An Investigation of Geographic Mapping Techniques for Internet Hosts

Venkata N. Padmanabhan*
Microsoft Research

Lakshminarayanan Subramanian†
University of California at Berkeley

ABSTRACT

In this paper, we ask whether it is possible to build an IP address to geographic location mapping service for Internet hosts. Such a service would enable a large and interesting class of location-aware applications. This is a challenging problem because an IP address does not inherently contain an indication of location.

We present and evaluate three distinct techniques, collectively referred to as *IP2Geo*, for determining the geographic location of Internet hosts. The first technique, *GeoTrack*, infers location based on the DNS names of the target host or other nearby network nodes. The second technique, *GeoPing*, uses network delay measurements from geographically distributed locations to deduce the coordinates of the target host. The third technique, *GeoCluster*, combines partial (and possibly inaccurate) host-to-location mapping information and BGP prefix information to infer the location of the target host. Using extensive and varied data sets, we evaluate the performance of these techniques and identify fundamental challenges in deducing geographic location from the IP address of an Internet host.

1. INTRODUCTION

In this paper, we ask the question: is it possible to build an IP address to geographic location mapping service for Internet hosts? Given an IP address, the mapping service would return the geographic location of the host to which the IP address has been assigned. This is a challenging problem because an IP address does not inherently contain an indication of geographic location.

Building an IP address to location mapping service (the *location mapping* problem for short) is an interesting problem in its own right. Such a service would also enable a large and interesting class of location-aware applications for Internet hosts, just as systems such as GPS [6] have for mobile devices. By knowing the location of a client host, an application, such as a Web service, could send the user location-based targeted information on local events, regional weather, etc. (*targeted advertising*), classify users based on location (e.g., count "hits" based on the region the user is located in), or control the availability of data based on user location (*territorial rights management* akin to TV broadcast rights). Each application may have a different requirement on the resolution of location information needed.

In this paper, we present several novel techniques, collectively referred to as *IP2Geo*, that approach the location mapping problem from different angles. These techniques exploit various properties of and observations on the Internet such as hierarchical addressing and correlation between delay and distance. We have analyzed 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 target host 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), state (e.g., *www.state.ca.us* indicates the state of California), or country (e.g., *www.un.cm* indicates the country of Cameroon).

The second technique, *GeoPing*, uses network delay measurements made from geographically distributed locations to infer the coordinates of the target host. It is based on the premise that the delay experienced by packets traveling 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 wireless user positioning systems such as RADAR[1]). This is, of course, only an approximation. So our delay-based technique relies heavily on empirical measurements of network delay, as discussed in Section 5.

The third technique, *GeoCluster*, combines partial (and possibly inaccurate) IP-to-location mapping information with BGP prefix 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 an online TV guide site. The data thus obtained is *partial* in the sense that it only includes a relatively small

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

†http://www.cs.berkeley.edu/~lakme/. The author was an intern at Microsoft Research through much of this work.

SIGCOMM'01, August 27-31, 2001, San Diego, California, USA..

number of IP addresses. We use BGP prefix 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. This technique is self-calibrating in that it can offer an indication of how accurate a specific location estimate is likely to be.

We have evaluated these techniques using extensive and varied data sets. While none of the techniques is perfect, their performance is encouraging. The median error in our location estimate varies from 28 km to several hundred kilometers depending on the technique used and the nature of the hosts being located (e.g., well-connected clients versus proxy clients). We believe that a significant contribution of our work is a systematic study of a broad spectrum of techniques and a discussion of the fundamental challenges in determining location based just on the IP address of a host.

The rest of this paper is organized as follows. In Section 2 we survey related work. In Section 3 we describe our design rationale and experimental methodology. We present the details of the three IP2Geo techniques and an analysis of their performance in Sections 4, 5, and 6. Finally, we present a summary and discuss the contributions of our work in Section 7.

2. RELATED WORK

There has been been much work on the problem of locating hosts in wireless environments. The most well-known among these is the Global Positioning System (GPS) [6]. However, GPS is ineffective indoors. There have been several systems targeted specifically at indoor environments, including Active Badge [9], Bat [10], and RADAR [1]. As we discuss later, our GeoPing technique uses a variant of one of the algorithms we had developed for RADAR. However, in general these techniques are specific to wireless networks and do not readily extend to the Internet.

In the Internet context, an approach that has been used to determine location is to seek the user's input (e.g., by requiring the user to register with and/or log in to the site, by storing the user's credentials in client-based cookies, etc.). However, such approaches are likely to be (a) burdensome on the user, (b) ineffective if the user uses a client other that the one where the cookie is stored, and (c) prone to errors due to (possibly deliberate) inaccuracies in the location information provided by an *individual* user. (In Section 6, we discuss how GeoCluster deals with such inaccuracies by aggregating information derived from individual users.)

An alternative approach is to build a service that maps an IP address to the corresponding geographic location [16]. There are several ways of doing this:

1. Incorporating location information (e.g., latitude and longitude) in Domain Name System (DNS) records.

2. Using the *Whois* [8] database to determine the location of the organization to which an IP address was assigned.

3. Using the *traceroute* [11] tool and mapping the router names in the path to geographic locations.

4. Doing an exhaustive tabulation IP address ranges and their corresponding locations.

The DNS-based approach was proposed in RFC 1876 [17]. This work defines the format of a new Resource Record (RR) for the DNS, and reserves a corresponding DNS type mnemonic (LOC) and numerical code (29). The DNS-based approach faces deployment hurdles since it requires a modification of the record structure of the DNS records. This also burdens administrators with the task of entering the LOC records. Moreover, there is no easy way of verifying the accuracy of the location entered.

An approach used widely in many tools is to query Whois servers [8]. Tools such as IP2LL [26] and NetGeo [14] use the location information recorded in the Whois database to infer the geographic location of a host.

There are several problems with Whois-based approaches. First, the information recorded in the Whois database may be inaccurate or stale. Also, there may be inconsistencies between multiple servers that contain records corresponding to an IP address block. Second, a large (and geographically dispersed) block of IP addresses may be allocated to a single entity and the Whois database may contain just a single entry for the entire block. For example, the 4.0.0.0/8 IP address block is allocated to BBN Planet (now known as Genuity) and a query to ARIN Whois database returns the location as Cambridge, MA for any IP address within this range.

An alternative approach is based on the traceroute tool. The basic idea here is to perform a traceroute from a source to the target IP address and infer location information from the DNS names of routers along the path. A router name may not always contain location information. Even when it does, it is often challenging to identify the location information since there is no standard naming convention that is used by all ISPs. We discuss these issues in more detail when we present GeoTrack in Section 4. Examples of location mapping tools based on traceroute include VisualRoute [31], Neotrace [29], and GTrace [15].

Finally, there are location mapping services, such as EdgeScape from Akamai [18] and TraceWare from Digital Island [22]. Given the extensive relationship that these large content distribution networks enjoy with several ISPs, it is conceivable that these location mapping services are based on an exhaustive tabulation of IP address ranges and the corresponding location. However, the algorithms employed by EdgeScape and TraceWare are proprietary, so it is difficult for us to compare them to our research effort.

2.1 Fundamental Limitation due to Proxies

Many Web clients are behind proxies or firewalls. So the "client" IP address seen by the external network may actually correspond to a proxy, which may be problematic for location mapping. In some cases the client and the proxy may be in close proximity (e.g., a caching proxy on a university campus). However, in other cases they may be far apart. An example of the latter is the AOL network [19], which has a centralized cluster of proxies at one location (Virginia) for serving client hosts located all across the U.S. Figure 1 shows the cumulative distribution function (CDF) of the distance between the AOL proxies and clients. (The *likely* location of clients was inferred from the data sets described in Section 3.5.) We observe that a significant fraction of the clients are located several hundred to a few thousand kilometers from the proxies.

Proxies impose a fundamental limitation on all location

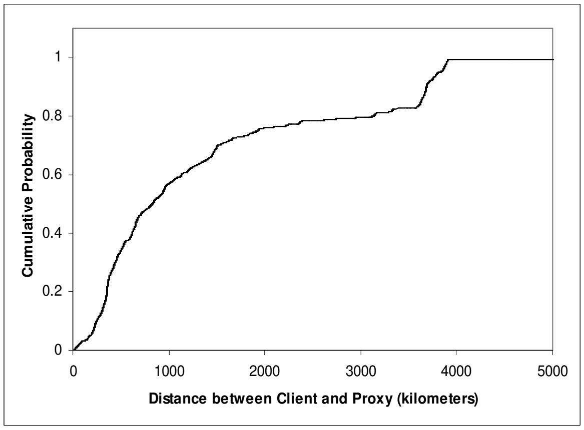

Figure 1: Distribution of distance between AOL proxies and clients.

mapping techniques that depend on client IP address. This includes techniques based on Whois, traceroute (e.g., GeoTrack), and network delay measurements (e.g., GeoPing). Not only are these schemes unable to determine the true location of a client, they are also oblivious to the error (i.e., these schemes would incorrectly return the location of the proxy without realizing the error). Our GeoCluster technique is an exception in that it is often able to automatically tell when its location estimate is likely to be erroneous. So rather than incorrectly deducing the location of the client based on the IP address of the proxy, GeoCluster would refrain from making a location estimate at all. We discuss this issue is more detail in Section 6.3.

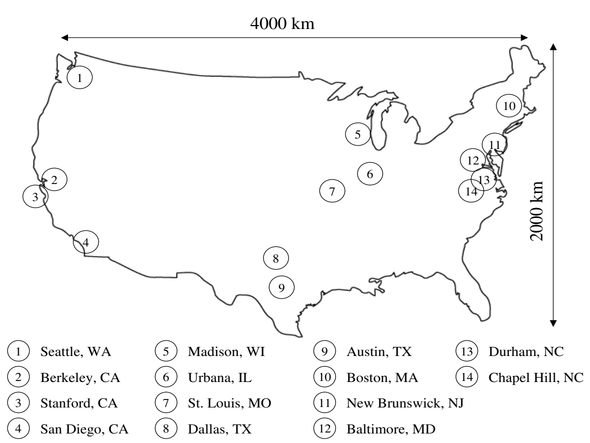

Figure 2: Outline map of the U.S. showing locations of our probe machines.

3. DESIGN RATIONALE AND EXPERIMENTAL METHODOLOGY

In this section, we first discuss the design rationale for IP2Geo in view of the limitations of existing techniques discussed above. We then describe the experimental setup and data sets used in our study.

3.1 Design Rationale

In Section 2, we discussed several existing approaches to location mapping and their limitations. Among these, Whois and traceroute based approaches are the most popular. To get a better understanding of the strengths and limitations of these two approaches, we have developed GeoTrack, a traceroute-based tool for performing location mapping that is largely based on the same principles as existing traceroute-based tools such as VisualRoute and GTrace. We compare the performance of GeoTrack with NetGeo, a Whois-based location mapping tool.

IP2Geo also includes two new techniques, GeoPing and GeoCluster, which operate very differently from existing approaches. GeoPing exploits the correlation between network delay and geographic distance to determine the location of a host. Although this correlation is not strong enough to be captured in a mathematical model, we show that is indeed possible to be build a coarse-grained location tracker using just delay measurements.

We also describe GeoCluster, a powerful new technique that combines partial IP-to-location mapping information obtained from a variety of sources and topological clustering data [12] to do location mapping. Our results indicate that GeoCluster performs the best among the IP2Geo techniques.

Before getting to the details of these techniques, we describe the experimental setup and data sets that we have used in our study.

3.2 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 that we have 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 2, 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.

3.3 Probe Machines

We obtained access to probe machines at the 14 locations depicted in Figure 2. These machines were distributed geographically across the U.S. All of them were well-connected hosts on university campuses except for the machine at Seattle, WA, which was located at a corporate site (Microsoft). These probe machines were used to make delay measurements for GeoPing and to initiate traceroutes for GeoTrack.

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. The selection of university servers as target hosts offered the advantage that we were quite certain of their actual geographic location.

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

3.4 BGP Data

BGP routing information was derived from dumps taken at two routers at BBN Planet [20] and MERIT [28]. Since GeoCluster only requires the *address prefix (AP)* information, we constructed a superset containing address prefix information derived from both sources. In all there were 100,666 APs in our list.

3.5 Partial Location Mapping Information

We obtained partial IP-to-location mapping information from three sources. The data sets we obtained were partial in the sense that they only covered a small fraction of IP address space in use. Note that in no case did we have access to user IDs or other user-specific information. Our data sets only contained IP address and location information. So our work did not compromise user privacy in any way.

1. *Hotmail*: Hotmail [24] is a popular Web-based email service with several million active users. Of the over 1 million (anonymous) users we obtained information for, we focused 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. *bCentral*: bCentral [21] is a business Web hosting site. Location information at the granularity of zip codes was derived from HTTP cookies. In all we obtained location information corresponding to 181246 unique IP addresses seen during (part of) a day in October 2000.

3. *FooTV*: FooTV is an online TV program guide where people look up program listings for specific zip codes. (We do not reveal the name of the site here due to anonymity requirements.) From traces gathered over a two-day period in February 2000, we obtained a list of 142807 unique client IP addresses and 336181 (IP,zip) pairs 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, which were usually clustered together geographically.

In the case of bCentral and FooTV, we mapped the zip code information to the corresponding (approximate) latitude and longitude using information from the U.S. Census Bureau [30]. In the case of Hotmail, we computed the *zip-center* of each state by averaging the coordinates of the zip codes contained within that state.

The partial IP-to-location mapping obtained from these sources may contain inaccuracies. For instance, in the case of Hotmail and bCentral users may have registered incorrect location information or may connect from locations other than the one they registered. In the case of FooTV, users may enquire about TV programs in areas far removed from their current location, although we believe this is unlikely. Regardless, we explain in Section 6 how GeoCluster is robust to such inaccuracies in location information.

4. THE GEOTRACK TECHNIQUE

The GeoTrack technique tries to infer location based on the DNS names of the host of interest or other nearby network nodes. Network operators often assign geographically meaningful names to routers[1], presumably for administrative convenience. For example, the name *corerouter1.SanFrancisco.cw.net* corresponds to a router located in San Francisco. We stress that having geographically meaningful router names is *not* a requirement or a fundamental property of the Internet. Rather it simply an observation that is generally supported by empirical data.

We define a router to be *recognizable* if its geographic location can be inferred from its DNS name. Routers whose IP address cannot be mapped to a DNS name or whose DNS name does not contain meaningful location information are considered as not being recognizable.

GeoTrack uses these geographic hints to estimate the location of the target host. First, it determines the network path between a probe machine and the target host using the traceroute tool. Traceroute reports the DNS names of the intermediate routers where possible. Then GeoTrack extracts location information from the DNS names of recognizable routers along the path. Thus, it traces the *geographic path* to the target host. Finally, GeoTrack estimates the location of the target host as that of the last recognizable router in the path (i.e., the one closest to the target).

As noted in Section 2, traceroute-based approaches that extract geographic hints from router names have been proposed before (e.g., GTrace [15], VisualRoute [31]). However, we are not aware of work in the open literature on a quantitative evaluation of the traceroute-based approach to determining the geographic location of hosts. Our goal is precisely to do such an evaluation. Due to the logistic difficulties associated with obtaining and running existing traceroute-based tools, we decided to write our own tool based on GeoTrack to do large-scale experimentation. We have tested our tool over a large sample of IP addresses and found that its coverage is comparable to VisualRoute within the U.S. and in Europe.

4.1 Extracting Geographic Information from Router Names

Geographic information is typically embedded in the DNS name of a router in the form of a *code*, which is usually an abbreviation for a city, state, or country name. There is no standard naming convention for these codes. Each ISP tends to use its own naming convention. This makes the task of extracting location information from DNS names challenging.

Based on empirical data, we have observed that there are basically three types of codes that indicate location: city codes, airport codes, and country codes. Some ISPs assign DNS names to routers based on the airport code of the city they are located in. Since airport codes are a worldwide standard, such a naming convention greatly eases the task

[1] To be precise, DNS names are associated with router *interfaces*, not routers themselves. However, for ease of exposition we only use the term "router".

of determining the router's location. For example *sjc2-cw-oc3.sjc.above.net* refers to a router in San Jose, CA (airport code *sjc*). However, many ISPs use non-standard codes for cities. We have noticed that the city of Chicago, IL has at least 12 different codes associated with it (e.g., *chcg, chcgil, cgcil, chi, chicago*). We have also observed that many routers outside the United States have the country codes embedded in their names. For example, the router with the name *asd-nr16.nl.kpnqwest.net* is located in the Netherlands (country code *nl*). The country information can be very useful in (partially) validating the correctness of the location guessed based on city or airport codes.

We examined several thousand distinct router names encountered in the large set of traceroutes that we performed from our 14 probe locations. We compiled a list of approximately 2000 airport and city codes for cities in the U.S. and in Europe. Of the entire set of airport codes [27], our list only includes a relatively small fraction of codes that are actually used in router names. Since GeoTrack deduces location by doing a string match of router names against the codes, constructing a list with as few superfluous codes as possible decreases the chances of an inadvertent match.

To further reduce the chances of an inadvertent match, we divided the list of location codes into separate pieces corresponding to each major ISP (e.g., AT&T, Sprint, etc.). When trying to infer location from a router name associated with a particular ISP, GeoTrack only considers the codes in the corresponding subset.

There is the question of how router names are matched against the location codes. Simply trying to do a string match without regard to position of the matching substring may be inappropriate. For example, the code *charlotte*, which corresponds to Charlotte, NC in the eastern U.S., would incorrectly match against the name *charlotte.ucsd.edu*, which corresponds to a host in San Diego, CA in the western U.S. Through empirical observation, we have defined ISP-specific parsing rules that specify the position at which the location code, if any, must appear in router names associated with a particular ISP. We split the router name into multiple pieces separated by dots. The ISP-specific parsing rules specify which piece(s) should be considered when looking for a match. For example, the rule for Sprintlink specifies that the location code, if present, will only be in the first piece from the left (e.g., *sl-bb10-sea-9-0.sprintlink.net* containing the code *sea* for Seattle). The rule for AlterNet (UUNET) specifies that the code, if present, will only appear in the third piece from the right (e.g., *192.atm4-0.sr1.atl5.alter.net* containing the code *atl* for Atlanta).

4.2 Performance Evaluation

We compare the performance of GeoTrack and a Whois-based tool, NetGeo [14], both for university hosts drawn from the UnivHosts data set and for a more diverse set of hosts drawn from the FooTV data set. The latter consists of a random sample of 2380 client IP addresses drawn from the FooTV data set. While many of the FooTV clients connected via proxies, none of the university hosts was behind a proxy. For this experiment, we used the probe machine at UNC in Raleigh, NC as the source of all traceroutes.

We quantify the accuracy of a location estimate using the *error distance*, which we define as the geographic distance between the actual location of the destination host and the estimated location. In the case of FooTV, the "actual" location corresponds to the zip code recorded in the FooTV data set which, as noted in Section 3.5, may not be entirely accurate. Also, an IP address may be associated with multiple locations, either because it was allocated dynamically (say using DHCP [5]) or because it belonged to a proxy host (such as a Web proxy or a firewall). GeoTrack, on the other hand, would only make a single location estimate for a particular IP address. In our evaluation, we compute separate error distances corresponding to the many "actual" locations associated with an IP address.

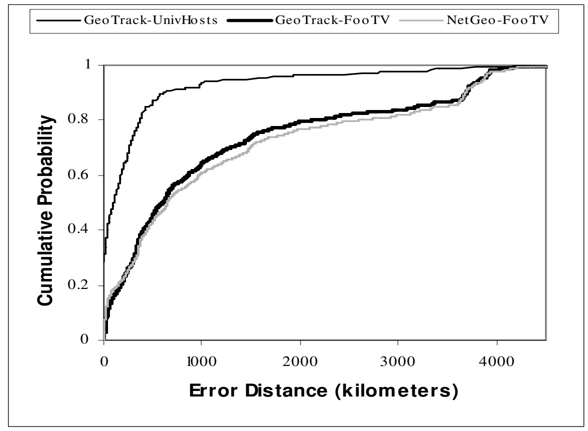

Figure 3: CDF of the error distance for GeoTrack and NetGeo.

Figure 3 shows the CDF of error distance for both GeoTrack and NetGeo. It is very interesting to note the similarity between the "NetGeo-FooTV" and "GeoTrack-FooTV" curves beyond the 70th percentile mark, and the distribution of distance of AOL clients from their proxies in Figure 1. GeoTrack determines the location of the AOL proxies as Washington, DC while NetGeo returns the location as Sterling, VA. The similarity in the curves can be attributed to the fact that these two locations are only about 35 km apart. (Moreover, AOL's proxies are also located in the same vicinity.)

We also observe that the performance of GeoTrack is only slightly better than that of NetGeo. GeoTrack exhibits a median error distance of 590 km and NetGeo a median of 650 km. Since many of the FooTV clients are behind proxies, neither Geotrack nor NetGeo is able to estimate the client's location accurately.

It is interesting to note that there is a significant difference in the performance of GeoTrack for the well-connected UnivHosts hosts as compared to that for FooTV clients. For instance, the median error distance is 102 km for the former while is is 590 km for the latter. The reason for this difference is that (a) none of the hosts in UnivHosts is behind a proxy, and (b) these hosts are well connected in the sense that a traceroute to them generally completes and yields a last recognizable router that tends to be close to the target host.

5. THE GEOPING TECHNIQUE

The GeoPing technique seeks to determine the geographic location of an Internet host by exploiting the relationship between network delay and geographic distance. GeoPing measures the delay to the target host from multiple sources (e.g., probe machines) at known locations and combines these delay measurements to estimate the coordinates of the target host.

5.1 Correlation between Network Delay and Geographic Distance

Conventional wisdom in the networking community has suggested that there is poor correlation between network delay and geographic distance [2]. This has largely been attributed to the presence of circuitous geographic paths in the Internet and bottlenecks links that cause congestion (and hence delay). However, in recent years, the Internet has grown at a very rapid pace, in terms of bandwidth as well as coverage (witness the rapid growth in the number and capacity of high-speed links, ISP points of presence, etc.). The richer connectivity (at least in well-connected portions of the Internet such as in the U.S.) often implies less circuitous routes.

To quantify impact of richer connectivity, we traced the network paths from several known locations to hosts in the UnivHosts data set. For each pair of hosts, we defined the *linearized distance* as the sum of the lengths of the individual hops along the path between the hosts. (We used GeoTrack to determine the geographic location of the intermediate nodes. We skipped over nodes whose locations could not be determined, so in general we might underestimate the linearized distance.) We compute the ratio of the linearized distance to the geographic distance between the hosts. The closer the ratio is to 1, the more "direct" (i.e., less circuitous) the network path is. Figure 4 shows the cumulative distribution of this ratio for paths originating from 3 different locations. The main observation we make here is that the ratio of linearized distance to geographic distance is close to 1 in the vast majority of cases. This implies that the corresponding network paths are not very circuitous.

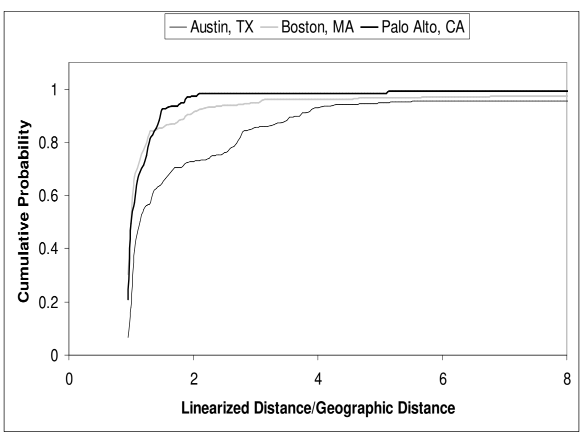

Figure 4: CDF of the ratio of linearized distance to geographic distance for Internet paths originating from three locations.

Congestion in the network may lead to significant queuing delays, which would also disrupt the relationship between network delay and geographic distance. To alleviate this problem, we gather several samples for the delay between two hosts and then pick the minimum among them. (This approach has been used in several networking protocols before, e.g., TCP Vegas [3].) While not perfect, picking the minimum enables us to eliminate much of the effect of congestion. Our experiments suggest that the minimum delay stabilizes once we have at least 10-15 delay samples.

The above approach would fail in the presence of special links (e.g., dialup, satellite, etc.) that have an inherent large delay that does *not* necessarily correlate with geographic distance. We discuss possible approaches to solving this problem in Section 5.3.

In the following sub-sections, we present delay measurements that support our contention that there is significant correlation between (the minimum) network delay and geographic distance. Although the correlation is not perfect, we are still able to exploit it to determine location at a coarse granularity. We present a robust algorithm for this in Section 5.2. We present experimental results that quantify the accuracy of this algorithm and also indicate the fundamental limitations of a delay-based approach.

5.1.1 Experimental Setting

We use the UnivHosts data set for performing our measurements. We perform traceroutes and ping measurements from 14 different sources (Figure 2) to all the 265 university servers in UnivHosts. After identifying the path from a given source to a host, we determine the *round-trip* delay to all intermediate routers using ping measurements. From multiple delay samples, we compute the minimum RTT to the destination and to each intermediate router in the path. We use GeoTrack to determine the physical location of intermediate routers. Using the data gathered for each source, we construct a large data set of [minimum delay, geographic distance] pairs corresponding to the paths from that source to the hosts in UnivHosts (and the intermediate routers).

5.1.2 CDF of Distance given Network Delay

We investigate whether there is a model that would enable estimation of geographic distance based on knowledge of network delay. For this purpose, we divide the delay range into several 10 ms wide bins and compute the CDF of geographic distance within each bin. (We decided to have a separate bin for the 0-5 ms delay range because we observed empirically that 5 ms often defines the threshold for a "metropolitan area". For instance, we found that more than 90% of the nodes within an RTT of 5 ms are located within a range of 50 km from the source.) So the delay bins we used to classify our measurements were: 0-5 ms, 5-15 ms, 15-25 ms, ..., 125-135 ms.

Figure 5 shows the CDF of geographic distance for our source host located in Seattle. Many of the delay bins exhibit distinct "cliffs" (i.e., sharp upswings) in the cumulative probability distribution for specific distance values. For example, the cliff around 1300 km for the 25-35 ms delay bin is mainly contributed by locations in the San Francisco Bay Area. The other noticeable trend is that as the delay increases from 0 to 80 ms, the cliff in the CDF shifts to the right. We observed similar trends for the probes at other locations as well.

While there is a definite trend in the cliffs of the CDF for each delay range, our results suggest that the relationship between delay and distance is not strong enough to be captured in a precise mathematical model. For small delay values (under 10 ms), we found that most of the hosts (over 90%) are within a radius of 300 km from the source. However for delay values more than 40 ms, we observed an

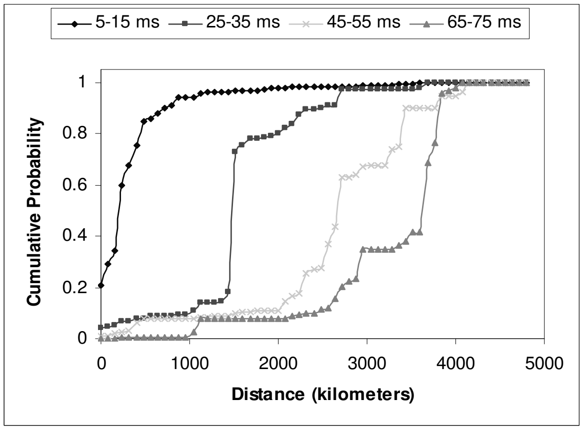

Figure 5: Cumulative Distribution of geographic distance for multiple delay ranges based on data gathered at the Seattle, WA probe location.

error of at least 300-400 km to obtain a 70% confidence in the distance estimate. We validated this for the data sets obtained from each of the 14 probe locations.

We also investigated how the relationship between delay and distance varies when we consider hosts belonging to distinct organizations located in the same geographic area. For this purpose, we considered probes located at Duke University and the University of North Carolina (UNC) located in the same metropolitan area on the U.S. east coast, and similarly Berkeley and Stanford on the west coast. We compared the CDFs corresponding to various delay ranges for each of these probe locations. The cliffs of the CDFs for Duke and UNC matched each other, and likewise for Berkeley and Stanford. This suggests that the cliffs of the CDFs are largely a function of the geographic location of a probe rather than the specific probe itself.

One limitation of our measurements is that most of our probe machines were located at university sites, many of which were connected to the high-speed Internet2 backbone [25]. The delay-distance relationship for nearby locations might not match quite as well if the probes were located at more heterogeneous sites with differing ISP connectivity. However, as we discuss next, our methodology for determining location is robust to such differences since we do not attempt to map directly from delay measurements to distance estimates.

5.2 Nearest Neighbor in Delay Space (NNDS)

We now discuss how GeoPing exploits the relationship between delay and distance to determine the geographic location of a host. Since we are unable to construct a precise and compact mathematical model that captures the relationship, we use an empirical approach, which we term *nearest neighbor in delay space (NNDS)*. NNDS is patterned after the nearest neighbor in signal space (NNSS) algorithm we had developed in the RADAR [1], a system to locate hosts in wireless LANs.

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. So the first step is to construct a *delay map* that records the relationship between delay and location. Each entry of the delay map contains: (a) the coordinates of a host at a known location, and (b) a delay vector, $DV = (d_1, \ldots, d_N)$, containing the measured (minimum) delay to the host from N probes at known locations. The delay map constitutes the *training* data and is constructed offline. Given a new target host, T, whose location is to be determined, we first measure the network delay to it from the N probes. We then construct a delay vector for T as $DV' = (d'_1, \ldots, d'_N)$. Finally, we search through the delay map to find a delay vector, DV, that matches DV' the best. To find the best match, we consider the delay vectors in the delay map as forming an N-dimensional *delay space* and find the "nearest" neighbor of DV' in this space. We use Eucledian distance as the measure of distance in delay space — the Eucledian distance between DV and DV' is $\sqrt{(d_1 - d'_1)^2 + \ldots + (d_N - d'_N)^2}$. Once the nearest neighbor in delay space has been found, the corresponding location recorded in the delay map is then GeoPing's estimate of the location of the target host T.

Several aspects of NNDS contribute to its robustness: (a) delay is measured from multiple distributed locations rather than a single location, (b) the minimum among several delay samples is considered rather than the individual delay samples, and (c) the delays measurements are used as a "signature" of a geographic location rather than being directly translated into distances and location coordinates.

Typically, the delay vectors corresponding to geographically proximate locations are clustered together in delay space. However, this is not essential for NNDS to be effective. Sites located in the same city but connected via different ISPs may form more than one distinct cluster in delay space. However, as long as the number of clusters remains small, NNDS will still be effective.

We now turn to evaluating the performance of GeoPing employing NNDS.

5.2.1 Experimental Results

We use the delay measurements from the 14 probe machines to the 265 hosts in UnivHosts to populate the delay map. We also use the hosts in UnivHosts as the target hosts for performance evaluation. Given a target host, T, in UnivHosts whose location we are trying to determine, we exclude all data points corresponding to T in the delay map before applying the NNDS algorithm. We study the impact of the number and distribution of probe machines on the accuracy of the location estimate. For a given number of probes (say n), we compute the mean error distance as the average over all the error distances corresponding to several geographically distributed placements of n probe 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. Due to the large number of possible combinations for certain values of n (such as $n = 7$), we do not necessarily consider all possible choices of n probes out of the set of 14.

Figure 6 shows several percentile levels of the error distance as a function of the number of probes. For example, the 75th percentile curve corresponds to the distance at which the CDF plot of mean error distance crosses the 0.75 probability mark. From Figure 6, 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

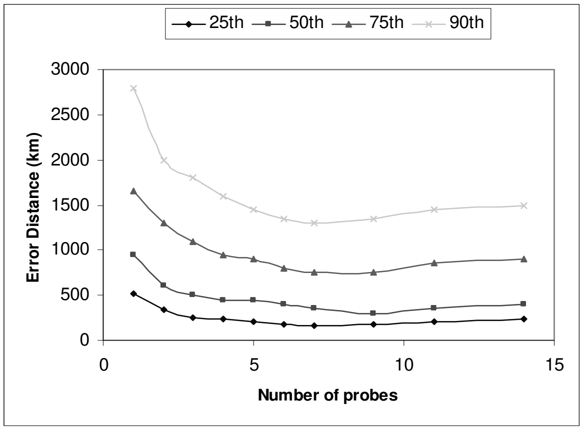

Figure 6: Error distance versus number of probes.

for higher values. This suggests that having 7 to 9 probes would be ideal for the NNDS algorithm. It is also encouraging to note that NNDS with 7 probes has an error distance of only about 150 km at its 25th percentile. Our results suggest that network delay can indeed be used to determine geographic location, albeit at a coarse granularity. We expect NNDS to perform even better with a delay map constructed using a more extensive training data set and plan to investigate this is future research.

We have also investigated the impact of various probe placement strategies on the accuracy of location estimation. We have examined the effects of probe placement on the error distribution. Our findings indicate that a geographically well-distributed set of probes yields better accuracy than a clustered set of probes. For instance, the median error distance with a probe each at Stanford and Illinois was about 19% lower (i.e., better) than a probe each at Berkeley and San Diego (both of which are on the U.S. west coast). However, we found that the placement of probes has a smaller impact on performance than the number of probes.

5.3 Miscellaneous Issues

Finally, we discuss a few miscellaneous issues pertaining to GeoPing and NNDS.

Other Statistical Methodologies

Besides the NNDS approach, we have also investigated other statistical techniques for estimating location from delay measurements. In particular, we tried constructing an approximate model that captures the relationship between delay and distance by generating a probability density function for every source based on a large set of measurements. However, none of the alternative techniques was able to match NNDS in terms of accuracy.

Impact of Congestion and ICMP Traffic

As mentioned earlier, network congestion can introduce significant noise in the delay measurements, thereby degrading the accuracy of GeoPing (and other delay-based approaches). Our experiments suggest that 10-15 delay samples are generally (but not always) sufficient to determine the minimum delay with high confidence (i.e., the minimum delay generally did not get any lower beyond the first 10-15 samples). However, sending or receiving several ICMP packets (for ping) to the target host from each probe location may be undesirable, both because it may aggravate congestion and because it may raise a flag with intrusion detection systems. We discuss a way of alleviating this problem in Section 6.5.

The Last Mile

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 that has a large delay (for example, a dialup link or a satellite link). While this is clearly problematic for GeoPing, we may be able to work around it in certain situations. For instance, if we are able to tell that the user is on a dialup line (say based on the observed bandwidth to the user or traceroute measurements), we could use GeoPing to determine the location of the last router (typically located at the dialup ISP's point-of-presence) 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 is different from GeoTrack and GeoPing in that it does not depend on active network measurements. Instead it uses knowledge of network routing information and location information for a few hosts to build a location map for a large subset of the IP address space.

GeoCluster operates as follows. First, the IP address space is broken up into clusters such that all hosts with IP addresses within a cluster are likely to be co-located[2], i.e., the addresses form a *geographic cluster*. Then, knowing the location corresponding to a few hosts in a cluster (and assuming the locations are largely in agreement), GeoCluster deduces the location of the entire cluster.

The key to the operation of GeoCluster is IP-to-location mapping information obtained from sources such as the ones mentioned in Section 3.5. (We discuss the general problem of obtaining such data in Section 6.5.) However, this mapping information tends to be *partial* in coverage (since it includes location information only for a relatively small subset of the IP address space) and possibly *inaccurate*. These problems limit the utility of the IP-to-location mapping data.

GeoCluster addresses both of these problems by clustering IP addresses according to their (likely) location. Clustering helps expand the coverage of the partial IP-to-location mapping information. The aggregation of location information also enables us to identify and eliminate outliers caused by inaccuracies in the individual location data points.

As an example, suppose we know that 128.127.126.0/24[3] 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 Seattle while that corresponding to one other IP address is Boston. Then we can reasonably deduce that the Boston data point

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

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

is erroneous and that all of the (256) IP addresses in this cluster (if they are indeed in use) are likely to correspond to hosts in (or near) Seattle.

6.1 Identifying Geographic Clusters

Identifying geographic clusters is a challenging problem. The basic approach used by GeoCluster is to combine partial IP-to-location mapping information with network routing information. We build on the work presented in [12] 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 Internet as a single aggregate, say as the address prefix 128.127.0.0/16, rather than as 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 [12]. We surmise that APs are also likely to 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 3.4). 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 more detailed knowledge of how a large aggregate is subdivided by the intra-domain routing protocol used within the ISPs. However, obtaining such information was not feasible for us, so we only use inter-domain routing information derived from BGP.

In summary, our baseline GeoCluster algorithm, which we term *BGPonly*, discovers APs based on BGP data and surmises that these APs are geographic clusters. However, as explained above this conjecture may not be always correct, for instance when ISPs only advertise large aggregates. We now present a sub-clustering algorithm designed to address this problem. We term the variant of GeoCluster that incorporates this algorithm as *BGP+subclustering*.

6.2 Sub-clustering Algorithm

The BGP+subclustering variant of GeoCluster depends only on inter-domain BGP data just like BGPonly. But the novel idea is to use partial IP-to-location mapping information 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 the pseudocode for GeoCluster, including the subclustering 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
```

Here is a simple example that illustrates the operation of the sub-clustering algorithm (assume that $cthresh = 15$). Consider an ISP who owns the address space 152.153.0.0/16.

Suppose that the ISP has allocated half of the address space (152.153.0.0/17) to a customer in New York, and a quarter each (152.153.128.0/18 and 152.153.192.0/18) to customers in Dallas and San Francisco, respectively. Suppose that the partial IP-to-location mapping information indicates that the location is New York for 50 IP addresses in 152.153.0.0/17, Dallas for 20 addresses in 152.153.128.0/18, and San Francisco for 10 addresses in 152.153.192.0/18. The ISP only advertises the 152.153.0.0/16 prefix via BGP, so the sub-clustering algorithm starts with 152.153.0.0/16 as the presumed geographic cluster. However, there is not sufficient consensus on the location of this cluster, so the cluster is subdivided into two halves, 152.153.0.0/17 and 152.153.128.0/17. There is sufficient consensus for the former address prefix, so the algorithm declares 152.153.0.0/17 as a geographic cluster with its location as New York. However, 152.153.128.0/17 still lacks consensus, so it is subdivided into 152.153.128.0/18 and 152.153.192.0/18. There is sufficient consensus on the location corresponding to 152.153.-128.0/18, so it is declared as a geographic cluster with its location as Dallas. However, there are fewer than *cthresh* IP-to-location data points for 152.153.192.0/18, so the algorithm terminates without declaring it as a geographic cluster.

The effectiveness of the sub-clustering 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.

We have not specified how it is determined whether a set of locations is geographically clusterered or how the consensus 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 using latitude and longitude. So we compute a *composite* location using linear averaging of the latitudes and longitudes[4]. We also compute a dispersion metric as follows: $dispersion = \sum_{l \in L} dist(l, l_{avg}) / \mid L \mid$, where L is set of location data points corresponding to the cluster, l_{avg} is the composite location computed via averaging, and $dist(x, y)$ is the geographic distance between the locations x and y. Intuitively, the dispersion quantifies the geographic extent or spread of a cluster. We decide whether a set of locations is geographically clustered by checking whether the dispersion is smaller than a threshold.

In case location information is coarse-grained (e.g., states), we test whether there are at least *cthresh* data points in the cluster and whether at least a threshold fraction, $fthresh$, of the points agree on location. If both conditions are met, then the consensus location is assigned to the entire cluster. As mentioned earlier, this aggregation procedure helps eliminate errors due to erroneous location information.

6.3 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, the proxy or firewall typically 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.)

The sub-clustering algorithm in GeoCluster 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) deduce 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 corporate 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 dispersed 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 find sufficient consensus to be able to identify any geographic clusters, so it will not try to map the "client" IP address to a location. We believe this is an important property of the sub-clustering algorithm because for many applications a highly inaccurate location estimate may be strictly worse than no location estimate at all. For instance, displaying a generic advertisement on a New York user's screen would probably be better than mistakenly displaying an advertisement tailored for California residents.

6.4 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 6.2 (*BGP+subclustering*). We compare both variants against a simplistic approach that ignores BGP information and assumes that all APs to have a 24-bit prefix length (*/24-clusters*).

6.4.1 Locating hosts in UnivHosts

We first analyze the ability of the BGPonly variant of GeoCluster in determining the location of hosts in the UnivHosts data set (Section 3.3). We use partial IP-to-location mapping data contained in the FooTV data set as input. We convert each zip code contained in the FooTV data to the corresponding (approximate) latitude and longitude. We then cluster the (IP,latitude,longitude) data points using BGP address prefix (AP) information and compute the composite location for each AP (Section 6.2). Given a target IP address, we find the matching AP using longest prefix match and declare the corresponding (latitude,longitude) pair as the location estimate. We quantify the accuracy of the location estimate using the error distance.

Figure 7 shows the CDF of error distance for GeoCluster computed over the 265 university hosts. We also show the CDFs of GeoTrack and the best case of GeoPing (using 9 probe machines) for comparison. GeoCluster is able

[4]While not strictly correct, such averaging is a good approximation when the individual points are close to each other

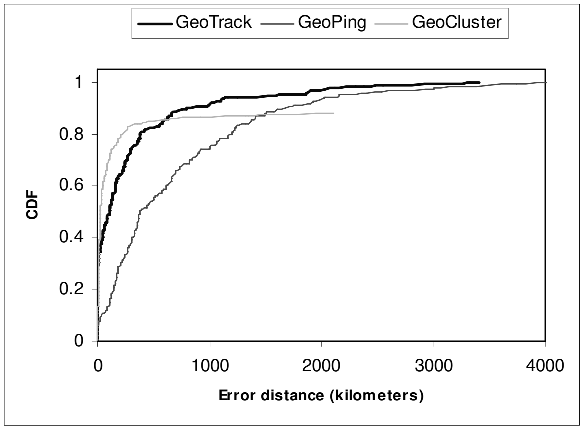

Figure 7: CDF of the error distance computed over the UnivHosts data set for GeoTrack, GeoPing, and GeoCluster.

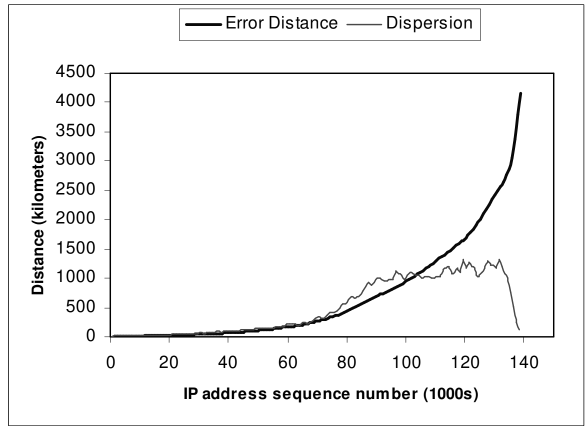

Figure 8: The error distance and the dispersion for hosts in bCentral.

to deduce the location of only 233 out of the 265 university hosts (i.e., about 88% of the 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 whose location it is able to determine, GeoCluster significantly outperforms both GeoTrack and GeoPing. For instance, the median and 80th percentile marks for GeoCluster are 28 km and 226 km, respectively. The corresponding numbers are 102 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 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.

6.4.2 Locating hosts in bCentral

We now analyze the performance of GeoCluster using the much larger bCentral data set. This data set contains 181246 unique IP addresses and their corresponding zip codes. (As noted in Section 3.5, the zip code information may not be entirely accurate. Hence, unlike the case of university hosts, we are not entirely certain of the true locations of the bCentral client hosts.) As before, we use the BGPonly variant of GeoCluster, with the FooTV and the BGP data sets as inputs to prime the GeoCluster algorithm.

For each IP address in bCentral, 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 8. 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 bCentral data set than for the UnivHosts data set.

The main reason for the worse performance is that the bCentral data set is much more diverse than the UnivHosts data set. Unlike UnivHosts, many of the IP addresses in bCentral fall within 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 proxies). Hence GeoCluster is only able to determine location accurately for a smaller fraction of the hosts.

Given the wide range of error distances 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 the *dispersion* metric for each AP (Section 6.2). 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 8, which depicts the (smoothed version of) dispersion curve for the bCentral 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 the 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 bCentral. Based on this examination, we have come to the conclusion that the discrepancy is caused mainly by clients that dial in remotely. For example, bCentral contains the IP address 140.247.147.42 (DNS name *roam147-42.student.harvard.edu*), which presumably corresponds a dial up connection at Harvard University in the northeastern corner of the U.S. (and which is what GeoCluster deduces the location to be). However, the corresponding location recorded in the bCentral data set is Portland, Oregon, 4000 km away in the northwestern corner of the U.S. We hypothesize that this discrepancy is due to a user in Portland remotely dialing in to a modem bank at Harvard and then connecting to bCentral. However, it is difficult to know for sure — the Portland location may simply be erroneous, in which case the (large) error distance would be misleading.

Our results suggest 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 often able to tell when a location estimate is likely to be accurate and when it is not.

6.4.3 Importance of the sub-clustering algorithm

Thus far we have considered the BGPonly variant of GeoCluster, which only uses AP information derived directly from BGP data. We now turn to the BGP+subclustering variant that employs the sub-clustering algorithm (Section 6.2) 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 offers.

We use the partial IP-to-location mapping data obtained from Hotmail (Section 3.5) as input to the sub-clustering algorithm. Recall that the location information in Hotmail is at the granularity of states. As discussed in Section 6.2, we deem an AP to correspond to a geographic cluster if it contains at least $cthresh$ data points drawn from the IP-to-location mapping data set and at least a fraction $fthresh$ of those data points agree on location (i.e., correspond to 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 bCentral as the test data. The location information in bCentral is at the granularity of zip codes whereas that in Hotmail 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 bCentral to the corresponding states. We then compute the *zipcenter* of each state by averaging the coordinates of the zip codes contained within that state (Section 3.5). 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.

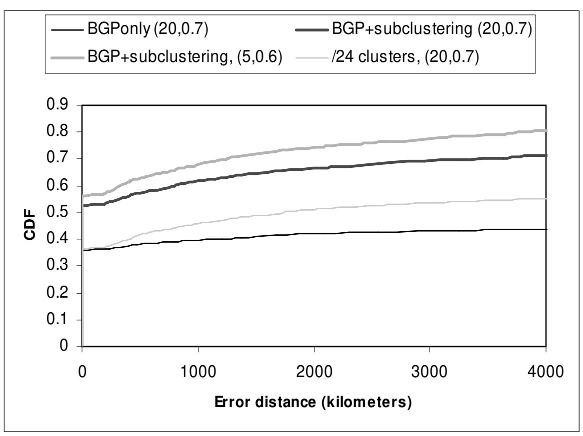

Figure 9: CDF of the error distance (computed at the granularity of states) for the BGPonly and BGP+subclustering variants of GeoCluster, and for the /24-clusters method.

Figure 9 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 only for 36% of the hosts. The reason is that BGPonly is often stuck with large and geographically dispersed APs obtained directly from BGP data while the sub-clustering algorithm is often able to break these down into smaller and geographically more 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.

6.5 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 (for example, as in the case of FooTV). Since it only considers such information in 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 [32], 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 presumably much larger number of) registered and casual users 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 [23]) that will employ GeoCluster (and possibly 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 would not need to be concerned with the details of the location mapping techniques.

On a final note, we believe that the idea in GeoCluster of clustering hosts together based on geographic location may be quite useful in conjunction with GeoTrack and GeoPing. Both GeoTrack and GeoPing conduct *active* measurements by injecting traffic into the network. This may be undesirable for several reasons (network load, security, etc.). Clustering can alleviate this problem by making it unnecessary to do pings or traceroutes to *each* new target host. It may suffice to do these measurements to just a fraction of the hosts within an address prefix cluster. In fact, GeoTrack and GeoPing, used in this manner, can help GeoCluster construct the partial IP-to-location mapping that it needs.

7. SUMMARY AND CONTRIBUTIONS

In this paper we have examined the interesting but challenging problem of determing the geographic location of an Internet host knowing only its IP address. We have designed and evaluated three distinct techniques, collectively referred to as IP2Geo, to address 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. These techniques span a broad spectrum. Our evaluation of these techniques was based on extensive and varied data sets.

Our findings suggest that GeoCluster is the most promising one of the IP2Geo techniques. The median error distance for GeoCluster varies from 28 km for well-connected university hosts to a few hundred kilometers for a more heterogeneous set of clients. Importantly, however, GeoCluster is self-calibrating in that the *dispersion* metric offers an indication of how accurate a location estimate is likely to be. Furthermore, the *sub-clustering* technique is often able to infer more fine-grained (geographic) structure in Internet address ranges than is present in BGP routing data. Both these features make GeoCluster more suitable than the other techniques in the presence of clients that connect via proxies. Finally, GeoCluster is passive in that it does not inject extra traffic into the network.

Our investigation of GeoTrack and Whois-based techniques reveals the fundamental limitation due to proxies. Our evaluation of GeoPing suggests that contrary to conventional wisdom there is a significant correlation between network delay and geographic distance that can be exploited to determine coarse-grained location. We believe this will be the case even more as the Internet becomes richly connected.

Our study also indicates that geography can be an interesting tool for analyzing the behavior of network routing. The ratio of *linearized* distance to geographic distance is indicative of how "direct" a network route is. A large ratio may be indicative of an anomalous route. For instance, by computing this ratio, GeoTrack was able to automatically flag an a highly circuitous route from Austin, Texas to Kentucky via California, New Jersey, and Indiana!

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

Acknowledgements

George Chung, Alex Daunys, Cheng Ku, and Dave Quick helped us obtain the Hotmail, bCentral, and FooTV data sets. Craig Labovitz helped us gather BGP data. Donald Wysocki provided us with geographic data 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 probe machines. Randy Katz and the anonymous SIGCOMM reviewers provided useful feedback on an earlier version of this paper. We would like to thank them all.

8. REFERENCES

[1] P. Bahl and V.N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. *IEEE Infocom*, March 2000.
[2] G. Ballintijn, M. van Steen, and A.S. Tanenbaum. Characterizing Internet Performance to Support Wide-area Application Development. *Operating Systems Review, 34(4):41-47*, October 2000.
[3] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. *ACM SIGCOMM*, August 1994.
[4] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE project. *ACM Mobicom*, August 2000.
[5] R. Droms, Dynamic Host Configuration Protocol, *RFC-1531, IETF*, October 1993.
[6] P. Enge and P. Misra, The Global Positioning System, *Proc. of the IEEE*, January 1999.
[7] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. *IEEE Infocom*, March 2000.
[8] K. Harrenstien, M. Stahl, E. Feinler, NICKNAME/ WHOIS, *RFC-954, IETF*, October 1985.
[9] Andy Harter and Andy Hopper. A Distributed Location System, for the Active Office. *IEEE Network* Vol.8 No.1, January 1994.
[10] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, The Anatomy of a Context-Aware Application, *ACM Mobicom*, August 1999.
[11] V. Jacobson, Traceroute software, 1989, *ftp://ftp.ee.lbl.gov/traceroute.tar.gz*
[12] B. Krishnamurthy and J. Wang. On Network Aware Clustering of Web Clients. *ACM SIGCOMM*, August 2000.
[13] J.Y. Li et al. A Scalable Location Service for Geographic Ad-Hoc Routing. *ACM Mobicom*, August 2000.
[14] D. Moore et.al. Where in the World is netgeo.caida.org? *INET 2000*, June 2000.
[15] R. Periakaruppan, E. Nemeth. GTrace – A Graphical Traceroute Tool. *Usenix LISA*, Nov 1999.
[16] U. Raz. How to find a host's geographic location. *http://www.private.org.il/IP2geo.html*
[17] D. C. Vixie, P. Goodwin and T. Dickinson. A Means for Expressing Location Information in the Domain Name System, *RFC-1876, IETF*, January 1996.
[18] Akamai Inc. *http://www.akamai.com/*
[19] America Online (AOL). *http://www.aol.com/*
[20] BBNPlanet publically available route server, *telnet://ner-routes.bbnplanet.net.*
[21] bCentral. *http://www.bcentral.com/*
[22] Digital Island Inc. *http://www.digitalisland.com/*
[23] DoubleClick, *http://www.doubleclick.com/*
[24] Hotmail. *http://www.hotmail.com/*
[25] Internet2. *http://www.internet2.org/*
[26] IP to Latitude/Longitude Server, University of Illinois *http://cello.cs.uiuc.edu/cgi-bin/slamm/ip2ll*
[27] List of Airport Codes. *http://www.mapping.com/airportcodes.html*
[28] MERIT Network, *http://www.merit.edu/*
[29] NeoTrace, A Graphical Traceroute Tool *http://www.neoworx.com/products/neotrace/default.asp*
[30] U.S. Gazetteer, U.S. Census Bureau, *http://www.census.gov/cgi-bin/gazetteer.*
[31] VisualRoute, Visualware Inc., *http://www.visualroute.com/*
[32] Yahoo Inc., *http://www.yahoo.com/*