city is being referred to in a router label. For example, many ISPs use the word "Bloomington" in their router labels to refer to the location Bloomington. Given that there are 21 different locations called Bloomington in the US, it is hard to determine the actual Bloomington where a particular router may be present.

3 Experimental Methodology

As outlined in Section 1, our techniques for determining location are varied in the information they draw upon and their mode of operation. In this section, we discuss the experimental setting and data sets we used to develop these techniques and analyze their performance.

3.1 Geographic Setting

All of our experiments are set in the United States (U.S.). The main reason for this restriction is that, as of the time of this writing, the bulk of the data sets and probe machines we have access to pertain to or are located in the U.S. While there may be limitations to studying a single country, the U.S. still offers a large and varied testbed for our research. The U.S. consists of 50 states, 48 of which are located in the large geographic area depicted in Figure 1, and two others that are located 2000 km to the northwest and 4000 km to the southwest, respectively, of this landmass. (In addition, our data sets recorded the U.S. capital, Washington DC, as a separate entity, so we effectively had 51 "states".) Thus, the U.S. is as large as certain continents in terms of geographic expanse. It is also home to a sizeable fraction of the Internet, in terms of networks, routers, end hosts, and users. So we believe the research reported in this paper is interesting despite being limited to the U.S. In Section 8, we discuss ongoing work that extends beyond the U.S.

3.2 GeoPing and GeoTrack

GeoPing requires network delay measurements to be made to the host of interest from several known locations. For this purpose, we obtained access to probe machines at the 14 locations depicted in Figure 1. These machines were typically well-connected hosts on university campuses and were geographically distributed across the U.S. In some instances, we deliberately picked probe points closely clustered together (e.g., Berkeley and Stanford) with a view to comparing the corresponding delay data to validate (or invalidate!) the basic premise of GeoPing that geographic distance is a significant determinant of delay.

As we explain later in Section 5.1.1, GeoPing is primed using a database of delay measurements from the probe machines to several "target" machines at known locations. To obtain such a database, we constructed a list of 265 Web servers (termed *UnivHosts*) spread across university campuses in 44 states of the U.S. using information from [24, 25]. The selection of university servers as target hosts offered the advantage that we were very certain of their location.

The UnivHosts data set is also used to evaluate the performance of GeoTrack and GeoCluster.

3.3 GeoCluster

GeoCluster draws upon a variety of datasets. BGP routing information is derived from dumps taken at two routers at BBNPlanet [18] and MERIT [28]. We were only interested in the *address prefix (AP)* information, so we constructed a superset containing address prefix information derived from both sources. In all there were 100,666 APs in our list.

We obtained IP-to-location mapping information from three sources (for the sake of anonymity, we do not reveal their actual names here). Note that the data sets we obtained were *partial* in the sense that they only covered a small fraction of IP address space in use.

1. *FooMail*: a popular Web-based email service with several million active users. Of the over 1 million (anonymous) users we obtained information for, we focussed on the 417721 users who had registered their location as being in the U.S. The location information we obtained from the users' registration records was at the granularity of U.S. states. In addition, we obtained a log of the client IP addresses corresponding to the 10 most recent user logins (primarily in the first half of 2000). We combined the login and registration information to obtain a partial IP-to-location mapping.

2. *FooHost*: a business Web hosting site. Location information (at the granularity of zip codes) was recorded at the time of user registration. This information was included in a cookie when a client sent a request to the server. In all we obtained location information corresponding to 181246 unique IP addresses seen during (part of) a day in October 2000.

3. *FooTV*: an online TV program guide where people check on program listings for a specific zip code. From traces gathered over a two-day period in February 2000, we obtained a list of 142807 unique client IP addresses and 336181 pairs of (IP, zip) corresponding to the client IP address and the zip code that the user specified in his/her query. A subset of the IP addresses had more than one corresponding zip code (usually clustered together geographically).

In the case of FooHost and FooTV, we mapped the zip code information to the corresponding (approximate) latitude and longitude using information from the U.S. Census Bureau [35]. Location information in FooMail is at the granularity of states. We computed the *zipcenter* of each state by averaging the coordinates of the zip codes contained within that state.

We do not expect the partial IP-to-location mapping obtained from these sources to be entirely accurate. For instance, in the case of FooMail and FooHost users may connect from locations other then the one they registered. In the case of FooTV, users may enquire about TV programs in areas far removed from their current location, but we believe this is unlikely. Regardless, in Section 6 we explain how the GeoCluster technique addresses this issue.

4 GeoTrack: A Traceroute Based Tool

This section gives a description of the *GeoTrack* tool that we have developed for tracing the geographic location of an IP address. The related work section describes several IP-to-geography mapping tools in use and discusses their shortcomings. The design and implementation of our tool has been motivated by many of these publicly available tools such as VisualRoute, Gtrace, and Neotrace [13, 30]. We now describe how GeoTrack works and how it is different from the existing tools.

4.1 GeoTrack

GeoTrack is a tool for tracing the geographic location of an IP address. It uses traceroute and ping measurements from a given source to the specified IP address. The router labels are converted into the geographic location using city codes, airport codes and country codes. Our tool can give a reasonable estimate of the location in the U.S., Canada, and 26 different countries in Europe. The tool can also decipher the router's location to a country's granularity based on the country codes for other countries.

The characteristic features of GeoTrack that is different from many of the existing traceroute based tools are the following:

- *No Whois:* Given the limitations of *whois* listed in Section 2.1, we decided not to use *whois*-based lookups in our tool.

- *Airport Codes:* The total number of airport codes is very large. In the U.S. alone, the number of airport codes is 2722. Often an airport code might inadvertently appear in a router label. For example, *MIT* refers to an airport in Shafter, CA, but also appears in all router labels in the *mit.edu* domain located in Cambridge, MA. In our GeoTrack tool we have pruned the airport code database to a much smaller data set. We determined from our traceroute logs that many ISPs outside the US do not use airport codes for naming routers. Therefore we restricted our list of airport codes to just the US.

- *Country and Continent Specific Networks:* Country codes may not always be present in router labels. In order to find the location of a router to a country or a continent's granularity, we generated a list of networks which are restricted to specific countries and continents. We have generated over 200 country and region specific networks in the U.S., Canada and Europe. These do not include any host networks.

- *City Code Database:* As in VisualRoute and Neotrace, we generated a list of city codes for a number of cities in the U.S., Canada, and 26 countries in Europe. To infer the different codes associated with a city, we perform traceroutes to several web sites in the city from multiple traceroute servers [32].

- *Partitioning of City Codes:* To minimize the chances of an inadvertent match between city codes and common substrings present in router labels, we partition the city codes based on the country and the continent. We use the information about the network to decide which partition of codes should be used to infer the location of a router.

- *Delay Based Correction:* As in the tool Gtrace [13], we use a delay based correction mechanisms to remove incorrect guesses of geographic location. If the difference in minimum delays to two adjacent routers is very small(less than 5 ms) then they are presumed to be very close to each other geographically.

GeoTrack determines the location of the routers based on router labels and returns the location of the router closest to the destination which is recognizable as the location of the end-host.

4.2 Performance Characteristics

This section details the results of the performance of our GeoTrack tool. We consider two test samples of IP addresses to determine the location using our tool. We perform our experiment from 14 different source points as explained in the previous section. The location of these 14 sources is shown in Figure 1.

4.2.1 Results

We performed two different experiments to test our GeoTrack tool. In the first experiment, we ran the GeoTrack tool from the 14 probe locations (Figure 1) using UnivHosts as the target dataset. For every data point, we define the *error distance* to be the geographic distance between the actual location of the data point and the location as determined by the GeoTrack tool. The distance between two geographic locations is computed as the shortest path between the two locations (*great circle route*). We compute the cumulative probability distribution function of the error distance for each of the 14 sources. The results are shown in Figure 2.

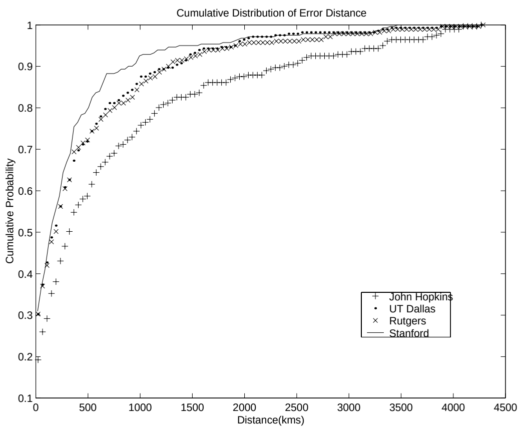

Figure 2: Cumulative Distribution of Error Distance for 4 different probe locations

Figure 2 displays the cumulative distributions of four of the probe locations located at JHU, Stanford, Rutgers and Dallas. The performance of Stanford is much better than the other three probe points. Among the 14 probe points, JHU reported the worst error distance characteristics. For many of the universities, JHU recognized the location only to the granularity of the closest well connected city in some backbone thereby increasing the error distance.

In our second experiment, we compute the error characteristics for a Client IP address dataset that we construct from the FooTV website. The Client IP dataset consists of a random sample of 2380 IP addresses of clients who accessed FooTV. We use GeoTrack to determine the location of the clients from 3 different sources. The 3 different sources in this experiment are located at Stanford,CA(West Coast), St.Louis,MO(Center) and Chapel Hill,NC(East Coast). In this experiment, we define the error distance of an IP address to be the distance between the location determined by the tool and the zip code location entered by the user. An important point to note in this experiment is that an IP address may be associated with multiple locations, suggesting that the IP address is allocated dynamically (say using DHCP [3] for dialup clients) or it is assigned to a proxy host (such as a Web proxy or a firewall). So multiple clients in different location may use the same IP address at different times. We compute the performance of the Client IP dataset to the UnivHosts dataset for the 3 different probe locations.

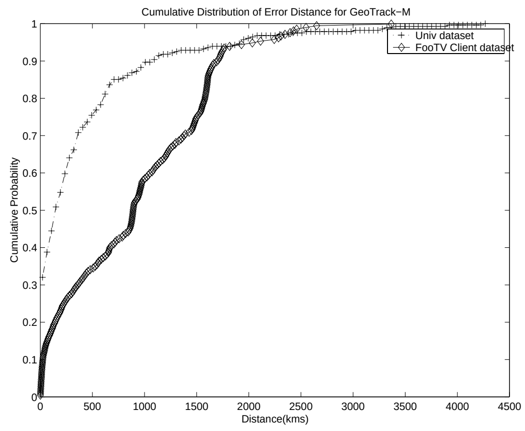

Figure 3: Comparison of the CDF of error distance for the UnivHosts dataset and the Client IP dataset for UNC

Figure 3 clearly indicates that the performance of the Univ dataset is much better than the Client IP dataset. Many of the clients are behind proxies, so the “client” addresses seen by Web servers may actually be addresses of proxies. GeoTrack can at best determine the location of the proxy. The error distance between the proxy and the end-host may be quite large and this can directly affect our results. The client dataset also reports multiple locations for a single IP address indicating either that the address corresponds to a proxy or that the same IP address is assigned (at different times) to two clients at different locations. These considerations impose fundamental limits on mapping the location of a client using a traceroute-based technique. The trends look similar for both Stanford and St.Louis. The fact that 30% of the clients have an error less than 200 kms, even for the FooTV dataset, is encouraging.

4.3 Discussion

Any traceroute-based approach has some fundamental limitations. Many commercial products have tried to get around these limitations by making use of databases such as *whois*. However, as we noted earlier, the *whois* database is not a good tool for locating clients. For example, VisualRoute reports all AOL clients with IP addresses in the block 205.188.128.0/17 as being located at Sterling, VA (near AOL’s headquarters). This anomaly arises from using *whois* records for determining location.

In our tool we tried to get eliminate this problem by performing traceroutes from multiple locations. There are still a lot of clients whose IP routes go through a common proxy no matter where you traceroute from. In such cases, one can determine only the location of the proxy and not the actual location of the client.

We have developed a variant of GeoTrack called GeoTrack-M which improves the average error distance of GeoTrack. GeoTrack-M runs GeoTrack from multiple sources and determines the last recognizable router from multiple sources. GeoTrack-M reports the majority of the locations reported from different sources. Experimental results indicate that GeoTrack-M performs much better than 7 of the 14 probe locations and is slightly inferior in performance to the best probe point(St. Louis).

5 GeoPing: A Delay Based Location Tracker

GeoPing is a tool that we have developed to determine the geographic location of an IP address by exploiting the relationship between network delay and geographic distance. The GeoPing tool measures the delay to a host from multiple sources at known locations and attempts to correlate the delay measurements to triangulate the coordinates of the host. GeoPing uses GeoTrack as a tool for determining the location of intermediate routers and determining a model that captures the relationship between network delay and geographic distance.

We have devised two very different statistical methods for developing a delay-based host location tool. We have named our two methodologies as NNDS and PDF-based. NNDS stands for *Nearest Neighbors in Delay Space* and borrows ideas from the NNSS approach detailed in [1]. The PDF-based approach computes PDFs (probability density function) for each source (i.e., probe location) that establish a relationship between the network delay to the geographic distance from that source. Using these PDFs, it attempts to find the the distance estimates from each sources and a composite location estimate. Before we describe the two methods, we present some measurements that motivate using delay measurements for determining the geographic location of an Internet host.

5.1 Correlation between Network Delay and Geographic Distance

This section presents some measurements that indicate the presence of a relationship between network delay and geographic distance. Using these measurements we would like to answer two specific questions.

1. Given the network delay of a host from a source, is there a model that can predict the distance of the host from that source?

2. If there is a relationship between network delay and geographic distance as measured from a given source, then is that relationship, a property of that source alone or is it a property of the geographic location of the source?

5.1.1 Experimental Setting

We use the UnivHosts dataset for performing our measurements. We perform traceroutes and ping measurements from the 14 sources in Figure 1 to all the 265 university servers in UnivHosts. We traceroute and find the path from a given source to the host and determine the (round-trip) delay to all intermediate routers using ping measurements. From multiple delay samples, we compute the minimum delay to each intermediate router in the path and the minimum delay to the destination. We use GeoTrack to determine the physical locations of intermediate routers and costruct a large dataset of (minimum delay, geographic distance) pairs for all intermediate routers in the path. We divide the delay range into bins of width 10 ms and determine the probability distribution (PDF) of the geographic distance for each delay bin. This probability distribution of distance for specific delay bins is used to study the correlation between network delay and geographic distance.

5.1.2 Results

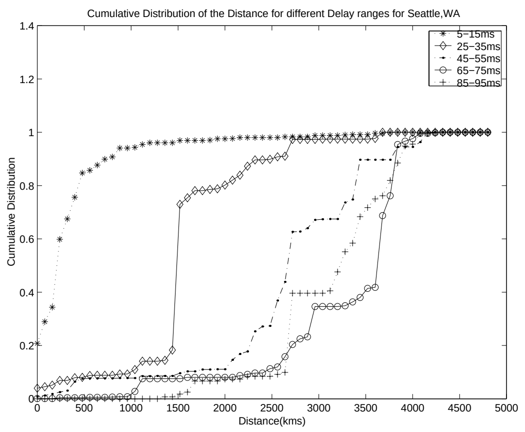

Figure 4: Cumulative Distribution of geographic distance for multiple delay ranges based on data gathered at the Seattle, WA probe location.

Figure 4 presents the cumulative distribution of the geographic distance from Seattle,WA as a function of the delay. We have split the delay measurements into non-overlapping delay bins of 10 ms width and we compute the CDF for every delay bin. Figure 4 presents the CDF for 5 delay bins. It is evident from the graph that for each delay bin, there are distinct peaks in the probability distribution for specific distance ranges. For example, the distinct peak around 1300 kms for the 25-35 ms delay bin is mainly contributed by locations in the Bay Area(Palo Alto, San Francisco). The other noticeable trend is that as the delay increases from 0 to 80 ms, the peaks of the probability distributions shift to the right until we reach the east-west coast distance limit of the US.

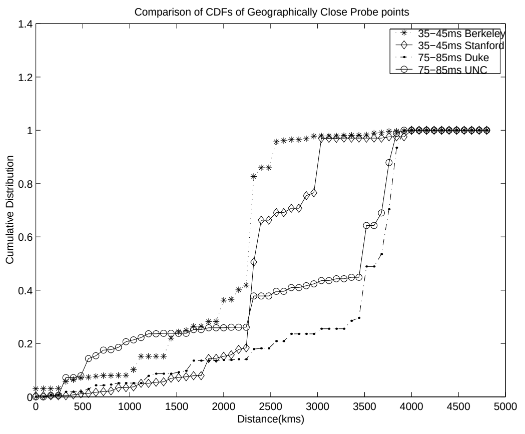

Figure 5: Comparison of CDFs for adjacent geographic locations

Figure 5 compares the CDFs of adjacent geographic locations. We compare the 35-45 ms delay bins for Stanford and Berkeley and the 75-85 ms delay bins of Duke and UNC. Berkeley and Stanford are 50 miles apart, and Duke and UNC are 8 miles apart. Figure 5 shows that the PDFs of adjacent locations are definitely correlated. The peaks for Berkeley and Stanford occur at the same distance range of 2200-2300 kms for the 35-45 ms delay range. The PDFs of Duke and UNC have peaks around 3500-3700 kms for the 75-85 ms delay range. We examined the other delay ranges for these two sets of locations and there is a reasonable match in the peaks of the corresponding PDFs. This indeed shows that the peaks associated with CDFs for specific delay ranges are a function of the geographic location of the source (i.e., probe location) and not the specific source alone.

5.2 PDF-based Delay Triangulation

In order to exploit the relationships between network delay and geographic distance, we propose a PDF-based delay triangulation approach. The basic idea used in PDF-based methods is to compute an error function over the entire location space and find the coordinates where the error function is minimized. Every location is associated with an error measure which gives an indication of how far it is from the actual location of the host. In our approach, we minimize the error function across a list of known cities, which are geographically distributed, and report the city with the minimum value of the error function as the location of the host.

In this section, we explore several *error functions* based on the probability distribution of geographic distance as a function of the network delay from each source. We evaluate the performance of four different error function based approaches using the UnivHosts dataset. Before explaining these approaches, we briefly describe our terminology. S denotes the location of a probe source, H refers to the host whose location needs to be determined, CT refers to a collection of cities within the US, L denotes an element drawn from the city list CT, $d(S,L)$ is the distance between source S and the location L and $rtt(S,H)$ is the round trip time of the host H from source S.

5.2.1 Weighted Least Mean Squares(LMSQ)

In weighted least mean squares, we compute the error function at a particular location L for a given source, S, as follows:

1. Given $rtt(S,H)$, we find the top $K(=3$ by default) peaks of the probability distribution of the distance for a small delay range around $rtt(S,H)$. Note that all our delay measurements are RTT measurements and not one way delay.

2. For each peak P among the K peaks, we compute the error to be $err(P,S,L) = (d(P) - d(S,L))^2/prob(P)$, where $d(P)$ and $prob(P)$ refer to the distance value of the peak P and its probability density in the PDF.

3. Among the K error values, we compute the minimum value of error to be the error measure of the location with respect to source S. Therefore $error(S,L) = min_P(err(P,S,L))$.

4. The error measure with respect to all sources for a particular location L is the sum total of all error from different sources. Mathematically, $error(L) = \sum_S error(S,L)$.

The LMSQ algorithm reports the location L with the least value of $error(L)$. The reasoning behind choosing the top K peaks is to take into consideration the presence of multiple disjoint peaks in the probability distribution function and the $prob(P)$ term was used to weigh the error inversely to the probability distribution around the peak.

5.2.2 Probability Density Estimation(PROB)

In the Probability based method, we compute a joint probability distribution for every location based on the delay information and maximize the joint probability distribution. The steps in the computation of the joint probability for a location L are:

1. We assume a distance window of $\delta(= 50 miles)$ around a given distance value.

2. Given the distance $d(S,L)$ for a location L and source S we define the probability $p(S,L)$ to be the area under the probability density curve within a distance window of δ around $d(S,L)$.

3. We compute the joint probability to be the product of the individual probabilities. Therefore $p(L) = \pi_S(p(S,L))$.

We compute the location L with the maximum value of $p(L)$ and report it as the location of the host H.

5.2.3 Composite Statistical Approaches

In the previous sections, we described two statistical mechanisms LMSQ and PROB which had completely different flavors for removing the noise in the system. Is it possible to combine different statistical approaches to improve the accuracy in the location estimation? We propose two composite statistical approaches LMPR and PRLM which combine the two approaches to determine the location of the host. The basic idea of composite schedulers is to use one statistical method to prune the solution space and the other for optimizing in the pruned space. In LMPR, the LMSQ method is used to prune the solution space and PROB is used to optimize on the pruned space. In PRLM, PROB is used as the pruning tool and LMSQ as the optimization tool.

The steps used in LMSQ are:

1. Compute $min_{LMSQ} = minimum_L(error_{LMSQ}(L))$.
2. Let X be the set of all locations L such that $error_{LMSQ}(L) \leq THRESHOLD \times min_{LMSQ}$.
3. Compute the location L in X with the maximum value of $p(L)$ as defined in the PROB method.

In a similar fashion, one can define the steps of the PRLM algorithm.

5.2.4 Experimental Results

We used traceroute and ping measurements to multiple locations to determine the probability distribution function for each source. To test the four PDF-based statistical approaches (LMSQ, PROB, LMPR, and PRLM) we used the UnivHosts dataset. We measure the delay to the endhost from each of the 14 sources and apply the 4 statistical approaches to determine the location of the host. We compute the error distance for each of the hosts and plot the CDFs of error distance for LMSQ, LMPR, PROB and PRLM.

Our results, as shown in Figure 6, indicate that PROB is a better statistical method than LMSQ. The performance of PROB and PRLM are very similar and these two techniques perform much better than LMPR or LMSQ. Least Mean Squares(LMSQ) is a widely used statistical tool and in this particular problem, LMSQ does not seem to work well. PROB performs much better and reports a location within 500 kms for 50% of the locations. The $75th$ percentile occurs around 1100 kms for the PROB method. By using LMSQ as the pruning method and PROB as the optimization tool, we improve the performance over LMSQ. However, using PROB as a

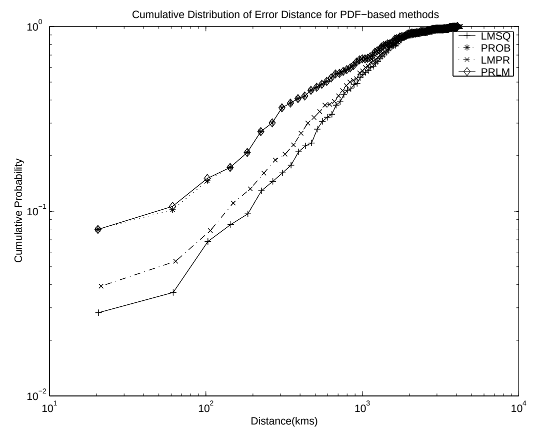

Figure 6: Comparison of CDFs for LMSQ, PROB, LMPR and PRLM

pruning tool improves the performance significantly over LMSQ.(Note that the axis of the graphs are in log scale)

5.3 Nearest Neighbors in Delay Space(NNDS)

We ask the question: Is there is a nice way of nullifying the noise in the measurements to improve the accuracy in determining the location of a host? We explore an entirely different approach in which we cluster points based on their locality in the delay space. We consider a list of known hosts and their locations and determine their delays from multiple known sources. We characterize the geographic location of a host using an N-dimensional delay vector, where each coordinate is the minimum network delay to that host from a particular source. Mathematically, every data point X can be characterized by a delay vector $D(X) = (d_1, \ldots, d_N)$ where d_i is the minimum delay of X from source i and this delay vector can be labeled with location L, the geographic location of X. Given the delay vector of $D(U)$ of an unknown host U we determine the nearest neighbor Y in the $N-$dimensional delay space (using the Eucledian measure of "distance" in delay space) and report the location of Y as the location of U. This approach is similar to the Nearest Neighbor in Signal Space (NNSS) algorithm in RADAR [1] which operated on vectors of signal strength rather than delay.

So why does this method reduce the noise in the system? From every source, we measure the minimum delay to every host as the representative delay sample. If a particular source reports highly noisy data for a few hosts then many hosts in that geographic location may experience similar noise levels. Since NNDS computes the nearest neighbors in the delay space, the noise of neighboring nodes nullify each other out in the Euclidean distance metric. This error neutralization however does not happen for the LMSQ method. This noise is however partially detected by the PROB method, since the presence of many hosts with similar noise levels in a particular geographic location (within a distance window of δ) increases the probability $p(S, L)$ for that particular source S and location L.

5.3.1 Experimental Results

We performed two different experiments to study the error distance characteristics. We use the UnivHosts dataset as the sample data set and the test data set in our experiments. In other words, for determining the location of one university server we use the data set of all other universities (excluding the one being located) as the sample dataset.

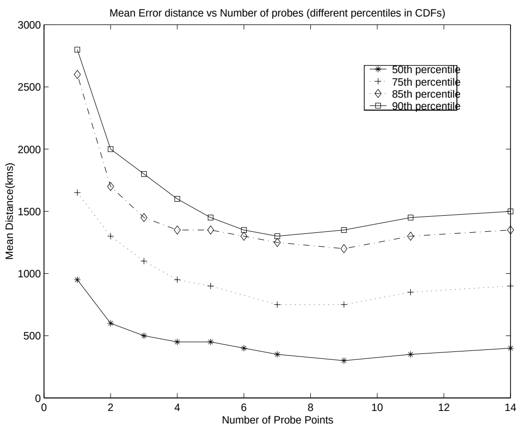

Figure 7: Mean Error Distance vs Number of Probe Points (characteristics at different percentiles in CDF plots)

Figure 7 shows several percentile levels of the mean error distance as a function of the number of probe points. For example, the 75th percentile curve corresponds to the distance at which the CDF plot of error distance crosses the 0.75 probability mark. For each percentile level and number of probes (say n), we compute the mean error distance to be the average of error distances corresponding to several geographically distributed placements of n probes locations (chosen from the set of 14 possible locations). For example, for 2 probes, we average the error distance over different placements of 2 probes in geographically dispersed locations among the 14 possible locations. From Figure 7, we infer that the error distance initially decreases sharply as the number of probes increases, then stabilizes and reaches an optimal value between 7 and 9 probe locations, and finally increases slightly for higher values. This suggests that having 7 to 9 probes is optimal for the NNDS algorithm.

5.4 Discussion

It is interesting to note that there is indeed significant correlation between network delay and geographic distance. This relationship is pronounced for small values of delay and the predictability of distance, as a function of delay, decreases as delay increases. We have presented several ways of exploiting the correlation between the two to determine the location of an unknown host. However, it is worth noting that GeoPing does not perform as well as GeoTrack. The best error distribution of GeoPing(9 probes NNDS) performs slightly better than the worst-case error distribution of GeoTrack(JHU) for the UnivHosts data set.

We have investigated various probe placement strategies for the NNDS approach. We studied the effects of probe placement on the error distribution. Our investigations indicate that a geographically well distributed set of probes give better error charecteristics than clustered probe placements. However, the placement of probes has a smaller impact on performance than the number of probes.

In our evaluation of GeoPing we have only considered hosts in the UnivHosts data set. These are typically well-connected hosts on university campuses. The correlation between delay and distance may break down when we consider hosts with a "last-mile" link with a large delay (for example, a dialup link or a satellite link). In the case of dialup link, we could use GeoPing to determine the location of the last router along the path to the target host. This location may serve as a good approximation for the location of the target host since users tend to dial in to modem banks in their local area.

6 The GeoCluster Technique

The GeoCluster technique learns about IP-to-location mapping from data sources such as the ones mentioned in Section 3.3. However, as noted before such information tends to be partial in coverage (since it includes location information for only a relatively small subset of the IP address space) and not entirely accurate. These problems limit the utility of the IP-to-location mapping data.

GeoCluster seeks to address both problems by clustering together IP addresses corresponding to hosts in the same geographic location[1], i.e., addresses that form a *geographic cluster*. Clustering enables us to expand the coverage of the partial IP-to-location mapping information. As a simple example, suppose we know that 128.127.126.0/8[2] forms a geographic cluster. Furthermore assume that the partial mapping information tells us that the location corresponding to 10 different IP addresses in this cluster is Foo City. Then we can reasonably deduce that the remaining 246 IP addresses in this cluster (if they are in-

[1]The granularity of the location depends on the application context.

[2]The notation $a.b.c.d/m$ denotes an address slice with a prefix of length m bits specified.

deed in use) are also likely to correspond to hosts in (or near) Foo City.

Identifying geographic clusters is a challenging problem. Our GeoCluster technique approaches this problem in a novel way by combining partial IP-to-location mapping information with network routing information. We build on the work presented in [10] on identifying *topological* clusters. Address allocation and routing in the Internet is hierarchical. Routing information is aggregated across hosts that are under a single administrative domain (also known as an *autonomous system (AS)*). For example, the routes for hosts on a university campus would typically be advertised to the rest of the network as a single aggregate, say the address prefix 128.127.0.0/16, rather than 65536 individual IP addresses. Thus knowledge of the address prefixes (APs) used by the routing protocol enables us to identify *topological* clusters as observed in [10]). We surmise that APs also constitute *geographic* clusters. We elaborate on this below.

We derive information on APs from the *border gateway protocol (BGP)* used for inter-domain (i.e., inter-AS) routing in the Internet. Each entry in the BGP table at a router specifies a destination AP and the AS-level path leading to it. For our purposes, we are only interested in the AP information, so we construct a list of unique APs (over 100000 APs, as mentioned in Section 3.3). The number of APs is an order of magnitude larger than the number of ASs. This is because an AS, such as an ISP, may advertise more specific routes (say for certain customers) due to policy and/or performance considerations (e.g., for load balancing).

An AS (and its associated AP(s)) often corresponds to a geographical cluster such as a university campus or a company office. Even when the AS is an ISP with large geographic coverage, the associated APs that are advertised via BGP may be more specific (say corresponding to individual customers), as explained above. In both these cases, GeoCluster is in a good position able to identify geographic clusters from AP information. However, large ISPs (e.g., AT&T, Sprint, UUNet, etc.) often advertise only aggregate APs for reasons of scalability. In such cases, a single AP may span a large geographical area. This problem would be alleviated if we had AP information not only from inter-domain BGP (also known as *External BGP (E-BGP)*) but also from intra-domain intra-domain BGP (*Internal BGP (I-BGP)*). I-BGP information would provide a window into how an AP is subdivided within a large ISP. However, I-BGP information is not made available to ISPs, so this was not a feasible option for us.

In summary, our baseline GeoCluster algorithm discovers APs only from E-BGP data and treats these APs as geographic clusters. We term this variant of GeoCluster *BGPonly*. As explained above, BGPonly has shortcomings, for instance when ISPs only advertise large aggregates. We now present a sub-clustering algorithm designed to address this problem. We term this variant of GeoCluster *BGP+subclustering*.

6.1 Sub-clustering Algorithm

The BGP+subclustering variant of GeoCluster depends only on inter-domain BGP (i.e., E-BGP) data just like BGPonly. But the novel idea is to use the (partial) IP-to-location mapping information available to subdivide APs that have a large geographic spread. For each original AP obtained from E-BGP, we use the IP-to-location mapping information to determine whether their is "significant" consensus on the geographic location of the AP. If there is, then we declare the AP to be a geographic cluster. If not, we subdivide the AP into two halves (e.g., the AP 152.153.0.0/16 would be subdivided into 152.153.0.0/17 and 152.153.128.0/17) and repeat the test on each half. We stop when the subdivision contains too few IP-to-location mapping data points for a reliable determination of geographic clustering to be made. In the end, we obtain a mapping from APs (both original and subdivided ones) to location. Given an IP address, we first find the matching AP using longest prefix match and then report the corresponding location as the location of the IP address.

Here is pseudocode for this sub-clustering algorithm. Let `IPLoclist` be the list of IP-to-location mapping data points sorted by IP address, `BGPAPlist` be the list of APs obtained from E-BGP information, `IPLocAPlist` be the sorted list obtained by augmenting the entries in `IPLoclist` with the APs corresponding to the longest prefix match, `newAPLoclist` be the new list mapping APs to location obtained by (possibly) subdividing the original APs, and `cthresh` be the minimum threshold on the number of IP-to-location mapping data points within a subdivision.

```
  /* initialization */
IPLoclist = sorted IP-to-location mapping
BGPAPlist = APs derived from E-BGP info
/* determine matching APs */
foreach ((IP,location) in IPLoclist) {
  AP = LongestPrefixMatch(IP,BGPAPlist)
  Add (IP,location,AP) to IPLocAPlist
}
/* subdivide APs using IPLocAPlist */
sameAPlist = EMPTY
curAP = AP in first entry of IPLocAPlist
foreach ((IP,location,AP) in IPLocAPlist) {
  if (AP in (IP,location,AP) == curAP) {
    /* contiguous list with same AP */
    Add (IP,location,AP) to sameAPlist
  } else {
    /* Subdivide curAP as appropriate */
    if (|sameAPlist|≥ cthresh) {
      if (sameAPlist is geographically
clustered) {
        avgLocation = average location of
cluster
        Add (curAP,avgLocation) to newAPLoclist
      } else {
        Divide curAP into two equal halves
        Divide sameAPlist accordingly
        Recursively test whether either/both of
subdivisions form a geographic cluster
      }
    }
    /* reset/reinitialize sameAPlist */
    sameAPlist = NULL
    Add (IP,location,AP) to sameAPlist
  }
}
newAPLoclist is the new list used for
IP-to-location mapping
```

The effectiveness of this algorithm depends on the richness of the (partial) IP-to-location mapping data available. If insufficient data is available for certain APs, these will not be included in `newAPLoclist`. So GeoCluster will be unable to determine the location of IP addresses that match those APs.

Considering aggregates of IP-to-location data points also offers the advantage that isolated errors in the individual data points can be neutralized. For instance, suppose that 90% of the IP-to-location data points corresponding to an AP agree on the location being California while the remaining 10% are scattered elsewhere. We can make the reasonable assumption that the entire AP is located in California and attribute the discrepancies to errors in the IP-to-location mapping data.

We have not specified how it is determined whether a set of locations is clusterered geographically or how the "average" location of a cluster is computed. The answers to both of these questions are context-dependent — dependent on the granularity of the location information contained in the partial IP-to-location mapping and on the needs of the application. In case the location information is relatively fine-grained (e.g., zip codes), the location of the individual points is quantifiable (say using latitude and longitude). So we could compute an *composite* location using averaging. We could also compute a dispersion metric that quantifies the spread. In case location information is coarse-grained (e.g., states), we could test to see if more than a certain fraction of the points agree on location. We describe the specific approach we use in our experimental analysis in Section 6.3.

6.2 Impact of Proxies and Firewalls

Many Internet clients lie behind proxies and/or firewalls that separate the corporate or ISP network from the rest of the Internet. In such a setting, often the proxy or firewall connects to external Internet hosts (such as Web servers) on behalf of the client hosts. The IP address of the client hosts remains hidden from the external network. As such there is no direct way to map from IP address to location for such clients. (After all we are interested in the location of the client, not that of the proxy or the firewall.)

Our sub-clustering algorithm deals with this issue elegantly. If the set of clients that connect via a group of proxies (having IP addresses that are contained within an address prefix AP) is clustered geographically (say at location L), then given a sufficient number of IP-to-location data points, the sub-clustering algorithm will (correctly) determine an association between the address prefix AP and the location L. This is what happens say in the case of clients on a university or coporate campus or clients of an ISP that connect via a local (or regional) proxy. However, there are instances (such as with the ISP America Online (AOL)) where clients in geographically diverse locations share a common pool of proxies. (With AOL we have seen clients thousands of kilometers apart connect via a proxy with the same IP address!) In such a case, our sub-clustering algorithm will not be able to determine any geographic clusters, so it will not try to map the IP address to a client location. We believe this is an important property of the sub-clustering algorithm because for some applications providing (highly) inaccurate location information may be much worse than not providing any information at all. For instance, displaying a generic advertisement on a New York user's screen would probably be better than mistakenly displaying an advertisement specific to California.

6.3 Experimental Results

We now analyze the performance of GeoCluster in several ways using a variety of data sets. We compare the performance of GeoCluster with that of GeoTrack and GeoPing. We analyze two variants of GeoCluster: (1) only using AP information derived from BGP tables (*BGPonly*), and (2) post-processing the BGP tables using the sub-clustering algorithm discussed in Section 6.1 (*BGP+subclustering*). We compare both variants against a simplistic approach that ignores BGP infomation and assumes that all APs to have a 24-bit prefix length (*/24-clusters*).

6.3.1 Locating hosts in UnivHosts

We first analyze the ability of GeoCluster (the BGPonly variant) in determing the location of the 265 hosts in the UnivHosts list (Section 3.2). We use IP-to-location mapping information contained in the FooTV data set. As explained in Sectionsubsec-geocluster-data, we convert each zip code contained in the FooTV data to the corresponding (approximate) latitude and longitude. Then using AP information derived from BGP data, we cluster the (IP,latitude,longitude) data points based on the AP that the IP address lies in. For each cluster we compute a *composite* location (as a (latitude,longitude) pair) by linearly averaging the latitudes and longitudes of the constituent points. Thus we finally obtain a mapping between APs and the corresponding (latitude,longitude) pairs. (Only APs that have at least data point indicating location are included in this mapping.) Given an IP address, we find the matching AP (using longest prefix match) and output the corresponding (latitude,longitude) pair as the location.

We quantify the accuracy of the location estimated by GeoCluster (and the other techniques) using the *error distance*.

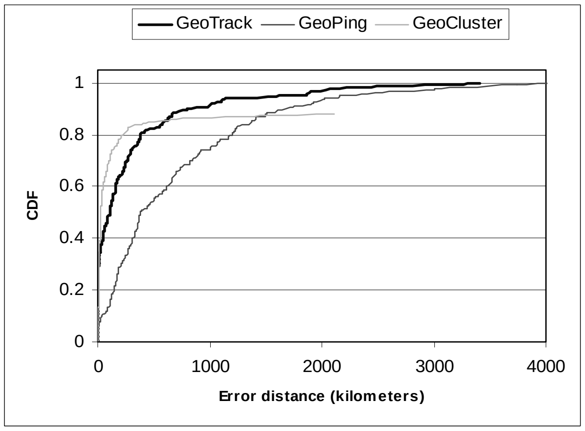

Figure 8: CDF of the error distance computed over the UnivHosts data set for GeoTrack, GeoPing, and GeoCluster.

Figure 8 shows the CDF of error distance for GeoCluster computed over the 265 university hosts. We also show the best case CDFs of GeoTrack (initiated from the St. Louis probe machine) and GeoPing (using 9 probe machines) for comparison. GeoCluster is able to deduce the location of only 233 out of the 265 (i.e., about 88%) university hosts. This is because the IP-to-location mapping data derived from FooTV is *partial* in coverage, and despite the clustering performed using BGP data, we still have no location information for about 12% of the hosts. However, for the vast majority of hosts for which it is able to determine location, GeoCluster significantly outperforms (the best cases of) both GeoTrack and GeoPing. For instance, the median and 80th percentile marks for GeoCluster are only 28 km and 226 km, respectively. The corresponding numbers are 108 km and 384 km for GeoTrack, and 382 km and 1201 km for GeoPing.

GeoCluster performs well on the UnivHosts data set because these hosts are often clustered together geographically on university campuses. Moreover, many (but not all) universities have distinct address allocations (e.g., 150.131.0.0/16 for the University of Montana) that are advertised via BGP as distinct address prefixes (APs). So GeoCluster is able to identify the universities as geographic clusters with relative ease.

6.3.2 Locating hosts in FooHost

We now analyze the performance of GeoCluster using a much larger test data set, namely the FooHost data set. This data set contains 181246 unique IP addresses and their corresponding zip codes. (As noted in Section 3.3, the zip code information may not be entirely accurate. Hence, unlike with the universities data set, we are not entirely certain of the true locations of the hosts.) As in our analysis using the university hosts, we use the BGPonly variant of GeoCluster with the FooTV and the BGP data sets used to prime the GeoCluster algorithm.

For each IP address in FooHost, we estimate its location and then compute the error distance. The error distance, with the IP addresses sorted in increasing order of error distance, is shown in Figure 9. We observe that GeoCluster is only able to estimate location for about 77% of the 181246 hosts. The 25th, 50th (median), and 75th percentile marks of the error distance are 84 km, 685 km, and 3056 km respectively. In other words, GeoCluster performs much worse for the FooHost data set than for the UnivHosts data set.

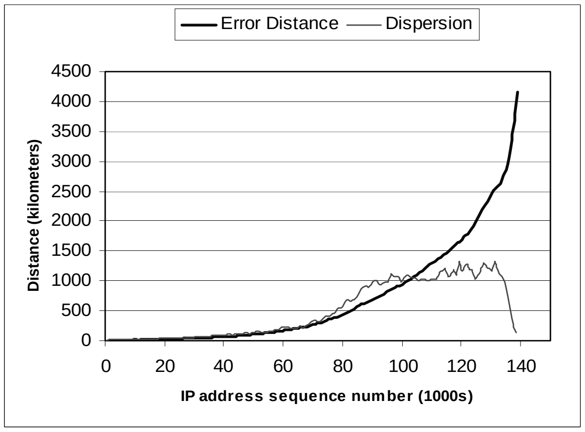

Figure 9: The error distance and the dispersion for hosts in FooHost.

The main reason for the worse performance is that the FooHost data set is much more diverse than the UnivHosts data set. Unlike UnivHosts, many of the IP addresses in FooHost fall within the APs corresponding to large and geographically-dispersed ISPs (e.g., 12.0.0.0/8 belonging to AT&T WorldNet) or belong to proxies or firewalls (e.g., AOL clients). Hence GeoCluster is only able to determine location accurately for a smaller fraction of the hosts.

Given the wide range of error distance for different hosts, it would be useful to be able to tell when GeoCluster's estimate is accurate and when it is not. For this purpose, we compute a metric, which we call *dispersion*, for each AP as follows. As oulined in Section 6.3.1, for each AP we consider all IP addresses from the training set (FooTV in the present instance) that are contained within that AP. We compute a composite location for the AP as a whole using linear averaging. We then compute the dispersion metric for this AP as the mean distance of the training points from the composite location of the AP. Intuitively, dispersion quantifies the extent of geographic spread in the AP.

We would expect that the larger the dispersion is, the less accurate GeoCluster's estimate of location would be. This is borne out by Figure 9 which depicts the (smoothed version of) dispersion curve for the FooHost data set. In fact, the dispersion curve matches the error distance curve quite well (except for hosts at the extreme right). This makes intuitive sense since the error in location estimation results from the geographic spread of APs, and it is exactly this spread that dispersion quantifies.

At the extreme right of the graph, we see that error distance shoots up while the dispersion drops sharply. To better understand this puzzling phenomenon, we took a closer look at the corresponding (IP,zip) data points in FooHost. Based on this examination, we have come to the conclusion that the apparent discrepancy is caused by errors in the FooHost data set. As discussed earlier, the location information in FooHost is gathered from user registration information contained in cookies. An IP address recorded in this log may not always correspond to a host in the same location as registered by the user. For example, FooHost contains the IP address 140.247.147.42 (which maps to the DNS name room147-42.student.harvard.edu), which presumably corresponds to a host at Harvard University in the northeastern corner of the U.S. However, the corresponding location conatined in the FooHost data is Portland, Oregon, 4000 km away in the northwestern corner of the U.S.! Thus we believe the sharp spike in error distance is misleading; the error distance may in fact be very small.

The errors in our test data notwithstanding, we believe that GeoCluster would not perform as well for a diverse set of hosts as for the university hosts. Still the error distance is relatively small (within a couple of hundred kilometers) for a substantial fraction (around 40%) of the hosts. And, quite importantly, GeoCluster is self-calibrating in the sense that it is able to tell when a location estimate is likely to be accurate and when it is not.

6.3.3 Importance of the sub-clustering algorithm

Thus far we have considered the BGPonly variant of GeoCluster that only uses AP information derived directly from BGP data. Now we now turn to the BGP+subclustering variant that employs the sub-clustering algorithm (Section 6.1) to construct an AP-to-location mapping. This algorithm makes use of both BGP data and partial IP-to-location mapping information. We are interested in studying what benefit, if any, the sub-clustering algorithm provides.

We use the partial IP-to-location mapping contained in FooMail (Section 3.3) as input to the sub-clustering algorithm. Recall that the location information in FooMail is at the granularity of states. This raises the question (discussed in Section 6.1) of deciding when a set of locations corresponding to an AP is deemed to be "geographically clustered". We use the following test: an AP is deemed to correspond to a geographic cluster if it contains at least *cthresh* (IP,location) data points, at least $fthresh$ of which (as a fraction) share the same location (i.e., the same state). In most of the results shown here, we set *cthresh* $= 20$ and $fthresh = 0.7$ and denote this as $(20, 0.7)$. We also briefly discuss results for the (5,0.6) setting.

We use FooHost as the test data. Recall that the location information in FooHost is at the granularity of zip codes whereas that in FooMail is at the granularity of states. This raises the question of how to quantify accuracy. We decided to do all of our calculation at the granularity of the states. We map the zip codes in FooHost to the corresponding states. We then compute the *zip-center* of each state by averaging the coordinates of the zip codes contained within that state (Section 3.3). The error distance is then computed as the distance between the zipcenters of the actual and deduced states. So the error distance is zero if the state is deduced correctly and non-zero otherwise.

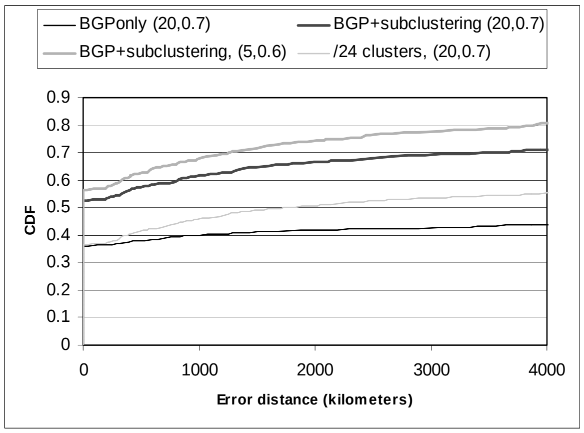

Figure 10: CDF of the error distance (computed at the granularity of states) for the BGPonly and BGP+subclustering variants of GeoCluster, and for the /24-clusters method.

Figure 10 shows the CDF of error distance. We observe that BGP+subclustering significantly outperforms BGPonly. In particular, with the $(20, 0.7)$ setting BGP+subclustering gets the state right (i.e., an error distance of zero) for 53% of the hosts while BGPonly does so for only 36% of the hosts. The reason is that BGPonly is often stuck with large, geographically-dispersed APs obtained directly from BGP data while the sub-clustering algorithm is often able to break these down into smaller and more (geographically) compact APs. It is interesting to note that even /24-clusters, which completely ignores BGP data, outperforms BGPonly slightly, although it is still much worse than BGP+subclustering.

Finally, we see that BGP+subclustering performs slighly better with the $(5, 0.6)$ setting compared to $(20, 0.7)$ (the correct state is deduced for 56% of the hosts compared to 53%). Nevertheless we believe that a $(5, 0.6)$ setting may be too aggresive in the sense that it may often misidentify geographic clusters (after all $(5, 0.6)$ requires just 3 out of 5 data points to agree on location for an AP to be deemed a geographic cluster). We are presently investigating this issue further.

6.4 Discussion

In summary, GeoCluster employs a novel algorithm that combines partial IP-to-location mapping information with BGP routing information to make an intelligent determination of a client's location. The algorithm is able to tolerate a limited amount of inaccuracy in the IP-to-location mapping information and remain effective in certain situations where clients connect via proxies or firewalls.

An interesting question is how one would obtain partial IP-to-location mapping information in general. There are several possible ways one might do this.

1. The *likely* location of a user can be inferred from the kind of information accessed or queries issued by the user (as, for example, in the case of FooTV). Since it only considers such information an aggregated form (corresponding to clusters), GeoCluster is able to tolerate a limited amount of inaccuracy in the inference.

2. Certain Web sites, such as Yahoo [38] and MSN [29], offer a mix of generic content (e.g., news) and user-specific content (e.g., email). Partial IP-to-location mapping information may be derived from accesses made by (registered) users to the latter content and then used in conjunction with GeoCluster to infer the location of (the often much larger number of) users (registered and unregistered) who access generic content.

In general, we expect that there will be a relatively small number of content providers and "location servers" (akin to advertisement servers such as DoubleClick [21]) that will employ GeoCluster (and the other techniques) to map IP addresses to geographic locations. The vast majority of Web sites would simply subscribe to the services provided by the location servers and so need not be concerned with the details of the location mapping techniques.

7 Conclusions

In this paper we have examined the interesting but challenging problem of determing the geographic location of Internet hosts. Our motivating example is an Internet server that determines the locations of its clients and moulds its behavior accordingly. In general the server only knows the IP address of the client host, so the challenge is to map this address to a geographic location.

We have designed and evaluated three distinct techniques to solve this problem: (a) *GeoTrack*, which extracts location information from DNS names of hosts and routers, (b) *GeoPing*, which determines location using network delay measurements made from several known locations, and (c) *GeoCluster*, which combines partial IP-to-location mapping information with BGP routing data to determine location.

While our location determination techniques are nowhere near wireless systems such as GPS in terms of accuracy, we believe that our experimental results are encouraging. Our techniques work best for well-connected and geographically clustered hosts, such as those on a university campus. The median error distance in such cases is as low as 28 km, roughly the size of a city. For a more heterogeneous mix of hosts, the median error distance is around 600 km, about the size of a large state in the U.S. Furthermore, the GeoCluster technique gives an indication of how accurate its location estimate is likely to be.

Our techniques are able to handle to an reasonable extent the difficulties posed by proxies and firewalls in certain situations (e.g., clients on a geographically compact corporate campus or clients that connect via local or regional proxies). However, we are unable to determine the location of client hosts behind centralized proxies such as those employed by large ISPs such as AOL.

We believe our techniques enable an interesting class of location-aware services for Internet hosts. Targeted advertising is one example which can be quite effective even if used for only 30-40% of clients. Such advertising can be done at the level of a city (e.g., sporting events), state (e.g., election advertisements), or region (e.g., weather alerts).

Besides the specific techniques that we have developed, we believe an important contribution of our paper is that it presents a systematic study of the IP-to-location mapping problem and analyzes a wide range of interesting data sets.

8 Ongoing and Future Work

We are presently working on extending our techniques to countries outside the U.S. We have already extended GeoTrack to Canada and 26 countries in Europe. We are presently enlisting probe locations in Europe to enable us to extend GeoPing as well.

We are investigating combinations of the three techniques to improve accuracy. Specifically, we working on a composition of GeoTrack and GeoPing that uses the network delay from the last recognizable router to the destination host to estimate the location of the latter.

In the future, we are plan to integrate our host location techniques with a Web server to provide simple location-aware services.

Acknowledgements

George Chung, Alex Daunys, Cheng Ku, and Dave Quick helped us obtain the FooMail, FooHost, and FooTV datasets. Craig Labovitz helped us gather BGP data and answered our numerous BGP-related questions. Donald Wysocki provided us with geographic mapping information for U.S. cities and zip codes. Arvind Arasu, B. R. Badrinath, Mary Baker, Paul Barford, John Byers, Imrich Chlamtac, Mike Dahlin, Kevin Jaffey, Randy Katz, Craig Labovitz, Paul Leyland, Karthik Mahesh, Vijay Parthasarathy, Jerry Prince, Amin Vahdat, Srinivasan Venkatachary, Geoff Voelker, Marcel Waldvogel, and David Wood helped us obtain access to the geographically-distributed probe machines. We would like to thank them all.

References

[1] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. *IEEE INFOCOM 2000*, Tel-Aviv, Israel.

[2] K. Cheverst, N. Davies, K. Mitchell and A. Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE project. *The Sixth Annual International Conference on Mobile Computing and Networking(Mobicom2000)* , Boston, August 2000.

[3] R. Droms, Dynamic Host Configuration Protocol, *RFC-1531, IETF*, October 1993.

[4] P. Enge and P. Misra, Special Issue on GPS: The Global Positioning System, *Proc. of the IEEE*, January 1999.

[5] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. *IEEE Infocom 2000*, Tel Aviv, Israel.

[6] K. Harrenstien, M. Stahl, E. Feinler, NICKNAME/WHOIS, *RFC-954, IETF*, October 1985.

[7] Andy Harter, Andy Hopper. A Distributed Location System, for the Active Office. *IEEE Network* Vol.8 No.1, Jan 1994.

[8] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, The Anatomy of a Context-Aware Application, *Proc. ACM/IEEE Mobicom*, August 1999.

[9] V. Jacobson, Traceroute software, 1989, *ftp://ftp.ee.lbl.gov/traceroute.tar.Z*

[10] B. Krishnamurthy, J. Wang. On Network Aware Clsutering of Web Clients. *ACM SIGCOMM 2000*, Stockholm, 2000.

[11] J. Y. Li et.al. A Scalable Location Service for Geographic Ad-Hoc Routing. *ACM Mobicom 2000*, Boston, 2000.

[12] D. Moore et.al. Where in the World is netgeo.caida.org? *INET 2000*, June 2000, Japan.

[13] R. Periakaruppan, E. Nemeth. GTrace- A Graphical Traceroute Tool. *Usenix LISA'99*, Nov 1999.

[14] D. C.Vixie, P.Goodwin and T.Dickinson. A Means for Expressing Location Information in the Domain Name System, *RFC-1876, IETF*, January 1996.

[15] Listing of Airport Codes. *http://www.mapping.com/airportcodes.html*

[16] Akamai Inc. *http://www.akamai.com/*

[17] AllWhois. *http://www.allwhois.com/*

[18] BBNPlanet publically available route server, *telnet://ner-routes.bbnplanet.net.*

[19] CAIDA. *http://www.caida.org/*

[20] Country Codes. *http://www.voicenet.com/ deptag/comp/co-codes.htm*

[21] DoubleClick, *http://www.doubleclick.com*

[22] GeoBoy, NDG Software *http://ndgsoftware.com/ ndgprod/page11.html*

[23] GeoPoint Service, Quova Inc. *http://www.quova.com/service.htm*

[24] List of Universities. *http://sunsite.berkeley.edu/Libweb*

[25] List of Top 100 Universities. *http://www.usnews.com/usnews/edu/college/corank.htm*

[26] List of Whois Servers. *http://consult.stanford.edu/pub/mail/whois-servers*

[27] Mapnet tool of CAIDA. *http://www.caida.org/tools/visualization/mapnet/Backbones*

[28] MERIT Network, *http://www.merit.edu*

[29] Microsoft Network (MSN), *http://www.msn.com*

[30] NeoTrace, A Graphical Traceroute Tool *http://www.neoworx.com/products/neotrace/default.asp*

[31] Raz,Uri. Finding a host's geographic location. *http://www.private.org.il/Ip2geo.html*

[32] Traceroute Servers. *http://www.traceroute.org/*

[33] Traceware, Digital Island, *http://www.digisle.net/services/app _serv/traceware.shtml*

[34] UIUC's IP to Latitude/Longitude Server, *http://cello.cs.uiuc.edu/cgi-bin/slamm/ip2ll*

[35] U.S. Gazetteer, U.S. Census Bureau, *http://www.census.gov/cgi-bin/gazetteer.*

[36] VisualRoute, Datametrics System Corporation, *http://www.visualroute.com*

[37] WhatRoute, Bryan Christianson, *http://homepages.ihug.co.nz/~bryanc*

[38] Yahoo, *http://www.yahoo.com*