DISTRIBUTED TRACEROUTE APPROACH TO GEOGRAPHICALLY LOCATING IP DEVICES

by

Gene Michael Connolly

A Thesis Submitted in Partial Fulfillment
of the Requirements for a Degree with Honors
(Computer Science)

The Honors College

University of Maine

May 2003

Advisory Committee:

Anatoly Sachenko, Professor of Computer Science, Advisor
George Markowsky, Professor and Chair of Computer Science, Supervisor
Thomas Wheeler, Assistant Professor of Computer Science
Sharon Tisher, Lecturer in Honors and Instructor in Resource Economics and Policy

Acknowledgments

This project has, no doubt, been my largest academic undertaking heretofore. Had it not been for the superb guidance and direction offered to me by my thesis committee, led by Anatoly Sachenko and George Markowsky, it would not have achieved its full potential. Additionally, it would be remiss if I did not thank my friends and family, especially my roommates, for their patience and support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Project Rationale

Following the terrorist attacks that occurred on September 11, 2001, the United States became all too aware that the current state of homeland security was insufficient and under appreciated. Currently, enhancing Homeland Security is of foremost importance, and is a new challenge that certainly will not be over anytime soon. Homeland security is viewed in a new perspective because terrorism is a form of war that is different from what has been seen, and because what is considered the 'Homeland' is also drastically changing.

In the time between the two attacks on the World Trade Center, February 23, 1993 and September 11, 2001, the United States has pioneered a revolution in information technology by popularizing a worldwide Internet. This revolution effectively expanded the 'Homeland' of the States past the east and west coast into a new, uncharted cyberspace.

The Internet poses many questions and problems in regards to homeland security. Many of the properties of the Internet allow its users to remain anonymous in several respects. This not only makes it difficult to trace action on the Internet, but it also leaves users more uninhibited in using the Internet as they please. With respect to homeland security, this has led to an increase in terrorist communication on the web, with few means of determining whether the content exists within American borders and where.

Devising a method of geographically locating terrorist content on the Internet is the first step in securing the homeland on the cyberspace front.

1.2 Background

This paper proposes the design of a distributed approach to the problem of geographically locating Internet Protocol (IP) addresses. However, the intent of this project is not to propose a new solution, but rather to investigate and evaluate existing methods and solutions to the problem. The design of the distributed approach is the product of that investigation.

The investigation examines several facets of the IP location problem. Available products and projects involved with IP location are examined. Their examination for accuracy, methodology, and rationale not only reveals the state of the art, but also sets the stage for the following steps in the investigation. As it stands, there exist many current projects built for many different applications in IP location. These projects, however, are created on the foundation of only a few different methodologies. An analysis made of each of these methods reveals their strengths and weaknesses.

Traceroute, a utility for network analysis, and the WHOIS database, a source of network information, comprise two of the most used and most useful tools for IP location. They also represent fundamentally different approaches: network analysis and databases, respectively. Both insufficient as stand-alone solutions, they use each other to create acceptable results. However, no combination of traceroute and WHOIS, and any other methodology for that matter, is going to produce precise and correct results with certainty.

In the problem of attempting to map cyberspace to geographical space, possibly the most important component of investigation is exploring the limitations and the scope of geographically locating IP addresses. There are several aspects of how the Internet was designed and how the Internet is monitored that prevent IP location from being certain in correctness or precise in accuracy.

Following the investigation, a design where multiple traceroutes are distributed over networks and geogrpahy spawned from several concerns regarding the state of IP location. First, none of the current solutions handle the scope and limitations of the problem effectively. Second, the methods being used to evaluate IP location are not provided the geographical basis needed to solve a geographical problem.

The distributed approach proposed is not a radically new approach to the problem. In fact, the previously mentioned tools, traceroute utility and WHOIS database – which are the basis for many contemporary IP location projects – are used as the basis for this location project. The distributed approach, however, executes the traceroute analysis from multiple geographically diverse locations, approaching the problem with a geographic basis, provides results conforming to the scope and limitations of the problem.

Chapter 2

Statement of Problem

Before examining the methods of IP location that are critical components to the distributed approach, it is important to formally define the problem. By examining the work that has been completed in the field and by examining the limitations with which the problem can be solved, the analysis performed on the methods may be focused on taking the appropriate next steps in advancing the state of IP Location.

2.1 Literature Search

Mapping of the Internet to geographical space is a problem that has garnered a fair amount of research and investigation. As the Internet becomes more integrated with many facets of our culture, the applications of being able to geographically map cyberspace continue to increase. Currently, there exist several methods and software systems to handle such a problem, as well as documentation regarding related issues. However, as there are many applications to the solution solution of determining a computer's location, many of these solutions are fundamentally different in their approach, as they attempt to obtain different types of results:

Web Hosting & Commercial sites - Attempting to obtain the geographical position of customers visiting commercial sites exists to benefit the marketing techniques associated with the website. By obtaining the geographical denomination of the customers, administrators are able to customize content. This does demand accurate geographical locating and the granularity of the result is not a priority. Often determining the state or country is enough.

Tracing Spammers & Hackers – The attempt to hunt down hacking activity on the Internet has increased as the problem has increased. Barring several extreme exceptions however, often the approach to handling these problems is by locating and contacting the hacker's ISP. This is not geographical location of the computer in question, and often can be accomplished with a simple WHOIS database lookup for network information.

Network Optimization - By determining the geographical path data travels on the Internet, one can analyze the efficiency of a network. This does not, however,

emphasize the accuracy of the geographical location regarding the destination computer.

Other Purposes – There exist many more applications yet; in IP location they are often a simple variation of the ones listed above. Other investigations have been simply an effort to document the map of cyberspace. The geography of the Internet can be an indicator of many trends in economics and business.

For the purposes of this investigation to geographically locate potentially dangerous web content, the following studies – which largely fall into the categories listed above – have been great resources.

It would be remiss if it were not mentioned that additionally there exists a wealth of information regarding the topic on hacker web pages, newsgroups and bulletin boards. Occasionally this information does not adapt effectively to efficient methodology (Web Content Analysis / Natural Language Processing) or the methods are archaic (Day-Time Server for Time Zone). Moreover, methods maybe intrusive to private networks and databases.

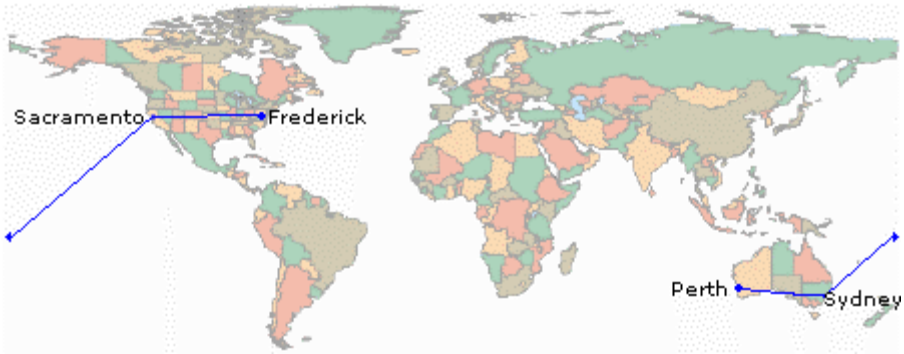


Figure 2.1: VisualRoute Visual Display [21]

Hop	%Loss	IP Address	Node Name	Location	Tzone	ms	Graph	Network
0		12.88.115.23	-	*			0 528	AT&T ITS
1		199.70.3.58	-	Parsippany, NJ		94		AT&T EasyL
2		199.70.3.49	-	Parsippany, NJ		139		AT&T EasyL
3		12.122.253.24	gbr6-p21.n54n	New York, NY, U	-5.0	128		AT&T ITS
4		12.122.5.114	gbr3-p90.n54n	New York, NY, U	-5.0	185		AT&T ITS
5		12.123.1.121	ggr1-p360.n54	New York, NY, U	-5.0	157		AT&T ITS
6		192.205.32.17	att-gw.ny.verio.r	New York, NY, U	-5.0	174		AT&T Data (
7		129.250.2.14	p4-1-3-0.r01.ch	Chicago, IL, US	-6.0	203		Verio, Inc.
8		129.250.2.253	p4-6-0.r00.chcg	Chicago, IL, US	-6.0	197		Verio, Inc.
9		129.250.4.89	p4-4-0.r00.dllst	Dallas, TX, USA		234		Verio, Inc.
10		129.250.3.74	p4-1-0-0.r01.dll	Dallas, TX, USA		221		Verio, Inc.
11		129.250.2.41	p1-0-0-0.r01.or	Orem, UT, USA	-7.0	269		Verio, Inc.
12		129.250.29.20	pvu1.vwhpvu1.v	Provo, UT, USA	-7.0	252		Verio, Inc.
13		**192.41.43.189**	visualroute.com	Highland, UT 8		265		Icon Develo
Roundtrip time to visualroute.com, average = 265ms, min = 195ms, max = 448ms -- 20-Apr-01								

Figure 2.2: VisualRoute Traceroute [21]

Project Profile: VisualRoute

Designer/Company Background: *Visualware*. Incorporated in January of 2001, Visualware offers software solutions in the Internet Security and Performance Management markets. The company offers a family of products that aid e-commerce companies in identifying and tracking network hackers, obtaining customer information for marketing and quickly locating network problems for immediate response. Their software products have won several awards and much recognition from such sources as *ZDNet* and *Microsoft Certified Professional Magazine.*[21]

Rationale for Project/Applications: VisualRoute, Visualware's most popular and robust software program, is designed to analyze Internet performance and as well as provide geographical information regarding routers, servers, and other IP devices. This has applications both in network analysis and the tracking of hackers/spammers.

IP Location Approach: Traceroute, WHOIS, Ping.

Specific Methodology: This program is an easy to use, highly integrated method of using the three network tools: traceroute, WHOIS, and ping. Traceroute populated a list of computer hostnames that the data travels through to reach its destination. Visualroute extracts location codes from these hostnames revealing the geographical path the data traveled. This approach is 'route' based, in that its focus is to obtain the general path that the data traveled, and not the specific location of the destination. Traceroutes are verified multiple times, in order to determine the optimal route by which the data travels.

Features: The program is extremely user friendly and thorough. Its selling point is its ability to visually represent the path of the data on a map of the world, seen in Figure 2.1. The product offers many additional features as well. As Figure 2.2 demonstrates, the traceroute utility uses WHOIS database lookups to determine the different networks the data travels upon. It attempts to determine the time zone, as well as do an analysis of the time required to reach each node. The program also provides server software as well, allowing web access to VisualRoute. None of these are features that could not be accomplished with effective use of a computer terminal and the UNIX operating system, but VisualRoute presents it in a very user-friendly format.

Project Analysis: VisualRoute is, no doubt, the most reliable, most accurate, most comprehensive and easy-to-use traceroute utility on the market. What it offers, however, in accurate, granular geographical information is achieved by its well-developed knowledge base, and not in methodology or approach.

Project Profile: NetWorldMap/GeoBytes

Designer/Company Background: GeoBytes, Inc. is a company dedicated to offering geographical information and geographical web customization. The company is an extension of the NetWorldMap Project. [17][9]

Rationale for Project/Applications: The NetWorldMap project is an investigation of the commercial applications of geographical information, from web traffic analysis to geographically targeted web content. This requires that the data be accurate to the granularity of a unified political region, such as a city, county, or state. Additionally it requires that the processing is very fast, in order to produce effective custom web content.

IP Location Approach: Offering a more unique approach, NetWorldMap determines geographical information by acquiring location data from willing participants, with the underlying belief that the distribution of IP addresses is generally locally clustered. The NetWorldMap web page requests simply that its visitors submit the city from which their computer is located.

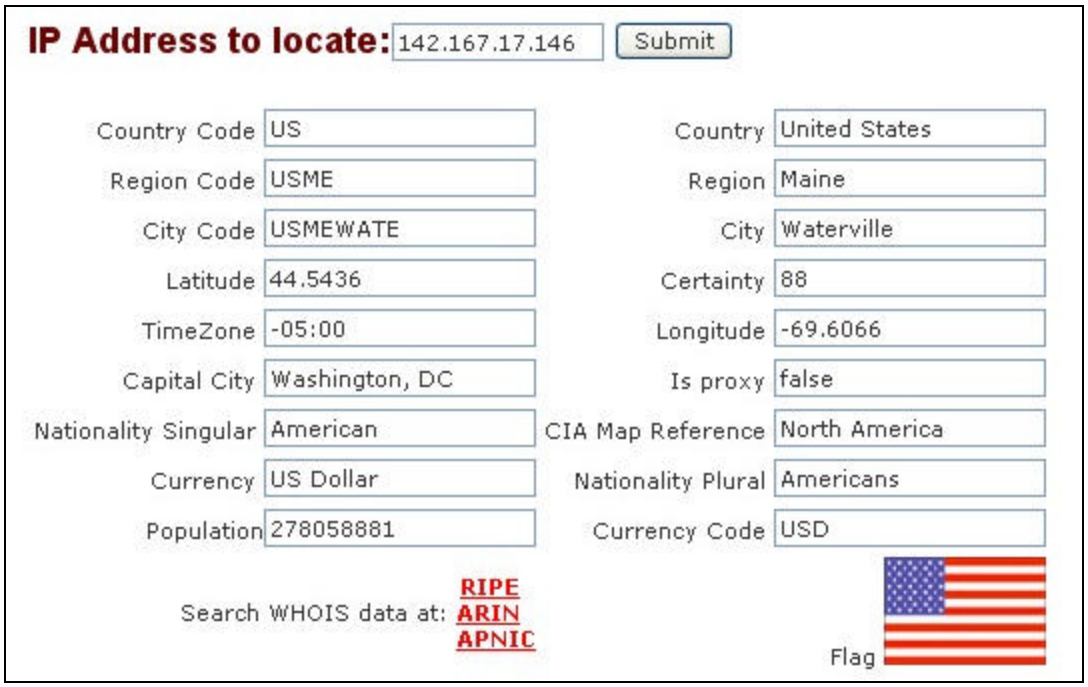


Figure 2.3: Screen shot of GeoByte's results [9]

Specific Methodology: Although specifics of the process are not revealed, it is clear that the knowledge base for such a project has been received by users that have willingly disclosed there geographical location to accompany their IP address to the project. Algorithms that use the voluntarily submitted data to reveal a geographical location then are

able to process unknown IP addresses. Ultimately, the system depends on a large amount of diverse and accurate data received by visiting web surfers.

Features: Geobytes offers easy web site integration, as well as a wealth of additional knowledge that was derived from the destinations location, such as latitude and longitude, currency, local time, language, etc. They also offer a field that offers a measure of certainty. For the most part this information is irrelevant to the geographical location. Examples of this data maybe seen in Figure 2.3, a screen shot of their web page.

Analysis: NetWorldMap certainly offers a unique approach to the problem of geographical location. It does, however, depend on the reliability of a couple sources: 1) People are providing accurate data from diverse locations. 2) Similar IP addresses have a tendency of being clustered together. Given these two qualities, it is still difficult to obtain a sufficient amount of data for the process to be effective.

Project Profile: GTrace/NetGeo

Designer/Company Background: NeoGeo is a project led by The Cooperative Association for Internet Data Analysis (CAIDA). CAIDA, which is based at the University of California San Diego, is a "collaborative undertaking among organizations in the commercial, government, and research sectors aimed at promoting greater cooperation in the engineering and maintenance of a robust, scalable global Internet infrastructure."[14]

Rationale for Project/Applications: One of the objectives of CAIDA is to develop methodologies and techniques for Internet Traffic Prototype and Visualization; Mapping IP devices to geographical locations one of the more difficult gaps to bridge. In this project, Internet geography is required for the purposes of Internet analysis.

IP Location Approach: DNS LOC, Traceroute, WHOIS, TLD Country Codes.

Methodology: NetGeo attempts several methods for determining geographical location. Upon a location request, the system checks its local database to see if that IP address has been successfully searched for previously. If this fails, the DNS LOC entry is checked to determine if it carries a Latitude/Longitude field. Next, a traceroute is performed and information is extracted in the same manner as the SarangWorld and VisualRoute projects. A WHOIS database lookup as well as examining the domain name for country codes are the last resorts for location. The different approaches are attempted in order of descending reliability.

Features: Because the program approaches the geographical location problem using several methods, it provides excellent information regarding the accuracy of the geographical results, as well as information regarding the method by which they were discovered. Additionally, the design of the system is extremely extensible; making it easy to incorporate new sources and databases for geographical investigation. GTrace, displayed in Figure 2.4, is a front-end graphical user interface that uses the NetGeo system. In additions to ease of use, it provides a visual world map representation of the results.

Analysis: The multiple approach design of NetGeo and GTrace is a logical decision for the designer of a geographical location system. Unfortunately, many of the approaches have not fully realized their potential. DNS LOC for example, (geographical information embedded in the Domain Name Server) is available on very few domains. The WHOIS server is only accurate on local and last-mile ISP's. Ultimately, the system all too often requires a traceroute to determine the location.

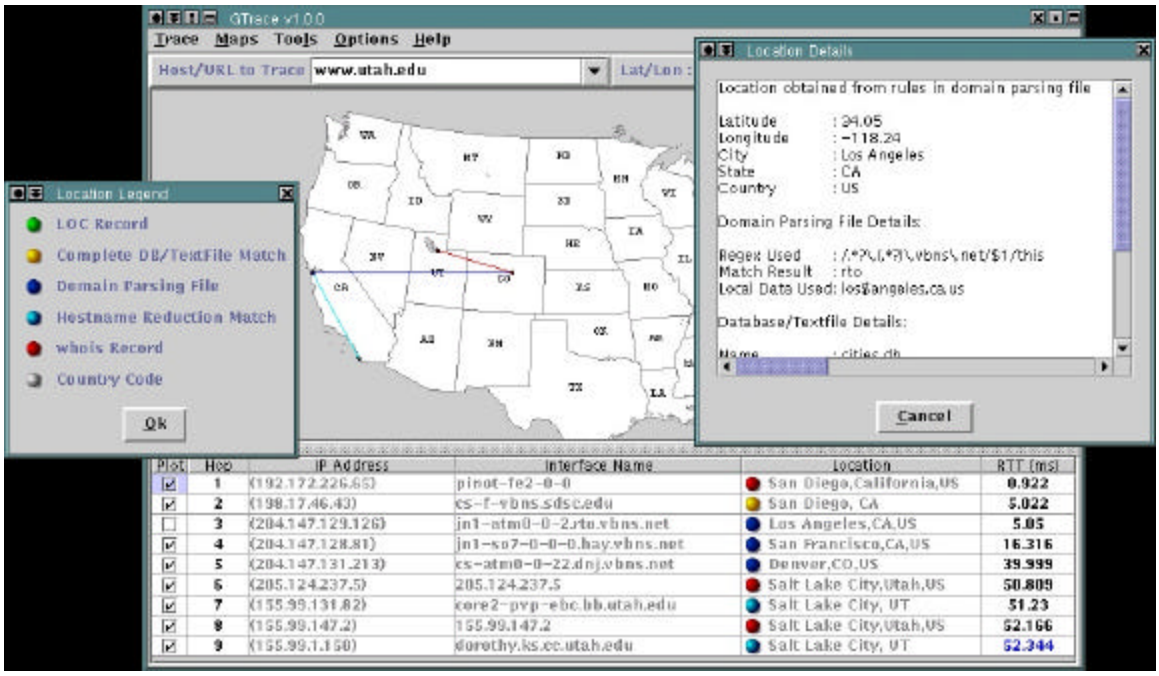


Figure 2.4: Screen shot of GTrace[15]

Project Profile: SarangeWorld Traceroute Project

Designer/Company Background: Sarang Gupta, of Albuquerque, New Mexico is the designer and developer of this traceroute project. Sarange attended the University of New Mexico, and then Stanford University for his Bachelor's and Master's Degrees in Mathematics (respectively). The SarangWorld project was developed in early 2001. [18]

Rationale for Project: The project, started as an intellectual pursuit, was developed to effectively map the path of a traceroute to geographical locations in addition to best locate the target computer. Explicitly stated in the documentation is that the project is not to be used for the identification or tracking of spammers/hackers.

IP Location Approach: Traceroute.

Methodology: This Perl script requests a traceroute from one of several remote traceroute servers. The hostnames at each traceroute step are then tested against a collection of patterns that represent the naming scheme of many networks and ISP's. Data is extracted from matching patterns, such as city abbreviations and airport codes are mapped to geographical locations using a database that also contain the latitude and longitude of the locations.

Features: SarangeWorld Traceroute Project is very much a 'work-in-progress'. It is fairly stripped down and its knowledge base of network patterns and database of city codes could be more comprehensive. Additionally, each request takes along the order of 1-3 minutes to complete. It does, however, disclose substantial documentation. Additionally, it offers the ability to run requests from numerous traceroute servers around the world.

Analysis: This project is a great source of documentation regarding the process that is used by many commercial traceroute projects, such as VisualRoute. It also contains a solid amount of available data that has been used as a springboard for the IP Locator Prototype. As stated in documentation of the project, however, it is fairly unreliable and not fully realized as an effective tool for location.

Project Profile: Zooknic Internet Geography Project

Designer/Company Background: Matthew Zook, Ph.D. has been involved in many efforts over the last five years that have investigated the topography of the Internet. The main focus of his investigation is the use of domain names and their effects on the economics and geography of business. [25]

Rationale for Project: The Internet Geography Project is unlike the other documented projects, in that it is not concerned with the accurate geographical location of IP devices. The investigation pertains to the geographical distribution of domain names. In this age of e-commerce and the impact of the Internet upon the economy, the distribution of domain names can often better indicate the economic growth and trends of particular regions, than many conventional methods.

Analysis: The inclusion of this project in the documentation of mapping Internet geography is to offer an alternative perspective in the geographical construction of the Internet infrastructure. Figure 2.5 displays the distribution of domain names in New York City, a city that accounts for the location of 5.9% of the worlds registered Top Level Domains. With all efforts to locate IP addresses and web pages, most of the information is maintained remotely in "cyber-cities of information". Should information or a computer be discovered in New York City, it is still often of little help in discovering the party that is responsible for it.

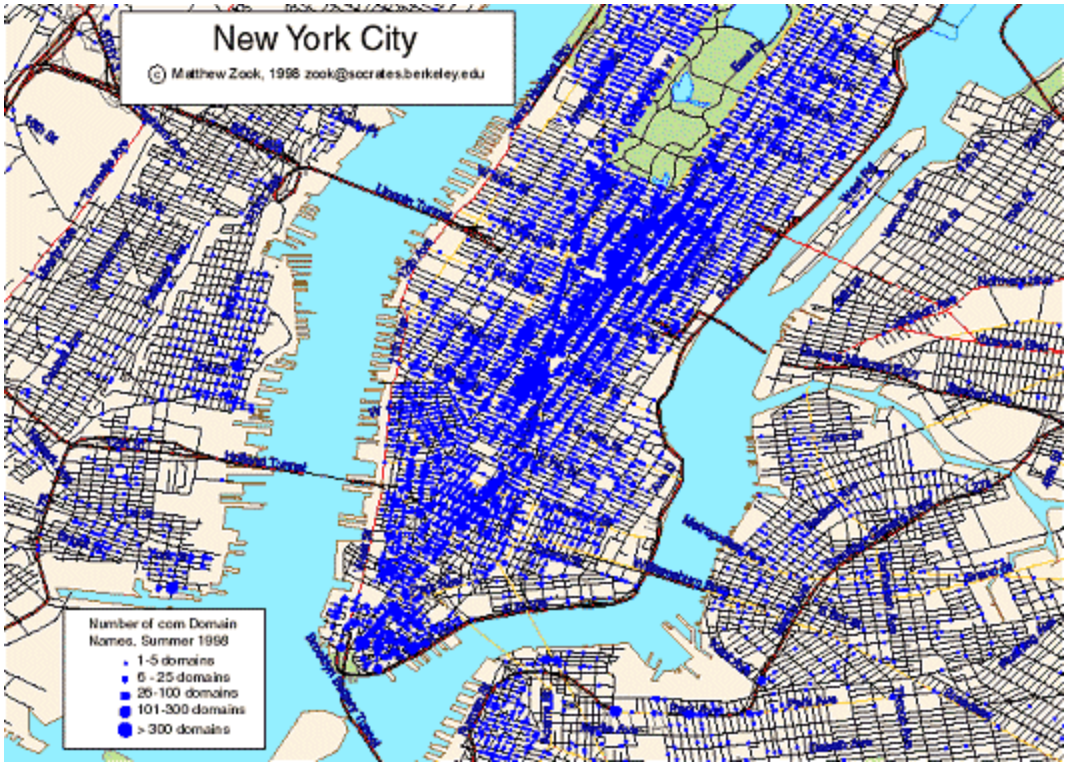


Figure 2.5: Zook image of New York domains [25]

2.2 Geographical Location Limitations

Mapping an IP address to a geographical location bears a strong resemblance to that of tracing a telephone call. Both telephone lines and computers are represented by unique addresses. Both telephones and computers communicate by transferring data over a global network. It is the differences between them, however, that expose why a geographical trace of an IP address is an exceedingly difficult task to accomplish with much certainty. The regulation, architecture, and the maintenance of the Internet all pose problems that limit the geographical ties of the Internet.

Internet Regulation

Many established communication networks, such as the Postal Service or Telephone Service, accommodate for some form of ability to determine the origins, or 'trace', a communication over their network. On a First-Class piece of mail, the postmark will reveal the zip code and city from which the piece of mail was delivered. Many telecommunications networks allow customers to trace calls they have received dialing '*57' [22]. This service is offered in order to protect customers from repeated harassing or offensive calls. Offering these services is a

trivial task for the networks to offer. Despite tense movie depictions of telephone calls where 'evil-doers' are held on the line by a task force of experts and computational power in order to determine his location, a telephone call trace is simply a matter of referencing a customer database.

Ten-Digit Phone Number:

(AAA) BBB – CCCC

A: Area Code	(City, State, County)
B: Prefix	(Local Switch, Neighborhood)
C: Telephone Line	(Unique Line Code)

Figure 2.6: Standard 10-Digit Telephone Number[22]

At first glance it is relatively easy to determine the general location of a telephone lines, given its address (telephone number). The assignment of telephone numbers is a process that is not only carefully regulated by the telephone companies and the Federal Communication Commission, but it is dependent upon the geography of the telephone line. Developed in 1943, AT&T set a 10 digit standard for telephone numbers in the United States. A unique first three digits, or 'Area Code', are designated to all telephone lines in a large geographical region, such as a small state, county, or a large city. The following three digits, or prefix, are unique to all of the telephone lines that connect to a switch at the local phone company. Although less regulated, these often map to a more specific geographical location. The final four digits are unique to each phone line and have no geographical binding [22].

IP Address:

255.255.255.255
A B

A: Network Address Octets
B: Computer Address Octets

Figure 2.7: Format of 32-Bit Internet Identifiers [1].

IP addresses, 32 bit identifiers that uniquely represent every computer on the Internet, look very similar to telephone numbers. Represented as a series of four octets, such as 130.111.112.21 (University of Maine, Computer Science Department), they are read in terms of significance from left to right, with the first two octets representing the network that it is on, and the last two octets representing the computer[1]. Unlike telephone numbers, there addresses are assigned in a fashion that is completely independent of geography. Regulated and maintained by the Internet Assigned Number Authority (IANA), the IP address space is allocated in blocks to large networks and organizations. IANA also has a standard for which networks and ISPs should distribute IP addresses to computers. These sets of standards, available at iana.org, offer no reference to geographical standards for IP distribution [10]. The highest-level classification is not regionally distributed, but instead it is distributed with

respect to networks and logical network subdivisions. Although there is a physical tendency for networks to be clustered geographically, this could not e the basis fro an accurate IP location methodology.

Of course, an overwhelming reason why the Internet is not regulated to geographical standards is that it is truly a worldwide web. Through government legislation, the FCC can keep the telephone companies to strict standards for how they operate the telephone networks. As soon as one places an international call, the 10-digit scheme for telephone numbers no longer applies, and tracing is certainly a non-trivial task. As far as the Internet is concerned in the United States, the FCC is of little impact. This suggests another reason why an IP geographical trace is difficult. The service of maintaining telephone lines and connecting calls is heavily monitored and documented by the telephone companies. There exists information at every telephone company as to the location of every telephone line, the billing address of every telephone company, not to mention the geographical schematic of the entire telephone network.

These standards and regulations imposed upon the Internet prevent any information available regarding geographical location from being considered generally reliable or comprehensive.

Packet-Switching Network

Transcending aforementioned differences is the fact that the telephone network and Internet achieve data communication using separate technology. Although the infrastructures for data communication is similar in that they are all linked networks, the Internet is a packet-switched network, and the telephone network is a circuit switching network. When two telephone lines communicate they create a circuit, allowing for information to freely travel from line to line along the same path. In a packet switching network, data travels in packets across the network, and is reconstructed at the client computer. This allows networks to function more efficiently, and communicate with multiple computers at once. This process does not imply, by any means, that the packets will travel the same network route to their destination. In the event that the most efficient path for a data packet to travel is experiencing high traffic, the packet will be routed to an alternate path. This characteristic of the Internet makes it a more malleable and dynamic environment. Each step that a data packet makes on the Internet might bring it across the room or across the country. Each path that a data packet takes may bring it to its destination in five steps or in twenty, through New York, or through Los Angeles, or both. With each step that a data-packet takes on an unidentified network, is a degree of anonymity that could leave the target computer anywhere.

Internet Service Providers

The architecture and regulation of the Internet clearly do not offer geographical representation; this makes it very difficult to form any solution. There exists, however, an issue that makes the task nearly impossible. The issue is that the majority of web content that makes up the Internet is maintained, not on an individual's personal computer, but on a large corporate web hosting company's server. Should one have the technology to trace web

content to the geographical position at which it is stored, it is often going to lead to the large server, located a non-deterministic distance away.

Many individuals gain access to the web through local ISPs. These 'last-mile' providers often provide limited web space to their customers. Cable and DSL providers will offer services to limited geographical zones. DSL services are limited in service to nearly an 18,000-foot radius, before the signal begins to terminally degrade [8]. Cable Internet providers offer service just as far as their cable network is available. Dial-Up providers can be reached as far as the telephone network reaches, and thus can be extremely remote to the provider. Although there is the tendency of users to access Dial-Up service at a local provider, certainly the web content can be maintained from anywhere in any of these situations.

Even more difficult to trace are web based hosting services such as Yahoo! Geocities. This service allows users to open a free account from anywhere on the web, and maintain their web page from anywhere they can obtain Internet access. In these cases there is a level of responsibility on the part of the hosts to ensure that the content available on their machines is not potentially dangerous. Members of Yahoo! Geocities, for example must accept 'Terms of Service' agreement that prohibits the use of dangerous content [24].

If most of the web content is being stored in locations remote from where they are maintained, and web content may be maintained from any location, the problem of geographically locating web content is not simply a matter of how it is accomplished, but a matter of determining the benefits of the results.

2.3 Scope of the Problem

This remote control of Internet access does not impact the methods of locating the servers that harbor the content. It does, however, provoke several questions regarding the question of geographically locating web content:

> **What is the location that is in question?** Trying to determine the physical location of data, and trying to determine the location from which the data is maintained are very different problems. Not only is there no established link between the two locations, but also the latter problem is essentially impossible to determine. Conversely, the physical location of potentially dangerous web content does not necessarily pose a threat to the surrounding area if maintained remotely.
>
> **What is the impact of Cyber-Geography?** Matthew Zook has done extensive research examining the correlations between Internet Geography and economic growth [25]. Do such applications apply to Internet Geography and terrorism? If the precise location of an IP address cannot be determined, is there merit to determining a region that it is in? Examining trends in the cyber geography of dangerous web content must be examined before value can be derived from locating the geographic region of an IP address.

What is an acceptable result of an IP location? If no result can be accepted with complete certainty, how is the location result to be presented? From global coordinates to time zone, all results are offered with uncertainty. In this situation, it is better to not present results as definitive answers, but as benchmarks for determining the potential location of the computer. Offering a definite location as a result would be misrepresentative.

The following is an investigation of geographically locating IP addresses. This implies that the resulting geographical information will refer to the physical location of the data – not the location from which the data is maintained. It is an investigation of methods that acquiring geographical data, with the intentions of determining trends in geography of dangerous content – not of precise computer location. It is an investigation in determining how to obtain a well-defined geographic region with which the IP address is located.

Chapter 3

Analysis of Problem

Chapter 2 established that there are no geographical ties between the Internet architecture and geography. Consequently, there are no methods or resources built to obtain geographical locations of computers on the Internet. Furthermore, all methods that are used for obtaining geographical data were not constructed to cater to these requests. For this reason, it is especially important to examine these methods not only for their applications in IP location, but to examine the nature of their functionality, as they were intended.

3.1 Traceroute

Traceroute is a utility designed for network analysis and debugging. This utility is standard on many machines, including all UNIX and Windows based systems. Originally developed as a debugging tool, traceroute was used by network administrators to determine the link in the network that was malfunctioning. The resulting data revealed the number of hops that the data made and which computers it hopped from in order to reach its destination.

As the Internet handles data communication using packet switching, a trace is not limited necessarily to any one distinct path from computer to computer. One issue that is associated with packet switching is the habit of packets to fall into network loops, where they continue to circle amongst the network without reaching the destination. Running a traceroute easily identifies this problem. Moreover, the solution to the network loop is the basis for which traceroute is achieved.

Each 'Data-Packet' contains a TTL (time-to-live) field. As the packet passes through routers en route to its destination, the TTL field is decremented. Should a router receive a packet with an expired TTL, it returns a 'Time Expired' error back to the origin computer. A traceroute is accomplished by distributing packets to the destination computer of incrementing TTL fields. With each packet returned is the IP address of the router where it expired. The resulting list of IP addresses represents the route with which the data packets traveled [7].

Figure 3.1 is an example of a traceroute. It reveals the step number of each hop, its IP address, the elapsed time required, and the computer host name, when available.

```
C:\>tracert umcs.maine.edu

Tracing route to umcs.maine.edu [130.111.112.21]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.0.1
  2    16 ms    16 ms    16 ms  142.167.3.11
  3    17 ms    17 ms    17 ms  142.167.4.2
  4    17 ms    16 ms    17 ms  bgr-cr01-f0-0-0.prexar.com [142.167.3.1]
  5    29 ms    29 ms    29 ms  500.Serial3-11.GW8.BOS1.ALTER.NET [63.111.121.197]
  6    28 ms    32 ms    28 ms  196.ATM2-0.XR2.BOS1.ALTER.NET [152.63.25.126]
  7    36 ms    33 ms    32 ms  190.ATM7-0.GW6.BOS1.ALTER.NET [152.63.21.237]
  8    34 ms    32 ms    32 ms  uofmaine.customer.ALTER.NET [157.130.14.146]
  9    31 ms    32 ms    33 ms  GW-P-C65-int.unet.maine.edu [130.111.2.33]
 10    42 ms    37 ms    36 ms  ATMPOR-9003.unet.maine.edu [130.111.33.69]
 11    45 ms    49 ms    40 ms  GW-O-C65-int.unet.maine.edu [130.111.33.6]
 12    41 ms    52 ms    56 ms  130.111.112.21

Trace complete.
```

Figure 3.1: Traceroute of UMaine.

Heretofore, applications of traceroute have not been mentioned. They may be used to discover network loops, diagnose malfunctioning network links, and analyze data flow on the Internet. One may select Internet service providers upon which service provide the most local access to a backbone provider.

As discussed earlier, the architecture of the Internet is independent of geographical ties. Although the trace in Figure 3.1 was made from several miles away and was contained within New England, it could have as easily occurred all within the same room or across the globe. The geographical information available from executing a traceroute is not in the process itself, but in the information embedded in the computer host names that are populated. Consider two sequential routers from the trace in Figure 3.2:

```
  4    17 ms    16 ms    17 ms  bgr-cr01-f0-0-0.prexar.com [142.167.3.1]
  5    29 ms    29 ms    29 ms  500.Serial3-11.GW8.BOS1.ALTER.NET [63.111.121.197]
```

Figure 3.2: Location Codes

By observing the top and secondary domains of each of the computer host names, prexar.com and ALTER.NET, respectively, one can observe that the data packet switched networks. If you look closer at the text that makes up the host names, you can extract the three character codes, ‘bgr’ and ‘BOS’. These codes refer to cities, and represent the geographical movement of the data packet from *Bangor, Maine* to *Boston, Massachusetts*. If you consider a separate traceroute, such as Figure 3.3, you can nearly determine the entire route of the data packet [6].

Based on the route listed, one can certainly conclude that the ICANN web server is located in the greater Los Angeles area, and traveled through several major American cities on its way there. Hunting for codes of data, like common city abbreviations, or airport ‘codes’ (In Figure 3.3, the code ‘lga’ certainly refers to La Guardia Airport, located in New York) the geographical route taken by the data can be determined.

```
C:\>tracert www.icann.org

Tracing route to www.icann.org [192.0.34.163]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.0.1
  2    17 ms    17 ms    17 ms  142.167.3.11
  3    18 ms    16 ms    17 ms  142.167.4.1
  4    35 ms    30 ms    26 ms  500.Serial3-11.GW8.BOS1.ALTER.NET [63.111.121.197]
  5    26 ms    26 ms    25 ms  196.ATM2-0.XR1.BOS1.ALTER.NET [152.63.25.122]
  6    35 ms    32 ms    32 ms  191.at-1-0-0.XR1.NYC1.ALTER.NET [152.63.27.110]
  7    36 ms    32 ms    33 ms  0.so-1-1-0.XL1.NYC1.ALTER.NET [152.63.19.86]
  8    34 ms    38 ms    34 ms  0.so-5-3-0.XL1.NYC8.ALTER.NET [152.63.1.49]
  9    39 ms    33 ms    33 ms  0.so-3-0-0.XR1.NYC8.ALTER.NET [152.63.19.30]
 10    35 ms    35 ms    39 ms  183.ATM4-0.BR1.NYC8.ALTER.NET [152.63.23.73]
 11    38 ms    42 ms    35 ms  200.atm6-0.pr1.lga2.us.mfnx.net [208.184.231.245]
 12    40 ms    36 ms    36 ms  so-2-2-0.cr2.lga2.us.mfnx.net [216.200.127.169]
 13    38 ms    36 ms    36 ms  so-0-0-0.cr2.lga1.us.mfnx.net [208.184.232.197]
 14    43 ms    43 ms    43 ms  so-1-0-0.cr2.iad1.us.mfnx.net [208.184.233.65]
 15    40 ms    39 ms    39 ms  so-1-0-0.cr2.dca2.us.mfnx.net [208.184.233.129]
 16   101 ms   103 ms   103 ms  so-5-3-0.mpr4.sjc2.us.mfnx.net [208.184.232.101]
 17   103 ms   105 ms   104 ms  pos8-0.mpr1.sjc2.us.mfnx.net [208.184.102.201]
 18   115 ms   118 ms   126 ms  pos0-0.mpr2.lax2.us.mfnx.net [208.185.156.126]
 19   121 ms   122 ms   114 ms  pos11-0-0.mpr1.lax1.us.mfnx.net [208.185.156.9]
 20   117 ms   116 ms   119 ms  208.184.95.130.mdr-icann.zoo.icann.org [208.184.95.130]
 21   126 ms   116 ms   116 ms  www.icann.org [192.0.34.163]

Trace complete.

Route:  Boston, MA → New York City, NY → Washington DC → San Jose, CA → Los Angeles, CA
```

Figure 3.3: Location Traceroute of ICANN

This algorithm for determining the geographical route of the data is sketchy, at best. It requires knowledge of typical city and country abbreviations and character representations. It requires the ability to observe various patterns of host names, which change with every different provider and network the data passes through. Often networks will not provide geographical information, creating gaps in the geographical trace. These issues pose difficult problems for computers to analyze.

Sarang Gupta overcomes this problem by using pattern matching of a city abbreviation knowledge base with a knowledge base of the regular expression representation of the computer host names of many networks and providers [18]. Consider the two major networks in the trace in Figure 3.4, ALTER.NET and MFNX.NET. Both of the networks use uniform computer host names and contain geographical information.

```
/\.([a-z]+)\d*\.us\.mfnx\.net$/i      MFNX.NET
/\.([a-z]{3})\d*\.alter\.net$/i       ALTER.NET
```

Figure 3.4: Perl Regular Expressions for Network Hostnames [18].

Figure 3.4 demonstrates two network patterns represented as PERL Regular Expressions. Extracted from the host names is a city code. With a database lookup mapping city codes to cities, the city may be determined.

Of course this process comes with a series of problems:

Multiple City Codes – Clearly, as each network consists of a different pattern, cities may not be represented in the same fashion. As seen above, New York City may be represented as 'NYC' or 'lga'. Other networks may refer to it simply as 'newyork'.

Ambiguous Codes – Clearly, if New York is represented as 'newyork' rather than 'NYC' there exists some doubts as to whether it is referring to the city or the state. Another classic example is 'Portland'. Determining whether this refers to the city in Oregon or the city in Maine may prove to be a challenge. As it turns out, Portland, Oregon is commonly referred by its 'airport code', 'PXE'. Additionally, such an issue can be resolved by examining the context of the search. Data is more likely to travel from Seattle Directly to Portland, Oregon, than it is to Portland Maine.

Non-Geographical Host Names – Not all networks and providers base the naming scheme to their computer host names on local cities, and geographical location. While often this is not a problem regarding the tracing of a route, it does make the geographical pinpointing of the destination computer a challenge. As it turns out, about 99% of the Internet traffic flows through about 1% of the networks that make up the Internet [1]. These few companies offer geographical codes, but as the data approaches the 'last mile' ISPs, geographical codes become less reliable and the possibilities of having the regular expression for pattern matching decreases.

This traceroute methodology is the basis for several of the contemporary systems. VisualRoute, SarangeWorld, and NetGeo all use a form of this technology. It is widely used as it depicts an accurate geographical trend toward the destination. This does not, however, guarantee that an accurate answer can be achieved. The geographical data revealed in a traceroute analysis can be considered 'route-based', as it reveals the path that the data traveled. This investigation is interested in obtaining information that is 'destination-based' and pertaining to the location of the target computer. The best 'destination-based' information that is obtained from traceroute is an approximation of the closest large city to the target computer.

3.2 WHOIS Database Approach

Long before the World Wide Web, there existed the WHOIS utility service. This service was used to provide network users with a way of obtaining the names, email addresses, postal addresses, and information of the like regarding other members on the network. With the rise of the World Wide Web, the WHOIS Service became an indispensable source for users to obtain network information and administrative and technical contact information for all Internet domains and IP numbers [23]. Figure 3.5 is a segment of a WHOIS database record, retrieved in regards to the University of Maine's Web Server:

```
Domain Name: UMAINE.EDU

Registrant:
  University of Maine
  UNET Computing
  Center University of Maine
  Orono, ME 04469
  UNITED STATES

Contacts:

  Administrative Contact:
  Irelann Kerry Anderson
  University of Maine System
  UNET Computing Center
  University of Maine
  Orono, ME 04469
  UNITED STATES
  (207) 581-3508
  ika@maine.edu


  Technical Contact:
  Same as above

Name Servers:
  NAMEO.UNET.MAINE.EDU 130.111.32.11
  NAMEP.UNET.MAINE.EDU  130.111.130.7

Domain record activated:   12-Sep-1997
Domain record last updated: 03-Aug-1999
```

Figure 3.5: WHOIS Entry for UMaine.

As the Internet continued to grow, the Natoinal Science Foundation (NSF) and IANA spawned InterNIC in 1992. InterNIC was established to organize and maintain the Domain Name Server, as well as the WHOIS database. Network Solutions (now owned by VeriSign) existed as a government endorsed monopoly over domain name registration. During this era, the accuracy and integrity of the WHOIS database, which was centralized at InterNIC was high. In 1998, the US Department of Commerce systematically deregulated the registration at Network Solutions in the interest of increasing competitive balance [23]. Simultaneously, the Internet Corporation of Assigned Names and Numbers (ICANN) was convened, and developed the Shared Registry System, by setting the standards for accrediting Internet Domain Registrars [11].

Although this has increased competition, it has significantly complicated the WHOIS service, and compromised its accuracy and integrity. Now the InterNIC WHOIS service, does not containing the WHOIS entries of all the domains, but rather a pointer for every domain to the Registrar that maintains the WHOIS database entry. Moreover, the service only offers access to the top level domains (TLD) of .com, .net, and .org. Recently, there has been a surge of new TLD's, such as country codes, and personalized (.info, .biz). All of the WHOIS entries for such domains are maintained on there own servers.

```
Domain Name: INTERNIC.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS.APNIC.NET
Name Server: NS.RIPE.NET
Name Server: NS1.CRSNIC.NET
Name Server: SVC00.APNIC.NET
Name Server: NS2.NSIREGISTRY.NET
Name Server: NS.ICANN.ORG
Status: REGISTRY-LOCK
Status: REGISTRAR-LOCK
Updated Date: 26-nov-2002
Creation Date: 06-jun-1997
Expiration Date: 10-nov-2012
```

Figure 3.6: InterNIC Shared Registry System Entry

The importance of the WHOIS database cannot be undersold. As the service exists as one of the few regulations on the overwhelmingly unregulated Internet, it is utilized for many purposes. Routinely, the service is used as a method for the investigation and selection of acquiring an unused domain name. More importantly, it is the only source for reliably contacting the proprietors of the domain, or the network. In the event that there exists a network problem, the technical assistance may be contacted. In the event that a hacker or spammer has been identified as operating from a particular network, administration may be contacted to investigate the problem.

Needless to say, many efforts have been made to circumvent revealing such contact or network information. Additionally, measures are being taken to reduce such efforts and minimize error within the database. Consider for a moment a WHOIS lookup on microsoft.com by a WHOIS telnet on port 43. The service will report not only domain name listings, but name server listings as well; the results contain a list of, often offensive, name server entries beginning 'microsoft.com'.

Faulty data also has become an increasing problem with the WHOIS database. Distinguishing intentional for typographical error can be trying. In May of 2002, a subcommittee of the House of Representatives was convened to examine the accuracy and integrity of the WHOIS database. During these proceedings, the following information regarding the inaccuracies of the WHOIS database was revealed:

Random Sampling Within .biz and .info TLDs [3]:

- Percentage with identical digits in phone number field: 4.7%
- Percentage with identical digits in ZIP code field: 6.5%
- Percentage with invalid ZIP code: 4.0%
- Average percentage with mismatched cities or states to ZIP codes: 7.0%

If this information were representative of the general TLDs (.com, .net, .org) over 800,000 entries would be considered incorrect [3].

Heretofore, the geographical implications of the WHOIS database have not been mentioned. It is clear, however, that the WHOIS database entries contain substantial information that directly or indirectly suggests a geographical location:

- City, State of Administrative & Technical Contacts
- ZIP Codes of Administrative & Technical Contacts
- Area Codes and Phone Numbers of Administrative & Technical Contact (Excluding Toll Free)

With simple parsing of a WHOIS database entry and access to the appropriate additional databases (ZIP code, Area Code); one may easily determine the geographical location. Unfortunately, the geographical location does not necessarily suggest the location of the destination computer. This geographical location merely represents the location at which an administrative or technical representative of the domain may be reached. Although this information does tend to suggest the general location of the destination computer, it by no means guarantees it.

Consider, for example, the most expansive backbone Internet provider worldwide, UUNET IP Network provided by WorldCom. They provide connectivity for over 100 countries, and offer over 3,800 point of presence locations to access the Internet [1]. Working under the domain name ALTER.NET, if you were to make a WHOIS database query you would receive the results in Figure 3.7:

```
Registrant:
WorldCom, Inc. (ALTER-DOM)
  2400 North Glenville
  Richardson, TX 75082
  US

  Domain Name: ALTER.NET

  Administrative Contact, Technical Contact:
     WorldCom, Inc.  (FSQHFFBMOO)                hostmaster@wcom.com
     WorldCom, Inc.
     2400 North Glenville
     Richardson, TX 75082
     US
     972-729-5738 fax: 123 123 1234

  Record expires on 27-Jul-2010.
  Record created on 26-Jul-1990.
  Database last updated on 11-Mar-2003 13:01:05 EST.

  Domain servers in listed order:

  AUTH00.NS.UU.NET         198.6.1.65
  AUTH60.NS.UU.NET         198.6.1.181
  AUTH200.NS.UU.NET        195.129.12.82
  AUTH210.NS.UU.NET        195.129.12.74
```

Figure 3.7: WHOIS look-up for ALTER.NET

Clearly, the geographical data provided in the WHOIS database search is not representative of the expansiveness of the network. Moreover, it is these expansive networks that route the

majority of the data packets around the global network. Unlike traceroute, the WHOIS database would not be a god tool for determining the route that data travels over the Internet.

That being said, many networks are not worldwide, especially the networks that commercially offer Internet access residentially. Many Internet service providers are owned and operated locally to their networks. Although any street address extracted from the WHOIS database is much too much specific to be valid, the ZIP codes and area codes give a general idea of the geographical location of the network and the destination computer. Unfortunately, it is not easy to determine the geographical size of a network, based on its WHOIS entry.

3.3 Other Approaches

Fundamentally, the Traceroute utility and the WHOIS database are the most useful tools for discovering geographical data, if for no other reason, because they capable of producing some form of result for every computer connected to the Internet. There does exist, however, several other methods and proposals for obtaining geographical information.

DNS LOC

Proposed in January of 1996, DNS LOC is a mechanism for integrating location information into the current Domain Name System [5]. The Domain Name System is a distributed, hierarchical database by which host names are resolved to IP addresses. Additionally, every DNS entry contains a series of resource records. These records contain assorted information associated with the host or network of the request. LOC currently is operating as one of the resource records in an experimental stage. The specification for the Resource is documented in RFC 1876, available at the IANA web page [10].

The data contained with in each location resource is composed of seven different fields to express the geographical location [4]:

> **Version** – This 8-Bit field simply contains the version number of the data representation. It must be assigned to zero.
>
> **Size** – 8-bit field represents the diameter of the sphere enclosing the entire network entity. Using centimeters as its unit base, it can express from less than a centimeter to 90,000km.
>
> **Precision** – Horizontal and Vertical precision compose the next two field is expressed in the same manner as the size, and represents the potential error in either the Horizontal of Vertical measurements.
>
> **Latitude/Longitude** – Expressed as a 32-bit integer, each field is a measurement of thousandths of seconds of an arc. In the case of latitude the equator is represented by 2^{31}; for longitude the prime meridian is represented by 2^{31}.

Altitude – Expressed as a 32-bit integer, the altitude is a measurement of centimeters from a base 10,000m below sea level.

The information provided in the DNS LOC resource record, and the distributed design of the Domain Name System is an excellent solution to the problem of geographically locating. It is not without its faults, however. Firstly, it is currently not widely supported by many computers or networks. Without support from a majority of network, the DNS LOC information paints an inaccurate image of the Internet. Secondly, the distributed nature allows for network administrators to maintain their own DNS LOC entry. This allows for inaccurate, imprecise, and faulty data to be submitted. Such fallacies would terminally compromise the accuracy and integrity of the DNS LOC system. Especially with regards to computer systems that contain terrorist or offensive content, that would largely like to maintain anonymity.

Traceroute & DNS LOC

Fully realized, DNS LOC would be an excellent resource to incorporate with the traceroute utility in order to achieve geographical location. While DNS LOC is still in an experimental phase, using pattern matching with host names is a more effective method of obtaining location data. DNS LOC information, however, is more specific, provides more geographically based information, and is standardized, requiring minimal system maintenance. Furthermore, using the trace information reduces the problems with DNS LOC, such as data integrity. Running one trace or multiple, the path of the data may be analyzed to confirm the accuracy of the geographic information at the target computer.

NetWorldMap

Discussed in the Chapter 2.1, NetWorldMap is another effort made to map the entire Internet to a geographical location [17]. This Map is being compiled by collecting seed data from willing web surfers who volunteer their location and IP address on the NetWorldMap web page. The location of other IP addresses is determined by using all of this seed data to lookup the geographical location of computers, and various algorithmic processes to determine the location of unknown computers. The validity of the seed data is based on collecting a very large sample of accurate, geographically and network-diverse IP locations from willing contributors. This process is a large a large undertaking, with potentially ineffectual results.

Daytime Server

Although not offered on many servers, if you attempt a socket connection with Port 13, you make a request to the Daytime server, which returns the current time of the system. By comparing local time with the server time, one can determine the time zone with in which the computer resides [6]. There are several problems that exist with this form of geographical location. First and foremost, the results are of a granularity of a time zone. This form of information is useful for confirming other, more specific data obtained, and that is about all. Moreover, the daytime server is not running on the majority of servers

these days, nor does the system time have to be accurate to the time zone, should the administrators not want it revealed.

Chapter 4

Design of Solution

In Chapter 3, both the strength and weaknesses of the traceroute and WHOIS approaches to IP location were discussed. Although both approaches have merit, they both contain characteristics that prevent them from being able to stand on their own as formidable solutions to the problem. Traceroute is fundamentally 'route-based' instead of 'destination-based' and host-name matching will fail on all small networks that do need-not contain location data. The WHOIS database contains geographical data that does not pertain to the location of the computer, but the address of administrative or technical support, and therefore it is largely inaccurate on larger networks.

Many IP location projects, such as VisualRoute and NetGeo, have recognized that the weaknesses of each approach are remedied by the strengths of the other, and made it the basis for the design of their solution. The distributed IP locator enhances this design by executing it in a geographical distributed manner.

4.1 Single Traceroute Approach

Typically only geographically expansive networks have reason to include geographical location codes with in the host name of their IP devices. For this reason, traceroute and WHOIS are very compatible in locating Internet computers. When a data packet travels from the east to west coast on a nationwide backbone Internet provider, a WHOIS look-up will return the same address of the Network's Administrative office for the routers on both coasts. A traceroute, however, will identify that they exist on the same network, and extract two different location codes. Moreover, when the data packet reaches its destination on a local network 200 miles from the nearest located city, the traceroute may not extract a local code because it is not available, but the WHOIS database will return an address considerably more local to that of the city 200 miles away.

The preceding scenario is an example of how traceroute and the WHOIS database can be used to determine the location of a computer with more accuracy and certainty than either of them could do alone. Running both a traceroute and a WHOIS look-up is fairly trivial, there are, however, more complicated issues involved in determining when and which of the two processes are more accurate.

Traceroute Analysis

The list of hostnames populated by running a traceroute serves as the basis for the single traceroute approach. It is from the hostname that not only are the location codes extracted, but also the domain associated with an entry in the WHOIS database. Rather than examining each node for both location codes and WHOIS data, the traceroute data is first examined for location codes. Figure 4.1 demonstrates the analysis in a single traceroute. The basis for this decision was made for the following reasons:

- **Abuse of the WHOIS Database**: It is in the interest of the system to minimize the burden that it puts on the WHOIS database, as it is a resource used for many purposes and cannot handle extensive strain from a particular source. This is compounded by the fact that much of the traceroute data is useless to subject to a WHOIS search. It is important to note that only half of any traceroute is approaching the target computer; the other half is leaving the origin computer.

- **Backbone Network Travel**: An efficient network will route a data-packet to a backbone provider in as few steps as possible. For this reason, most Internet traffic takes place on Backbone providers with which there is not much beneficial WHOIS data available.

- **Effectiveness of Location Codes**: Although location codes are often not accurate to the exact town, in examining traceroute data it is a very effective procedure for determining the closest big city to the target computer. Obtaining this data in advance improves the effectiveness of a WHOIS database search.

Each of the host-names is examined for location codes and then linked to a location. The location-identified host-name closest the fewest number of network steps from the target, is declared the city closest to the target computer.

Linear Representation of a Traceroute

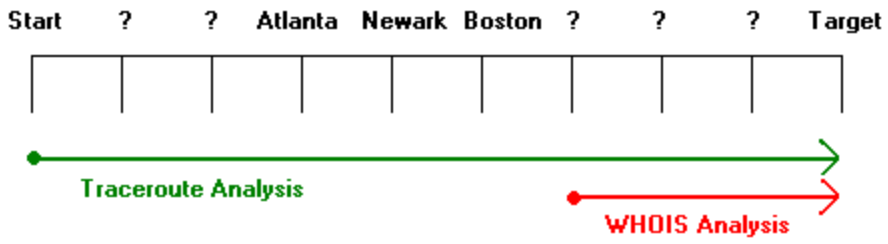


Figure 4.1: Linear Diagram of Single Traceroute Analysis

Following host-name analysis, the hostnames of the network steps between the closest identified city and the target are queried on the WHOIS database. This information collected from the WHOIS database is used for two purposes:

- **Support the accuracy of the 'Closest City'**: The network steps made between the proposed closest city and the target computer reveal information about the accuracy of the closest possible city, even though they cannot be located. If there are more than a few steps, or the steps are made on expansive networks, it decreases the certainty that the proposed 'Closest City' is indeed local.

- **Refine the location of the 'Closest City'**: In the event that the network steps between the proposed closest city and target computer support its accuracy, then the WHOIS data can also be used to refine that location. Should the WHOIS database data contain a geographical location local to the proposed 'Closest City', such as a suburb, this new city is a more refined, granular assessment of the target's location.

The preceding location analysis is the final step in the single traceroute approach, following the WHOIS database lookups.

Limitations of the Single Traceroute Approach

The single traceroute approach is certainly an effective method of using traceroute, host-name analysis, and the WHOIS database to determine a location with more certainty than they could alone. It is not, however, without its limitations. First, it does not address the issue of how traceroute is 'route-based' and not 'destination-based'. Second, As the Internet is a packet-switched network; it does not take advantage of the fact that there exist several different network paths en route to the target computer. Different networks can only increase the amount of information available regarding the target location. Finally, by executing a single trace from a single location, the results are geographically biased and insufficient. If the traceroute is only approaching from the north, the data is unable to determine a reasonable southern boundary with which the target computer could exist.

4.2 Distributed Traceroute Approach

Several of the limitations in the Single Traceroute Approach, as documented in Chapter 4.1, are easily remedied if traceroute data on the same target computer could be collected from a number of geographically and network-diverse locations. Integrating the resulting data would provide a considerably more accurate and certain location of the target computer. By using multiple traceroutes, more network paths are explored, and geographical limitations are refined. Figure 4.2 demonstrates how multiple traces confirm the location of the target computer.

Many networking and technology companies, as well as educational institutions offer public traceroute access from their servers. A distributed traceroute approach was proposed to access to a geographically diverse selection of these servers [26]. Fundamentally, the process still is based on the Single Traceroute Approach. The only difference in the process is that the collection of trace data is delegated to other machines before they are processed, and that the results of each trace are compiled individually and then analyzed together.

Distributed Vs. Single Traceroute Approach

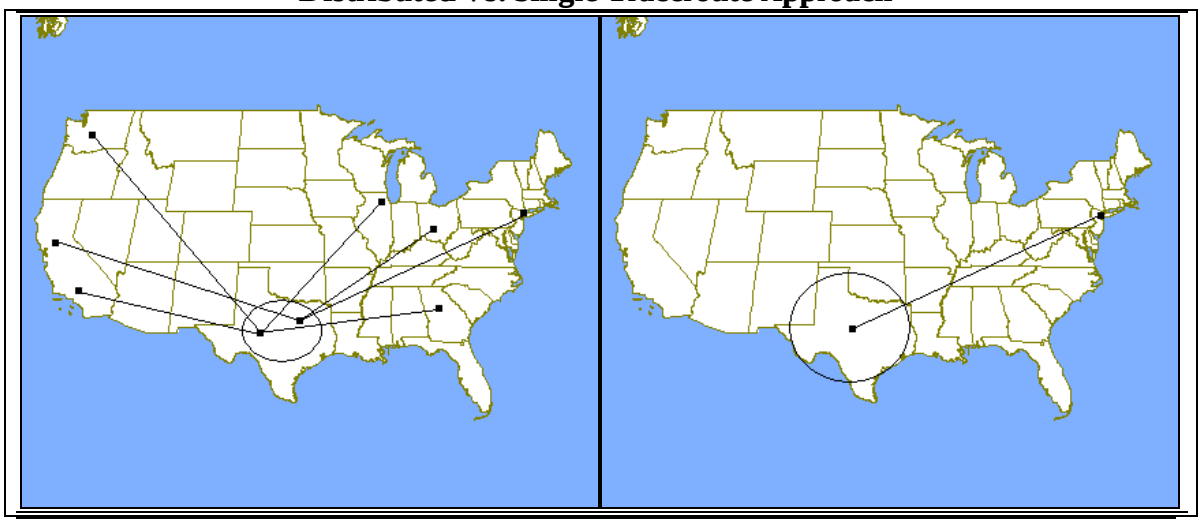


Figure 4.2: Distributed Vs. Single Traceroute Approach.
Map Courtesy of TIGER [20].

Analysis of Distributed Traceroute Data

The design of the distributed traceroute approach is simply an extension of the single traceroute approach, as displayed in Figure 4.2. The heart of the distributed approach is in the results analysis following the execution of each trace. Analysis consists of two steps: Consolidating the trace data, and integrating the trace data.

- **Consolidating Data**: This stage examines each trace, removing them when they are unnecessary. Many of the traces take several different routes to achieve the same destination; there is a tendency, however, for traces to rendezvous at a unique node and proceed through the same cities on the same networks. In other cases, two traces will rendezvous in a city, and one trace will proceed to reveal a more precise location. In both situations the trace containing less data may be removed. Heuristics such as these reduce the trace data to only the most valid, and applicable data available. Consider Figure 4.3, two of the traces pass through Boston en route to the destination. Because one trace is able to identify another location (Portland) between Boston and the Destination, the other trace may be pruned.

- **Integration of Trace Data**: Following trace data consolidation, the traces are linked to create a 'Location Tree'. Each trace is examined for their most precise location. Like cities are linked together. Each identified city preceding the most precise locations are linked to these cities. The resulting tree is constructed of three tiers: Target computer, possible location cities, and the closest identified cities preceding the possible locations.

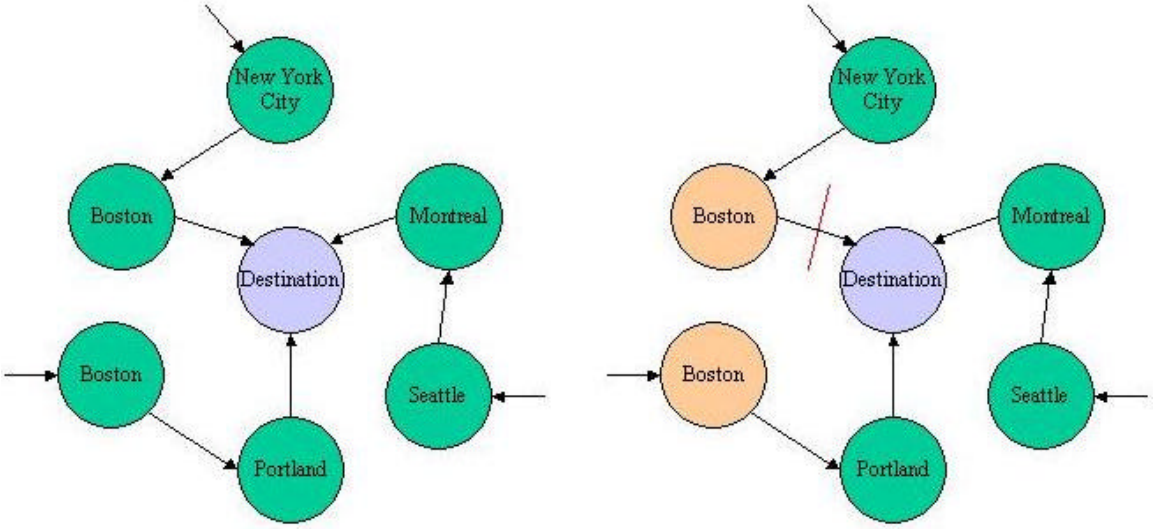


Figure 4.3: Pruning of Traces

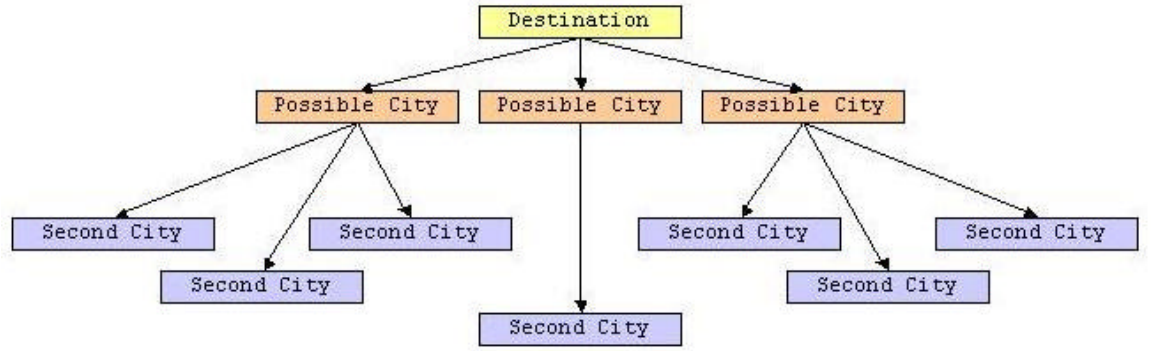


Figure 4.4: Location Tree

Distributed Approach Data Structures

The intent of the 'Location Tree' data structure is to examine the region in which the target computer is possibly located. Each tier of the tree, stepping back from the target node, is of descending relevance to its geographical location. Each step back, however, provides increased information regarding the region with which the target computer is located. Figure 4.4 demonstrates the three tier structure of the 'Location Tree'. Consider the following scenarios, outlining the geographical information available at each tier:

- **1st Tier:** Location codes available in the destination node that represents the hostname of the target computer offers as close of an accurate ID to its geographical location that can be achieved. Unfortunately, at the level of terminal or server computers, there are not often location codes within their hostname. Unless a router or networking computer is being traced, this will offer no geographical data. The rightmost image in Figure 4.5 demonstrates the precision of a single tier examination. The yellow circle represents the span of the local network, as revealed by the WHOIS database.

- **2nd Tier:** When location information is not available in the hostname of the destination computer, plotting the 'possible city' locations – contained in the second tire of the tree – will reveal a geographical cloud engulfing all 'possible cities' that represents the location of the target computer. The red triangle in the middle image of Figure 4.5 marks the bounds of the possible cities, creating a geographical cloud.

- **3rd Tier:** Refining the cloud, the closest identified cities preceding the 'possible cities' are plotted and joined by a straight line to their respective 'possible cities'. The resulting lines represent directional vectors; these vectors suggest the direction and locality of the target computer. A vector headed south, suggests the target is south of the possible city, or at least no further north than the city at the tail of the vector. Furthermore, a vector that spans a state suggests a much smaller range for geographical error than a vector that spans the country. The final image in Figure 4.5 creates the directional vectors. As displayed by these vectors, the geographical trend is south, suggesting the southern possible city.

Expanding the 'Location Tree' past three tiers begins to start providing increasingly irrelevant data, as it is impacted more by the geographical origin of the trace than of the geographical destination. Additional information regarding the Distributed Approach design is available in Appendix D.

Progression of Location Tree Analysis

Figure 4.5: Progression of Location Tree Analysis.
Map Courtesy of TIGER [20].

4.3 IP Locator Design

The design of the Distributed Traceroute Approach to IP location is decomposed into five separate modules. Two of the modules, specifically pertain to execution of the single traceroute approach, of which a thread is executed for each trace executed. Figure 4.6 may be used for reference regarding the interaction of the modules.

Execution Module

This module handles the initialization of the IP Location process. From this module, each single traceroute thread is spawned. Additionally, the execution module handles the interaction with all of the systems external resources. This includes collecting the target computer address, as well as the public traceroute servers. Other external resources include the database of network host-name patterns and location code translations.

Single Traceroute Module: Pattern Matching

The first component of the single traceroute analysis is executing the traceroute, matching the hostnames of each network step, and mapping the location codes to cities and locations. Both of these forms of information are contained within external resources.

Single Traceroute Module: WHOIS Analysis

WHOIS analysis involves two steps. First, the traceroute must be examined to determine the appropriate host-nodes with which to query the database. Following this step the WHOIS lookups may be executed.

Obtaining a WHOIS database record involves two steps since the Shared Registry System was imposed. Before any database entry can be obtained, the InterNIC WHOIS service is queried to determine the appropriate WHOIS database to query [19]. There is no standard format for the WHOIS database, so the WHOIS Analysis module must examine each entry and extract the appropriate data. Extracting the five-digit zip code and accessing a zip-code database, is a simple method of accomplishing this.

Distributed Analysis Module

Distributed analysis, as discussed in Chapter 4.2, consists of data consolidation and integration. It is in the Distributed Analysis Module where each single traceroute thread reports their results that are consolidated and integrated. The resulting data structure of these steps is the 'Location Tree' used to demonstrate the region in which the target computer may be located.

Output Module

This distributed traceroute design represents the back-end of an IP location system. The output module is the component of the system that acts as the back-end interface.

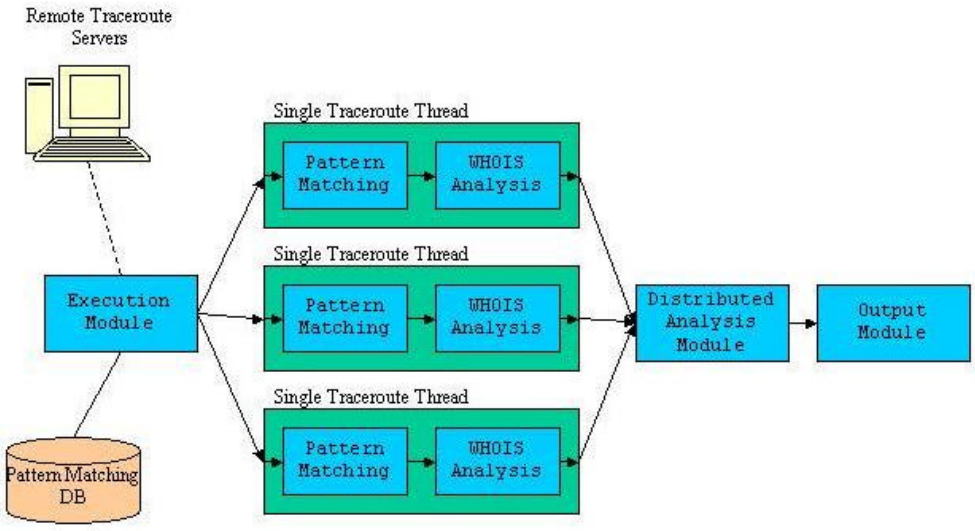


Figure 4.6: IP Locator Module Design

Chapter 5

Implementation

Accompanying the designed proposed in Chapter 4, is the following prototype of an IP Locator. A prototype of the IP Locator serves not only to demonstrate several aspects of the design of the system and its processes, but also as a fully functioning demonstration [27]. This functioning demonstration was achieved by pooling multiple available resources to produce a process that representative of the IP Locator. Moreover, it is able to produce results that not only demonstrate the capabilities of the design, but also shed light upon the limitations of IP location.

5.1 Basis for Implementation

The implementation of the IP Locator Prototype was approached with the intent to fulfill several objectives; consequently, there are aspects of the implementation that are developed and other aspects that are under developed. The following objectives served as the basis for the implementation:

- Demonstrate the design and architecture of the IP Locator.
- Implement the algorithm for traceroute analysis.
- Present the solution in a simple, understandable fashion.
- Produce results representative of the solution's effectiveness.

The resources integrated with the implementation were chosen based on achieving these objectives.

Perl

The choice to write the prototype in Perl was based on several qualities to the language that appeal to this particular project.

> **Scanning & Pattern Matching** – Perl, the 'Practical Extraction and Reporting Language', was developed with the intentions of handling string manipulation in a practical fashion [2]. This functionality presents an elegant solution to matching and extracting location codes from computer host names, in addition to the parsing of the host names from raw traceroute results [13].

Library & Available Resources – Not only does Perl offer an expansive standard library, but additionally, the popularity and open nature of its programming community has provided many additional modules and forms of documentation. Functionality such as web content retrieval from traceroute servers, and WHOIS database queries were handled, by such components [16].

Web Integration – Perl has been long used as the language of choice for writing CGI (Common Gateway Interface) scripts, as well as many other networking and World Wide Web solutions. This not only has increased the documentation of such procedures, but also it offers the possibility to easily create a web interface for the prototype.

Resources

The showcase of the implementation is the demonstration of the advantages of the distributed approach. When available other aspects of the prototype are handled by available resources.

Traceroute.org – This web page, maintained by Thomas Kernan, is an extensive list of trraceroute servers available to the public worldwide [12]. For this particular prototype, the only public traceroute servers supported, must use the Common Gateway Interface 'GET' protocol. Traceroute are achieved by requesting the accessing the host name appended to a '?' and the target domain or IP address.

LWP::SIMPLE – The web content desired from traceroutes, courtesy of traceroute.org are obtained by use of the 'get()' function contained within the LWP::SIMPLE module, standard with the Perl Library. The function offers simplified interface for obtaining web content. It is easy to use a simple, but unable to detect timeouts [2].

NET::WHOIS – A Perl module, written by Chip Salzenberg and Dana Hudes, is used to obtain all WHOIS database queries [16]. The module is used to interface specifically with the InterNIC WHOIS database, and therefore can only access the records to the '.com', '.net', '.edu', '.org' Top-Level-Domains.

SarangWorld Resources – Used as the knowledge basis for the prototype, Sarang Gupta has compiled a database of Perl regular expressions for known network containing geographical clues [18]. Additionally, he has compiled a database of translations for each of these codes to their geographical location.

5.2 Design Module Frame for Implementation

The design of the IP Locator Prototype is composed of 4 distinct stages, each of which map to the corresponding Module in the design.

Traceroute Execution / Data Population

The first step simply is simply an initialization stage. The host name regular expressions, code translations, and traceroute URLs are all populated from separate files into their respective data arrays. Additionally, the traceroute data from the distributed traceroute servers is collected. The traceroute data acquired is retrieved as HTML from a web server. With each call to the web server a traceroute script is executed and the results are posted in HTML.

Pattern Matching

This stage involves three steps. This stage takes raw trace data and returns an array of node locations and domains. If the location of any individual node cannot be determined, the location for the particular node is entered as '<unknown>, <un>'.

> **Extracting Host-Names** – Each traceroute request returns an HTML document as the raw data. The document is scanned for each string segment that matches the pattern of a hostname. Each match in sequence is pushed on an array; each array is referenced on an array of traces.
>
> **Matching Host-Name** -- Each host-name is matched against an array of host-name regular expressions. If a match is found, a location code is extracted from the host-name and the domain is extracted as well. Each node is stored as a location-code/domain pair:
>
> <locationcode> :: <domain>
>
> **Translation Location Codes** – The final step in pattern matching is looking up each location code in the array of code translations. In the array of location-code/domain pairs, the location code is replaced with the City and State of the particular code.
>
> <city>, <state> :: <domain>

Build Location Tree

Once the traces have been interpreted, location analysis begins.

> **Consolidate Searches** – Before each trace is analyzed, it is pruned of useless nodes. This includes nodes that are of unknown location at the origin of the trace Consecutive nodes of the same location and domain may be consolidated to one node.
>
> **Build Trace Data** – To build the Location Tree, each trace is examined for the following information:

- Destination Node: The final node discovered on the trace, the target computer.
- First Possible City: The closest located node, in terms of network steps, to the destination.
- Second Identified City: The second closest located node, in terms of network steps, to the target computer.
- The Local Network: All of the hops between the destination and the first possible city.

The preceding information is stored in an array based data structure, as displayed in Figure 5.1. For each trace, with the reference of the local network pointing to an array the local network hops.

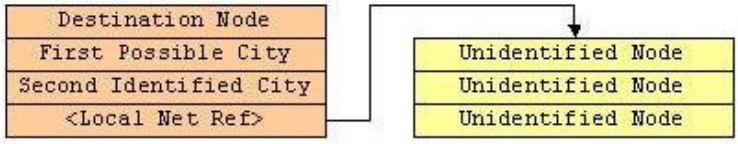


Figure 5.1: Trace Data Structure.

Build Location Tree – The Location Tree is composed by linking the trace data arrays constructed in the pervious step. The result, as demonstrated in Figure 5.2, is a three-tiered tree, linking 'First Possible Cities' to the 'Destination Node', and linking 'Second Identified Cities' to 'First Possible Cities'. Additionally, each possible city entry is associated with a 'Local Network' array.

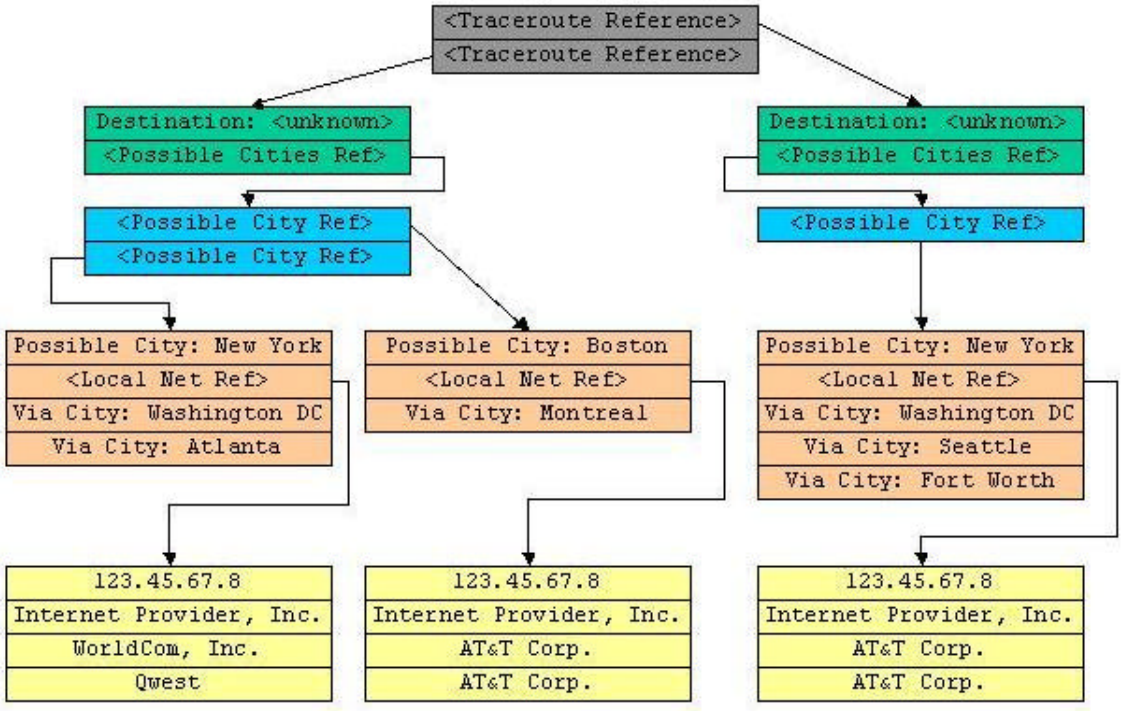


Figure 5.2: Location Tree Structure.

Output

For the purposes of this prototype, the output is simply formatted text results, displaying the location tree constructed for the target computer. This is accompanied by printout of the results of each trace executed. Information regarding the interpretation of these results is offered in Chapter 5.3.

5.3 Prototype Results

In most scenarios, running the prototype will fail in determining the location of the 'Destination Node'. Considering this failure, one is delegated to examine the location tree. Whereas the result of a single traceroute design is one specific result, and the distributed traceroute design is several possible locations, this does not imply that the distributed approach is less accurate. In fact, neither approach may be considered more accurate than the other, as the geographical location of any computer cannot be definitively determined. The distributed approach, however, is certainly no less accurate, and additionally provides more information regarding the certainty of the result. Figure 5.3 represents the output of prototype for a typical location query. In this case it is IEEE.org, located in Piscataway, NJ.

```
************************************************
---Destination:         <unknown>, <un>
---Network:             *
************************************************
**Possible City:        New York, NY
**Network:              bbnplanet.net
**Via:
**                      <unknown>, <un>
**                      Philadelphia, PA
**
**Intermediate Networks:
**                      ieee.org
**                      bbnplanet.net
************************************************
**Possible City:        Pennsauken, NJ
**Network:              sprintlink.net
**Via:
**                      Fort Worth, TX
**                      Newark, NJ
**
**Intermediate Networks:
**                      ieee.org
**                      sprintlink.net
************************************************
```

Figure 5.3: Prototype results for IEEE.org lookup.

Destination – The Destination field displays the discovered location of the target computer. This does not guarantee that there necessarily was a location code discovered in the host name. In this event, the field displays: '<unknown>, <un>'. It should be noted that, although under most circumstances the location tree should only contain one destination,

the location tree is capable of accommodating for multiple destination nodes. More often than not, this simply is a different entry for the same computer, or that particular traceroute failed.

Network – The 'Destination Network' is also displayed along with the 'Destination' field. This field displays either the name of the Network, if a WHOIS lookup was successful; or the secondary and top-level domain name is displayed.

Contact Address *(if available)* – In the event of a successful WHOIS database lookup, the prototype report will display a contact address of the network. Although often is the case that the network is national- or world-wide, if the address is local to the location of the destination or possible cities, it can confirm their accuracy.

Possible City – The possible city entry reveals the closest identifiable city, in terms of network hops. Depending on the destination and the diversity of traceroutes executed, a number of different 'possible cities' may be discovered by other traces. This is where the benefit of a distributed traceroute is revealed. Plotting each of the possible cities effectively surround the destination of the target machine. Additionally, as the number of traceroutes increase, erroneous data begins to reveal itself.

Network & Contact Address *(if available)* – Just as displayed with the destination, all possible cities are accompanied by the network information.

Via – This list of cities represent the 'Second Identified Cities' in the location tree. Although, at surface value, they appear to offer little help in addition to that of the possible cities there approach gives the possible cities more qualities of a vector, as discussed in Chapter 4.2. The 'Possible Cities' can, by no measure be assumed to be the location of the target computer. By examining the direction from which the data is arriving, a more accurate picture can be created of the possible location of the target computer.

Intermediate Networks – Associated with each possible city is a 'Local Network Array'. This array represents the network steps made between the possible city and the destination. If the location of the destination is unknown, each network step on the array is an additional degree of anonymity to which the location of the target computer is unknown. In the prototype output, the network and contact address, if available, are listed. This information can additionally reveal more information regarding the anonymity. A series of network steps on an international network, often suggest major geographic movement.

Raw Trace Data *(not displayed)* – Following the printout of the location tree is a printout of every successful traceroute printed as location/domain pairs, as described in Chapter 5.2. This data is simply to be used as reference regarding the construction of the location tree.

Chapter 6

Results & Conclusions

6.1 Prototype Analysis

Chapter 2.3 examined the limitations and scope of the IP Location problem, concluding that it is not possible to locate a computer with complete precision or certainty. This can make the analysis of any solution difficult. For the purposes of this project, the intent of the following results analysis is to demonstrate that executing the distributed approach provides better results than would be provided by any of the individual traceroutes that comprise it. It should be noted that the selection of web pages for analysis was based on their high visibility in the online world, and not because they are potentially harboring terrorist information. The trace results of each example are available in Appendix C.

Scenario #1: IEEE.org

Actual Location:	Piscataway, New Jersey
Location Tree Destination: Possibly Cities:	Unknown
Three Degrees Anonymous:	New York, NY (Philadelphia, PA) Pennsauken, NJ (Fort Worth, TX; Newark, NJ)

Analysis: This example contains two possible destinations of the target computer, each possibility being three steps away from the target computer. Although, neither one of these locations (New York, Pennsauken) is the exact destination, the composite of the two confirm a location in New York City/New Jersey. Philadelphia and Newark serve as more evidence to this conclusion, and by approaching the possible cities from the south (with Forth Worth) suggest it location is more local to the northern focal point (New York).

Accuracy: Piscataway is located 30 miles west-southwest of New York, and 51 miles Northeast of Pennsauken. Using each of the two possible cities as focal point in this example, the target zone could be reduced to between the two cities. Either of the traces standalone would offer no more information regarding the position of the target relative to the final possible city.

Plotted Results of IEEE.org Prototype Trace

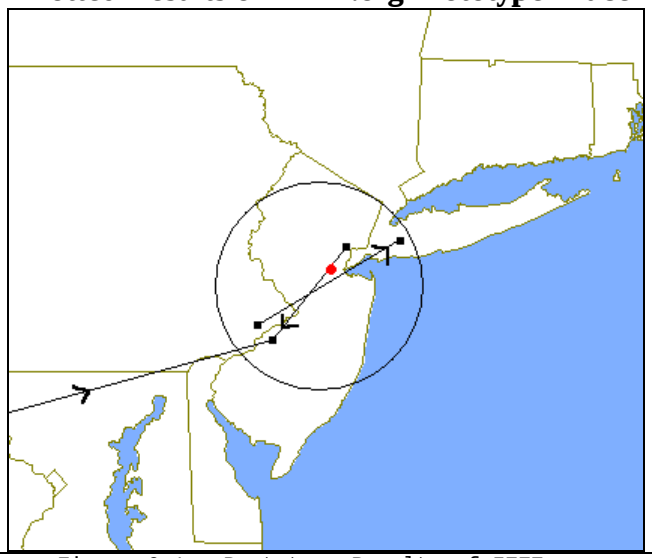


Figure 6.1: Prototype Results of IEEE.org
Maps Courtesy of TIGER [20].

Scenario #2: ICANN.org

Actual Location:	Marina Del Rey, CA
Location Tree Destination:	Unknown
Possibly Cities:	
Two Degrees Anonymous:	Los Angeles, CA (San Jose, CA; Milpitas, CA; Dallas, TX)

Analysis: With only one possible city, there is no means for confirming a zone in which the solution exists. There is sufficient data however, to confirm that the target computer exists in the Los Angeles area. Firstly, there is only two degrees of anonymity, suggesting that the computer only passed through another two machines before it reached its destination. Secondly, with three cities representing three separate approaches to Los Angeles from different directions, Los Angeles is repeatedly confirmed as the appropriate destination.

Accuracy: Marina Del Rey is located 9 miles north-northeast of Los Angeles. Because the only possible city is approached by three separate traces, one may conclude that the destination is strongly correlated to the particular city. With each additional approach the correlation grows stronger. In the case of a single trace, the correlation is weak.

Plotted Results of ICANN.org Prototype Trace

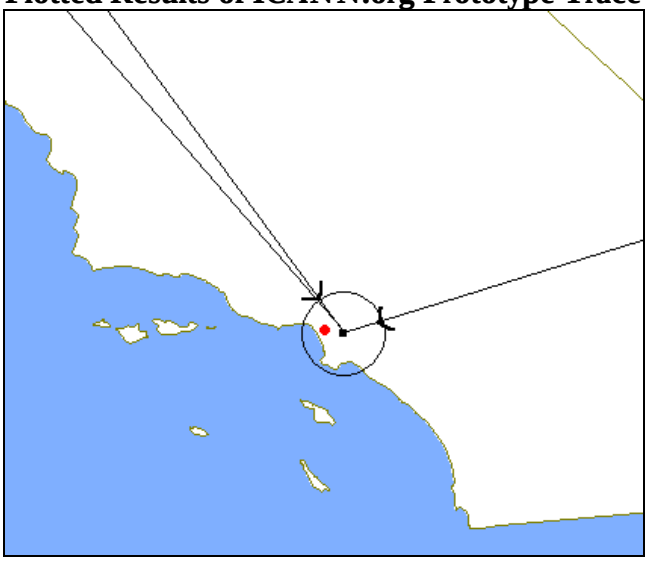


Figure 6.2: Prototype Results of ICANN.org
Maps Courtesy of TIGER [20].

Scenario #3: CNN.com

Actual Location: Atlanta, GA

Location Tree Destination: Unknown
Possibly Cities:

One Degree Anonymous: Atlanta, GA (Houston, TX; Charlotte, NC)

Analysis: With one possible destination, and only one degree of anonymity between that location and the target computer, evidence in this scenario is strong that the target computer does indeed exist in Atlanta. It should also be noted that Atlanta, which is located in the Southeast portion of the country, is approached by traces from the North and the West (Charlotte, Houston). This further confirms the evidence.

Accuracy: The only possible city correctly identifies the city if the target computer. Although every trace individually concluded upon Atlanta as the destination, each additional trace confirms that information, and confirms the general location by offering direction.

Plotted Results of CNN.com Prototype Trace

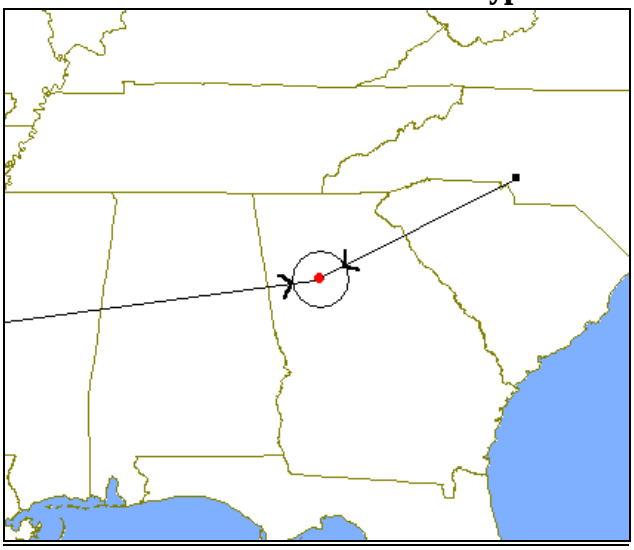


Figure 6.3: Prototype Results of CNN.com
Maps Courtesy of TIGER [20].

Scenario #4: Residential Computer

Actual Location: South Berwick, ME

Location Tree Destination: Unknown
Possibly Cities:

One Degree Anonymous: Dover, NH (Portsmouth, NH; Cambridge, MA)

Analysis: This final example contains only one possible city, Dover, NH. The possible city is only one degree on anonymity from the target computer suggesting strong correlation. The possible city is approached from the south by Cambridge and Portsmouth. As Portsmouth is within 5 miles of Dover, one can conclude that it is unlikely that the target computer exists south of Dover.

Accuracy: South Berwick is located 5 miles northeast of Dover. The intent of this trace was to demonstrate that traces are also very accurate in tracing individual machines, as well as domain servers.

Plotted Results of Residential Prototype Trace

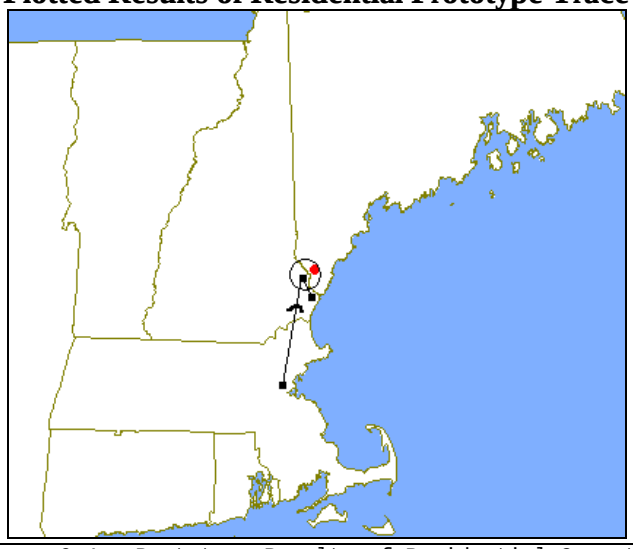


Figure 6.4: Prototype Results of Residential Computer
Maps Courtesy of TIGER [20].

6.2 Conclusions

The distributed approach to IP location was built upon the research and methods of other contemporary projects in the field. Following the investigation of such projects and the limitations of the problem, it was determined that there were two issues that had not been properly addressed in any of the investigated projects.

- There must be a geographical basis from which the geographical problem is solved.
- The results must accommodate the limitations of the problem.

Using available methodology and distributing it geographically address both of these concerns in an eloquent manner. Whereas before, a single traceroute would provide a specific result of uncertain correctness, the distributed traceroute is able to provide a more general result of increased correctness. Generally, this is a better approach when handling problems of such uncertainty. Moreover, the distributed approach will produce results no less correct than the single traceroute approach.

Much of the effectiveness of the approach hinges upon the set of traceroute servers that are used to populate the location data. The direction from which the target computer is approached, and the networks by which the trace travels determine its geographical locations and limitations.

Unfortunately, accommodating the limitations mean understanding that the granularity with which potentially dangerous web content can be located. The granularity is not very high. Using this distributed approach, or any approach for that matter, to determining the location of a specific terrorist machine is not going to be effective. The distributed approach is, however, effective for examining geographical and network trends of specific terrorist content.

6.2 Future Work

This design and prototype of the Distributed Traceroute IP Locator is merely the first step of problem solving. The future goals for such a project include not only developing and improving the system as an effective tool for geographically locating IP addresses, but also integrating it as a tool for monitoring terrorist content.

Certainly, the prototype is merely for demonstrative purposes, and must be overhauled with a more stable effective implementation of the solution. The patchwork nature of the prototype does not suit the implementation well, as there are several incompatibilities interfacing the different components.

An equally as important task is keeping the knowledge base up-to-date and robust. The design of the system is only as good as the data it has to work with. In the case of a distributed traceroute IP locator, that data is a comprehensive database of all networks that provide location clues in their computer host-names, and a collection of geographically and network-diverse public traceroutes.

These preceding steps suggest enhancements to improve the distributed approach proposed. The next logical evolution of the design would involve incorporating intelligent analysis to the results, and integrating the system into a project that will use the geographic information to examine trends in geographical location and web content.

Appendix A: Bibliography

[1] Brian, Marshall. How Web Servers Work. HowStuffWorks.com. 18 Apr. 2003 <http://www.howstuffworks.com/web-server10.htm>.

[2] Chritiansen, Tom, Randal L. Schwartz, and Larry Wall. Programming Perl. 2nd ed. Sepastopol, CA: O'Rielly & Associates, Inc., 1996.

[3] Cong. House. Accuracy And Integrity Of The WHOIS Database. US 107th Cong., 2nd sess. 22 May 2002. 30 Jan. 2003

[4] Davis, C., et al. RFC 1876 - A Means for Expressing Location Information in the Domain Name System. Jan. 1996. 30 Jan. 2003 <http://www.ckdhr.com/dns-loc/rfc1876.txt>.

[5] Davis, Christopher. DNS LOC – Geo-enabling the Domain Name System. 18 Mar. 2001. 30 Jan. 2003 <http://www.ckdhr.com/dns-loc/>.

[6] Fadia, Ankit . Getting geographical Information using an IP Address. Astalavista.net. 30 Jan. 2003 <http://www.astalavista.com/library/basics/organisation/ip-geography.shtml>.

[7] Fadia, Ankit. Tracing The Traceroute. 24 Jan. 2002. 30 Jan. 2003 <http://www.ankitfadia.com/traceroutew.html>.

[8] Franklin, Curt. Hows DSL Works. HowStuffWorks.com. 18 Apr. 2003 <http://computer.howstuffworks.com/dsl.htm>.

[9] Geobytes HomePage. 11 Apr. 2003. GeoBytes, Inc. 18 Apr. 2003 <http://www.geobytes.com>.

[10] IANA Home Page. 30 Dec. 2002. IANA. 30 Jan. 2003 <http://www.iana.org/>.

[11] ICANN. 18 Apr. 2003. The Internet Corporation For Assigned Names And Numbers. 18 Apr. 2003 <http://www.icann.org>.

[12] Kernen, Thomas. traceroute.org. 30 Jan. 2003 <http://www.traceroute.org>.

[13] Kvale, Mark. Perl Regular Expressions - Expanded Tutorial. 2000. Creative Fundamentals, Inc. 30 Jan. 2003 <http://tlc.perlarchive.com/articles/perl/pm0001_perlretut.shtml#using%20regular%20expressions%20in%20perl>.

[14] Moore, David, Jim Donohoe, and Ram Periakaruppan. Where in the World is netgeo.caida.org? Cooperative Association for Internet Data Analysis. 30 Jan. 2003 <http://www.caida.org/outreach/papers/2000/inet_netgeo/inet_netgeo.html>.

[15] Nemeth, Evi, and Ram Periakaruppan. "GTrace - A Graphical Traceroute Tool." (n.d.). 30 Jan. 2003 <http://www.caida.org/tools/visualization/gtrace/paper/GTrace.pdf>.

[16] NET::WHOIS. Comp. Dana Hudes , and Chip Salzenberg . Vers. 1.9. Apr. 1997. CPAN. 30 Jan. 2003 <http://search.cpan.org/author/DHUDES/Net-Whois-1.9/Whois.pm>.

[17] NetWorldMap. 4 Jan. 2003. NetWorldMap. 30 Jan. 2003 <http://www.networldmap.com/>.

[18] SarangWorld Traceroute Project. Comp. Sarang Gupta. 9 Mar. 2003. 30 Jan. 2003 <http://www.sarangworld.com/TRACEROUTE/>.

[19] The Internet's Network Information Center. 7 Feb. 2003. InterNIC. 18 Apr. 2003 <http://www.internic.com>.

[20] TIGER Map Server. Vers. 1998. US Census Bureau. 20 Apr. 2003 <http://tiger.census.gov/cgi-bin/mapbrowse-tbl>.

[21] VisualRoute by VisualWare. VisualWare. 30 Jan. 2003 <http://http://www.visualware.com/visualroute/index.html>.

[22] What do the digits in phone numbers mean? HowStuffWorks.com. 18 Apr. 2003 <http://www.howstuffworks.com/question659.htm>.

[23] WHOIS. The Chinese University of Hong Kong. 18 Apr. 2003 <http://www.cuhk.edu.hk/guides/earn/whois.html>.

[24] Yahoo! Geocities - Terms of Service. Yahoo! Geocities. 30 Jan. 2003 <http://docs.yahoo.com/info/terms/geoterms.html>.

[25] Zook, Matthew. Zooknic Internet Intelligence. 5 Aug. 2002. 30 Jan. 2003 <http://www.zooknic.com>.

[26] Connolly, Gene M., Anatoly Sachenko and George Markowsky. "Web Neighborhood Watch." Proceedings of 2003 Spring IEEE Conference on Technologies For Homeland Security, May 7-8, 2003, Waltham, Massachusetts. P.

[27] Connolly, Gene M., Anatoly Sachenko and George Markowsky. "Distributed Traceroute Approach to Locating IP Devices." Proceedings of IEEE Second Workshop on Intelligent Data Acquisition and Advanced Computing Systems, September 8-10, L'viv, Ukraine. ACCEPTED.

Appendix B: Source Code

Hostname Regular Expressions & Code Translations are courtesy of Sarang Gupta, and available at:

http://www.sarangworld.com/TRACEROUTE/

```
use LWP::Simple;
use Net::Whois;
use Carp;

#Gather Data
@hostlist = Execute_Traces( "tracert.txt", $ARGV[0] );
@patterns = Collect_Patterns( "patterns.txt" );
@translations = Collect_Translations( "out.txt" );

#Translate Host Names
@citylist = Locate_Hosts( [ @hostlist ], [ @patterns ], [ @translations
] );
@formatcitylist = Consolidate_Paths( @citylist );

#Build Location Tree
@dat = Gather_Trace_Data( @formatcitylist );
@location_tree = Build_Location_Tree( @dat );

#Output Results
Output( @location_tree );
Output_Trace( @formatcitylist );


#########################################################################
## Subroutine Collect_Translations
##
## Arguments: Translation File Name
## Returns:   Array of city name translations
## Data Entry Format:
##
##    <citycode><statename><cityname><latitude><longitude>
##
## Description: The file specified by the only argument is opened.
##          Each line of the file is pushed onto a data array
##
#########################################################################

sub Collect_Translations
{
      open(TRANSLATION, pop( @_ ) );
      while($entry = <TRANSLATION>)
      {
            push( @trans, $entry );
            #push every translation on to an array
      }
```

```
      @trans; #return the array of translations
}

#########################################################################
## Subroutine Collect_Patterns
##
## Arguments: Pattern File Name
## Returns:   Array of known network host name regular expressions
## Data Entry Format:
##
##    Regular expression, or '#' delimited comment.
##
## Description: The file specified by the only argument is opened.
##          Each regular expression string in the file is
##          converted to a regular expression, and pushed on
##          a data array.  All comments are removed.
##
#########################################################################

sub Collect_Patterns
{
      open(PATTERNS, pop(@_) );
      while( $pattern = <PATTERNS> )
      {
        chomp($pattern);#Remove 'newline' character

        if( substr($pattern, 0, 1) ne "#" )
        #exclude all entries that start with '#'
        #these are comments
        {
            $lastchar = chop($pattern);
            #Examine the final character

            if( $lastchar eq "i" )
            {#Case Sensitive
                  chop($pattern);
                  $patreg = substr( $pattern, 1 );
                  $regex = qr/$patreg/i;
                  push(@patternlist, $regex); #push regular expression
onto array
            }
            else
            {#Not Case Sensitive
                  $patreg = substr( $pattern, 1 );
                  $regex = qr/$patreg/;
                  push( @patternlist, $regex ); #push regular
expression onto array
            }
        }
      }

      @patternlist; #return array of regular expressions
}
```

```
#########################################################################
## Subroutine Execute_Traces
##
## Arguments: IP address of Domain of Target computer
##          Filename of file containing Traceroute URLs
## Returns:   Array of references to unique arrays containing a list
##          of computer host names that comprise each traceroute.
##
## Description: For each traceroute URL in the given file, a trace is
## is run.  Every trace is then parsed for the computer host names
## and each is stored into and array.  The function returns an array
## of references that point to each trace array.
##
#########################################################################

sub Execute_Traces
{
      $target = pop(@_); #Target Computer IP or Domain Name.

      open(TRACESITES, pop(@_) ); #Open Traceroute File
      $count = 0;

      while( $traceurl = <TRACESITES> )
      {#Foreach URL in Traceroute file
        $count++;
        chop($traceurl);
        $traceaddy = $traceurl . "?" . $target; #Format URL for
Traceroute
        print "Running Trace #", $count, "... ";
        $tracecontent = get($traceaddy); #Request content
        @hostdata = Parse_Raw_Trace( $tracecontent ); #Parse Raw Data
into an
                                            #array of cmputer host names
        if( length(@hostdata) == 1 )
        {#Don't Add if Trace failed
            print "Failed.\n"
        }
        else
        {#Push reference to array of trace host names onto array
            push(@content, [ @hostdata ]);
            print "Completed.\n";
        }
      }
      print "\n\n";
      @content; #return Array of references to traces.
}
```

```
#########################################################################
## Subroutine Parse_Raw_Trace
##
## Arguments: String of HTML code that contains trace results
##
## Returns:   Array computer host names that comprise a trace
##
## Description: This function scans a string for each computer host
## name contained within it.  Each hostname is pushed onto an array
## and the array is returned.
##
#########################################################################

sub Parse_Raw_Trace
{
       $tracecontent = pop( @_ );
       # take argument

        undef @hostpath;
        while( $tracecontent =~ /(\d\s\s(\S+)((.*\n)*))+?/ )
        #Search for hostnames
        {
          push(@hostpath, $2);
          #Add discovered hostname to trace array
          $tracecontent = $3;
        }
      @hostpath;
}


#########################################################################
## Subroutine Locate_Hosts
##
## Arguments: Ref Array of Hostname Pattterns
##          Ref Array of Location Code Translations
##          Ref Array of References to Trace data arrays
##
## Returns:   Array of References to Tracedata arrays
## Data Format:
##          <city>, <state> :: <domain>
##
## Description: This function examines each trace data, attempts to
## match each hostname with a network pattern, and does a lookup
## on its code.  each entry in the trace arrays is replaced with a
## location/domain pair
##
#########################################################################

sub Locate_Hosts
{
      $translations = pop( @_ );
      $patterns = pop( @_ );
      $dataarray = pop( @_ );
      #take three arguments.

      foreach $traceref (@$dataarray)
```

```
	#examine each trace
	{
		undef @codelist;
		for $hop (@$traceref)
		#attempt to determine location of every
		#network hop on trace
		{
			if( $hop =~ /.*\.(.+)\.(\D+)/)
			#extract secondary and top level domain
			#from hostname.  Only process if the hop
			#is a hostname, not an IP
			{
				$dom = $1 . "." . $2;

				$citycode = Match_Pattern( $hop, $patterns );
				#find pattern and extract code for hostname

				$location = Match_Translation( $citycode,
$translations );
				#find city & state for the discovered location
code

				$codeentry =  $location ."\t\t :: " . $dom;
				push(@codelist, $codeentry);
				#store results as location/domain pair
			}
			else
			{
				#if hop is an IP address, location is unknown.
				$codeentry = "<unknown>, <un>\t\t :: " . $hop;
				push(@codelist, $codeentry);
			}
		}
		push( @citylisting, [ @codelist ]);
	}

	@citylisting;
}

#########################################################################
## Subroutine Match_Pattern
##
## Arguments: Array of Hostname Pattterns
##          Hostname
##
## Returns:   Location code, or "<unknown>"
##
## Description: This function attempts to match a hostname against a
## list of hostname patterns.  Once a match is found, a location code
## is extracted from the hostname, and the function returns.  If the
## matching fails, the function returns "<unknown>"
##
#########################################################################

sub Match_Pattern
{
	$patternlist = pop( @_ );
```

```
	#get pattern array argument

	$hop = pop( @_ );
	$city = "<unknown>"; #default, if pattern cannot be matched
	foreach $pattern (@$patternlist)
	{
	#compare with each pattern
		if( $hop =~  /$pattern/ )
		{
			#if there is a match extract the code
			$city = $1;
			if( $2 )
			#code may have two parts
			{
				$city = $1 . "." . $2;
			}
			last;
		}
	}
	#return code
	$city;
}

#############################################################################
## Subroutine Match_Translation
##
## Arguments: Array of Location Code Translations
##          Location Code
##
## Returns:   "<city>, <state>" or "<unknown>"
##
## Description: Matches the location code with the associated city
## and state
##
#############################################################################

sub Match_Translation
{
	$trans = pop( @_ );
	$city = pop( @_ );
	#take translation array and location code
	#off argument list

	foreach $tran (@$trans)
	#attempt to match location code with each translation
	{
		$locationstate = "<un>";
		$locationcity = "<unknown>";
		$tran =~ /(\S+)/;
		if($city eq "<unknown>")
		{
		#if no location code is known, escape
			last;
		}
		if( $city eq $1 )
		{
		#if location code matches translation entry
```

```
            #extract location city and state.
                  $tran =~ /\S+\s+(\w+)\s+(.+)\s+\S+\s+\S+/;
                  $locationstate = $1;
                  $locationcity = $2;
                  last;
            }
      }

      $location = $locationcity . ", " . $locationstate;
      #return the determined location
}

#########################################################################
## Subroutine Consolidate_Paths
##
## Arguments: Array of references to trace data arrays
##
## Returns:   Array of references to trace data arrays
##
## Description: This function runs the tracedata through two functions
## inorder to reduce unnecessary information.
##
#########################################################################

sub Consolidate_Paths
{
      Remove_Duplicate_Nodes( Trim_Leading_Unknowns( @_ ) );
}

#########################################################################
## Subroutine Trim_Leading_Unknowns
##
## Arguments: Array of references to trace data arrays
##
## Returns:   Array of references to trace data arrays
##
## Description: This function examines each array of tracedata,
## removing all unknowns, from the beginning until a node is located
##
#########################################################################

sub Trim_Leading_Unknowns
{
      foreach $traceref (@_)
      #examine each trace
      {
            $bool = 1;
            undef @ftrace;
            foreach $node ( @$traceref )
            #examine each node of each trace
            #starting from the beginning
            {
                  if( $node =~ /<unknown>/ && $bool == 1)
                  #if the node is unknown, and a city
                  #has not been located proceed.
                  {
                        next;
```

```
                }
                else
                #otherwise push it on to the trace, and
                #set the bool to prevent removing future unknowns
                {
                        $bool = 0;
                        push( @ftrace, $node );
                }
        }
        push( @fcitylist, [@ftrace] );
    }

    @fcitylist;
}

#########################################################################
## Subroutine Remove_Duplicate_Nodes
##          Remove_Duplicates
##
## Arguments: Array of references to trace data arrays
##
## Returns:   Array of references to trace data arrays
##
## Description: This function examines each pair of nodes, removing
## one if it is a duplicate
##
#########################################################################

sub Remove_Duplicate_Nodes
{
    foreach $traceref (@_)
    #examine each trace for duplicates
    {
        push( @fnodupcitylist, [  Remove_Duplicates( @$traceref )
] );
    }

    @fnodupcitylist;
}

sub Remove_Duplicates
{
    @list = @_;
    undef @nodup;
    for $i (0 .. @list )
    #for each node check to see if the following node is
    #identical, if it is not, push the node on to the trace.
    {
        if( $list[$i] ne $list[$i + 1] )
        {
            push( @nodup, $list[$i] );
        }
    }
    @nodup;
}
```

```
#########################################################################
## Subroutine Sort_Prune
##
## Arguments: Array
##
## Returns:   Sorted Unique Array
##
## Description: This function takes an array, sorts its contents, and
## then removes all duplicates.  the intent of this function is to
## create a list where every entry is unique
##
#########################################################################


sub Sort_Prune
{
      @list = sort @_; #sort
      Remove_Duplicates(@list); #prune
}

#########################################################################
## Subroutine Gather_Trace_Data
##          Trace_Data
##
## Arguments: Array of references to tracedata arrays
##
## Returns:   Array of references to Trace Result Structure
## Data Format:  The trace result structure contains the following
## entrys in the following order:
##
##          <Destination Node Location/Domain Pair>
##          <First Located City Location/Domain Pair>
##          <Second Located City Location/Domain Pair>
##          <Reference to Array of Local Network Steps>
##
## Description: This function examines each trace.  By moving backward
## through the trace the following entries are extracted: Desintaion,
## Closest Located City to desitnation (in terms of network steps),
## Second Closest City, and an array of all the nodes between the
## closest city and the destination.
##
#########################################################################
sub Gather_Trace_Data
{
      foreach $trace ( @_ )
      #execute for every trace
      {
            push( @tracedat, [ Trace_Data( @$trace ) ] );
      }

      @tracedat;
}

sub Trace_Data
{
      @reversetrace = reverse @_;
```

```
      #reverse trace so it can be examined starting with the
destination
      $first = 1;
      $nextbest = 1;

      undef @tracedata;
      undef @localnet;

      #default entry if extraction fails.
      $destination = "<unknown>, <un> :: <unknown>";
      $firstcity = "<unknown>, <un> :: <unknown>";
      $secondcity = "<unknown>, <un> :: <unknown>";

      foreach $node (@reversetrace)
      #examine each node starting with the destination moving back
      {
            if( $first )
            #extract first node as desintation
            {
                  $destination = $node;
                  $first = 0;
                  next;
            }
            elsif( $nextbest && ($node =~ /<unknown>.*/) )
            #while no city can be located stepping back from
            #the destination, add nodes to local network array
            {
                  push( @localnet, $node );
                  next;
            }
            elsif( $nextbest )
            #once a city is found, store it as the closest city
            {
                  $firstcity = $node;
                  $nextbest = 0;
                  next;
            }
            elsif( ($node ne $firstcity) && !($node =~ /<unknown>.*/) )
            #find the next closest city, that is not unknown, and not
the
            #same as the closest.
            {
                  $secondcity = $node;
                  last;
            }
      }

      push( @tracedata, $destination );
      push( @tracedata, $firstcity );
      push( @tracedata, $secondcity );
      push( @tracedata, [ @localnet ] );
      #create trace data array structure.
      @tracedata;
}
```

```
#########################################################################
## Subroutine Build_Location_Tree
##          Add_Possible_Cities
##
## Arguments: Array of References to the trace data structures
##
## Returns:   Location Tree data structure
## Data Format: The following are the tiers of the location tree.
##    1st Tier: References to Possible Destinations
##    2nd Tier: <Destination Location/Domain Pair>
##          <Reference to a list of Possbile Cities>
##    3rd Tier: References to Possible Cities Structure
##    4th Tier: <Possible City Location/Domain Pair>
##          <Reference to Local Net Array>
##          List of Second City Location/Domain Pairs
##    5th Tier: List of Local Net Location/Domain Pairs
##
## Description: This function takes each Trace Data strucutre and
## integrates it into a location tree structure.
##
#########################################################################

sub Build_Location_Tree
{
      foreach $tracedat (@_)
      #integrate every piece of trace data
      {
            $found = 0;
            if( $$tracedat[0] =~ /.*::\s<unknown>/ )
            #skip entry if destination exists on a
            #unknown network
            {
                  next;
            }

            foreach $approved (@approvedlist)
            #examine exisiting desintaion node entries on the
            #location tree
            {
                  if( $$tracedat[0] eq $$approved[0])
                  #if the trace data destination is already represented
on the
                  #tree, examine lower tires of that destination branch
                  {
                        $found = 1;
                        Add_Possible_Cities( $$tracedat[1],
$$tracedat[2],
                                             $$tracedat[3], $$approved[1]
);
                        last;
                  }
            }

            undef @possiblecity;
            undef @possiblecities;
            undef @info;
```

```
            if( $found == 0 )
            #if destination node is not represented on the tree, add
            #the appropriate node.
            {
                  push( @possiblecity, $$tracedat[1] );
                  push( @possiblecity, $$tracedat[3] );
                  push( @possiblecity, $$tracedat[2] );
                  push( @possiblecities, [ @possiblecity ] );

                  push( @info, $$tracedat[0] );
                  push( @info, [ @possiblecities ] );
                  push( @approvedlist, [ @info ] );

                  #create new structure to append to the tree,
including
                  # possible city information
            }
      }
      @approvedlist;
}

sub Add_Possible_Cities
#This function is called to add new possible cities to
#a destination node that already exists on the location tree
{
      $possiblecitiesref = pop( @_ );
      $localnet = pop( @_ );
      $secondcity = pop( @_ );
      $firstcity = pop( @_ );
      #grab arguments

      $cityfound = 0;

      foreach $possiblity ( @$possiblecitiesref )
      #examine each possible city attached to particular
      #destination node
      {
            if( $firstcity eq $$possiblity[0] )
            #if the possible city is already represented,
            #add examine its second cities
            {
                  $cityfound = 1;
                  $new = 1;
                  for $i ( 2 .. @$possiblity )
                  #examine the second cities, to see if new second city
                  #is already represented on the location tree
                  {
                        if( $secondcity eq $$possiblity[$i])
                        {
                              $new = 0;
                              last;
                        }
                  }
                  if( $new )
                  {
                        #add new second city to possible city node
                        push( @$possiblity, $secondcity );
```

```
                  }
                  last;
            }
      }

      undef @newpossiblity;
      if( $cityfound == 0 )
      #if possible city is not represented, add the appropriate node
      {
            push( @newpossiblity, $firstcity);
            push( @newpossiblity, $localnet );
            push( @newpossiblity, $secondcity );
            push( @$possiblecitiesref, [ @newpossiblity ] );
      }
}

#############################################################################
## Subroutine Output
##
## Arguments: Location Tree Data Structure
##
##
## Description: This funciton outputs a representation of the
## Location tree.  It also makes several calls to the WHOIS database.
##
#############################################################################

sub Output
{
      foreach $Destination (@_)
      #print the following for each desintation node in the tree
      {
            ($city, $network) = Split_Entry( $$Destination[0] );
            #Separate the Location/Domain pair of the Destination Node
            $w = new Net::Whois::Domain $network;
            #execute WHOIS lookup

            print
"**************************************************\n";
            print "---Destination: \t", $city , "\n";
            if( defined $w && $w->name ne "" )
            #print WHOIS info, only if lookup was successful
                  {print "---Network:       \t", $w->name , "\n";
                   print "---ContactAddress:\t", $w->address, "\n\n";}
            else
            #otherwise print domain name
                  {print "---Network:       \t", $network , "\n";}


            $PossibleCitiesPtr = $$Destination[1];

            foreach $PossibleCity (@$PossibleCitiesPtr)
            #print the following for each possible city
            {
                  ($city, $network) = Split_Entry( $$PossibleCity[0] );
                  #Separate the Location/Domain pair of the Possible
city Node
```

```
                  $w = new Net::Whois::Domain $network;
                  #execute WHOIS lookup

                  print
"************************************************\n";
                  print "**Possible City: \t", $city , "\n";

                  if( defined $w && $w->name ne "" )
                  #print WHOIS info, only if lookup was successful
                        {print "**Network:       \t", $w->name , "\n";
                         print "**ContactAddress:\t", $w->address,
"\n";}
                  else
                  #otherwise print domain name
                        {print "**Network:       \t", $network , "\n";}

                  print "**Via: \n";
                  for $i (2 .. @$PossibleCity)
                  #print the following for each second city
                  {
                        ($city, $network) = Split_Entry(
$$PossibleCity[$i] );
                        #Separate the Location/Domain pair of the
Possible city Node
                        print "**\t\t\t", $city, "\n";
                  }
                  $localnet = $$PossibleCity[1];

                  print "**Intermediate Networks: \n";
                  foreach $net (@$localnet)
                  #Print out each node of the local net
                  {
                        ($city, $network) = Split_Entry( $net );
                        #Separate the Location/Domain pair of the
Possible city Node
                        $w = new Net::Whois::Domain $network;
                        #execute WHOIS lookup

                        if( defined $w && $w->name ne "" )
                        #print WHOIS info, only if lookup was
successful
                              {print "**\t\t\t", $w->name , "\n";
                              print "**\t\t\t->", $w->address, "\n";}
                        else
                        #otherwise print domain name
                              {print "**\t\t\t", $network , "\n";}
                  }
            }
            print
"************************************************\n";
      }
}
```

```
#########################################################################
## Subroutine Output_Trace
##
## Arguments: Array of References to Trace data arrays
##
##
## Description: This funciton outputs the data from each traceroute
## executed
##
#########################################################################

sub Output_Trace
{
      foreach $trace (@_)
      #output each trace
      {
            print "Trace: \n";
            print
"************************************************\n";
            foreach $node (@$trace)
            #print each location/domain pair
            {
                  print "**  ", $node, "\n";
            }
            print
"************************************************\n\n";
      }
}

#########################################################################
## Subroutine Split_Entry
##
## Arguments: Locatio/Domain Pair
##
## Return: Array containing two elements: Location and Domain
##
##
## Description: This function is used to split the Location/Domain
## pair into separate entries, Location and Domain.
##
#########################################################################
sub Split_Entry
{
      undef @couple;
      $entry = pop( @_ );

      $entry =~ /(.*)\s*::\s(.*)/;
      #split each entry at the double colon

      push( @couple, $1);
      push( @couple, $2);
      @couple;
}
```

Appendix C: Results

Trace: IEEE.org

Geographical Location: Piscataway, NJ

```
**************************************************
---Destination:         <unknown>, <un>
---Network:             *
**************************************************
**Possible City:        New York, NY
**Network:              bbnplanet.net
**Via:
**                      <unknown>, <un>
**                      Philadelphia, PA
**
**Intermediate Networks:
**                      ieee.org
**                      bbnplanet.net
**************************************************
**Possible City:        Pennsauken, NJ
**Network:              sprintlink.net
**Via:
**                      Fort Worth, TX
**                      Newark, NJ
**
**Intermediate Networks:
**                      ieee.org
**                      sprintlink.net
**************************************************
Trace:
**************************************************
**  New York, NY                :: bbnplanet.net
**  <unknown>, <un>             :: bbnplanet.net
**  <unknown>, <un>             :: ieee.org
**  <unknown>, <un>             :: *
**************************************************

Trace:
**************************************************
**  Phoenix, AZ          :: twtelecom.net
**  Los Angeles, CA             :: twtelecom.net
**  Anaheim, CA          :: sprintlink.net
**  Fort Worth, TX              :: sprintlink.net
**  Pennsauken, NJ              :: sprintlink.net
**  <unknown>, <un>             :: sprintlink.net
**  <unknown>, <un>             :: ieee.org
**  <unknown>, <un>             :: *
**************************************************
```

```
Trace:
*****************************************************
**  Boston, MA           :: qwest.net
**  Newark, NJ           :: qwest.net
**  <unknown>, <un>              :: 205.171.1.122
**  Pennsauken, NJ               :: sprintlink.net
**  <unknown>, <un>              :: sprintlink.net
**  <unknown>, <un>              :: ieee.org
**  <unknown>, <un>              :: *
*****************************************************

Trace:
*****************************************************
**  Washington, DC               :: att.net
**  <unknown>, <un>              :: genuity.net
**  Washington, DC               :: bbnplanet.net
**  Philadelphia, PA             :: bbnplanet.net
**  New York, NY                 :: bbnplanet.net
**  <unknown>, <un>              :: *
**  <unknown>, <un>              :: ieee.org
**  <unknown>, <un>              :: *
*****************************************************

Trace:
*****************************************************
**  Austin, TX           :: att.net
**  <unknown>, <un>              :: *
**  Dallas, TX           :: att.net
**  <unknown>, <un>              :: sprint.net
**  Fort Worth, TX               :: sprintlink.net
**  Pennsauken, NJ               :: sprintlink.net
**  <unknown>, <un>              :: sprintlink.net
**  <unknown>, <un>              :: ieee.org
**  <unknown>, <un>              :: *
*****************************************************

Trace:
*****************************************************
**  Carrollton, TX               :: bbnplanet.net
**  Indianapolis, IN             :: bbnplanet.net
**  Philadelphia, PA             :: bbnplanet.net
**  New York, NY                 :: bbnplanet.net
**  <unknown>, <un>              :: bbnplanet.net
**  <unknown>, <un>              :: ieee.org
**  <unknown>, <un>              :: *
*****************************************************
```

Trace: ICANN.org

Geographical Location: Marina Del Rey, CA

```
**************************************************
---Destination:          <unknown>, <un>
---Network:              *
**************************************************
**Possible City:         Los Angeles, CA
**Network:               verio.net
**Via:
**                       San Jose, CA
**                       Milpitas, CA
**                       Dallas, TX
**
**Intermediate Networks:
**                       192.0.33.1
**************************************************
Trace:
**************************************************
**  San Jose, CA                 :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Phoenix, AZ           :: twtelecom.net
**  Modesto, CA           :: twtelecom.net
**  Oakland, CA           :: twtelecom.net
**  San Francisco, CA            :: twtelecom.net
**  San Jose, CA                 :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Boston, MA            :: qwest.net
**  Newark, NJ            :: qwest.net
**  Newark, NJ            :: verio.net
**  Milpitas, CA                 :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************
```

```
Trace:
**************************************************
**  Washington, DC               :: att.net
**  Ashburn, VA          :: verio.net
**  Sterling, VA                 :: verio.net
**  Dallas, TX           :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Austin, TX           :: att.net
**  <unknown>, <un>              :: *
**  Dallas, TX           :: att.net
**  Dallas, TX           :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Dallas, TX           :: verio.net
**  Los Angeles, CA              :: verio.net
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**  <unknown>, <un>              :: 192.0.33.1
**  <unknown>, <un>              :: *
**************************************************
```

Trace: ICANN.org

Geographical Location: Atlanta, GA

```
**************************************************
---Destination:         <unknown>, <un>
---Network:             *
**************************************************
**Possible City:        Atlanta, GA
**Network:              atdn.net
**Via:
**                      <unknown>, <un>
**                      Houston, TX
**                      Charlotte, NC
**
**Intermediate Networks:
**************************************************
```

```
Trace:
**************************************************
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Phoenix, AZ          :: twtelecom.net
**  Los Angeles, CA              :: twtelecom.net
**  <unknown>, <un>              :: atdn.net
**  Phoenix, AZ          :: atdn.net
**  Houston, TX          :: atdn.net
**  <unknown>, <un>              :: atdn.net
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Vienna, VA           :: atdn.net
**  Charlotte, NC                :: atdn.net
**  <unknown>, <un>              :: atdn.net
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Washington, DC               :: att.net
**  <unknown>, <un>              :: winstar.net
**  <unknown>, <un>              :: atdn.net
**  Charlotte, NC                :: atdn.net
**  <unknown>, <un>              :: atdn.net
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************

Trace:
**************************************************
**  Austin, TX           :: att.net
**  <unknown>, <un>              :: *
**  Chicago, IL          :: att.net
**  <unknown>, <un>              :: aol.com
**  Chicago, IL          :: atdn.net
**  Vienna, VA           :: atdn.net
**  Charlotte, NC                :: atdn.net
**  <unknown>, <un>              :: atdn.net
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************
```

```
Trace:
**************************************************
**  Dallas, TX           :: atdn.net
**  Houston, TX          :: atdn.net
**  <unknown>, <un>              :: atdn.net
**  Atlanta, GA          :: atdn.net
**  <unknown>, <un>              :: *
**************************************************
```

Trace: Residential Computer

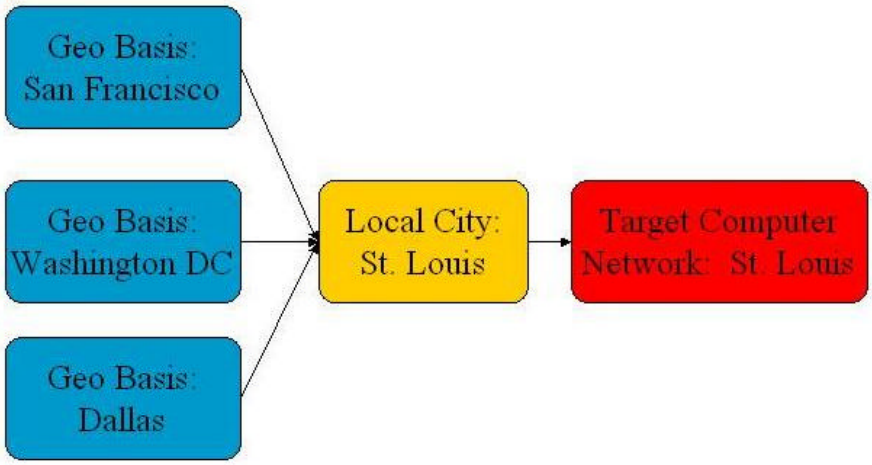
Geographical Location: South Berwick, ME

```
**************************************************
---Destination:         <unknown>, <un>
---Network:             attbi.com
**************************************************
**Possible City:        Dover, NH
**Network:              AT&T Broadband
**ContactAddress:       188 INVERNESS DR WENGLEWOOD, CO 80112-5202
**Via:
**                      Portsmouth, NH
**                      Cambridge, MA
**
**Intermediate Networks:
**************************************************
Trace:
**************************************************
**  Chicago, IL          :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Portsmouth, NH               :: attbb.net
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************

Trace:
**************************************************
**  Phoenix, AZ          :: att.net
**  <unknown>, <un>              :: *
**  Chicago, IL          :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Portsmouth, NH               :: attbb.net
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************
```

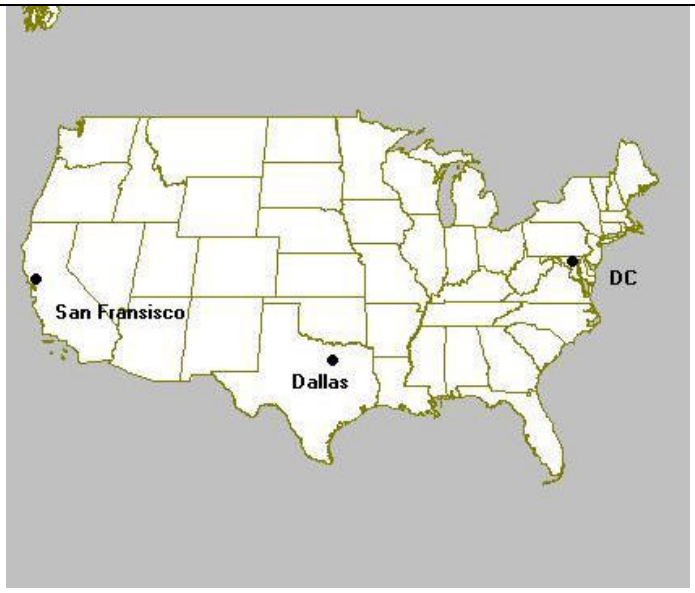
```
Trace:
**************************************************
**  Boston, MA           :: qwest.net
**  New York, NY                 :: qwest.net
**  New York, NY                 :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Portsmouth, NH               :: attbb.net
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************

Trace:
**************************************************
**  Washington, DC               :: att.net
**  New York, NY                 :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Portsmouth, NH               :: attbb.net
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************

Trace:
**************************************************
**  Austin, TX           :: att.net
**  <unknown>, <un>              :: *
**  Chicago, IL          :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Portsmouth, NH               :: attbb.net
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************

Trace:
**************************************************
**  Austin, TX           :: att.net
**  <unknown>, <un>              :: *
**  Chicago, IL          :: att.net
**  Cambridge, MA                :: att.net
**  <unknown>, <un>              :: 12.125.47.18
**  <unknown>, <un>              :: 24.147.0.213
**  Dover, NH            :: attbb.net
**  <unknown>, <un>              :: attbi.com
**************************************************
```

Appendix D: Design Example

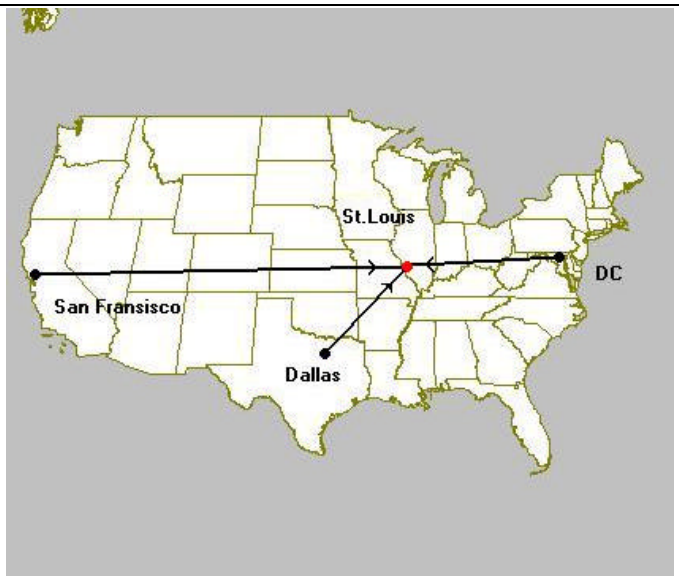
The figure below represents the 'Location Tree' of the distributed traceroute search on slso.org, the domain for the St. Louis Symphony Orchestra. In this example, the location tree is color coded for its three levels. The red node denotes the target computer, the orange nodes denote the 'Possible Cities', and the blue nodes denote the 'Second Cities'.



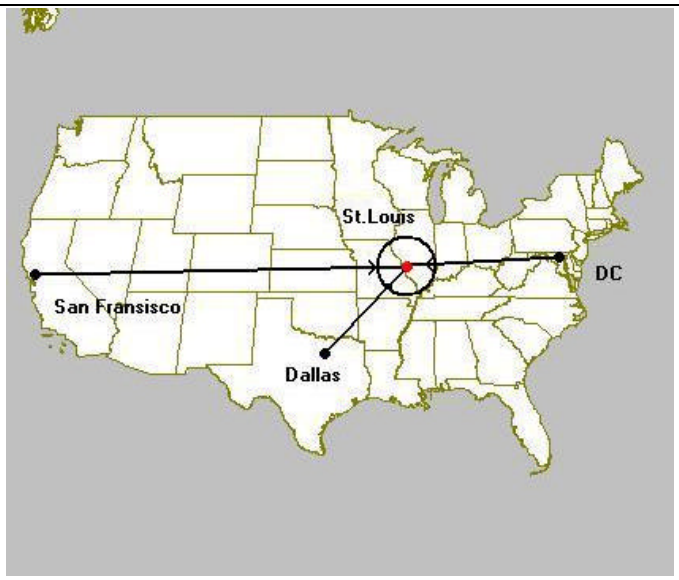
Each level of the 'Location Tree' provides additional information regarding the location of the target computer. Consider the following three Figures, each of which represent another level of the 'Location Tree' above. The first figure only plots the 'Second Cities'. This creates the geographical basis with which to solution can be achieved. San Fransisco, Washington, and Dallas are geographically diverse within the United States.



The second figure links these ‘Second Cities’ to their appropriate ‘Possible Cities’. This creates directional vectors that suggest the location of the target computer. The set of ‘Possible Cities’ suggests a ‘cloud’ or ‘region’ that the target computer exists within. In the slso.org example, St. Louis is the only ‘Possible City’.



The final figure includes a circle surrounding the location of the location of the ‘Contact Information’ included in the target computer’s WHOIS database entry. This data confirms the location of target computer as a reflection of the ‘Possible Cities’.



Appendix E: Author's Biography

Gene Michael Connolly was born in South Berwick, Maine on April 14 1981. He was raised in South Berwick and graduated from Marshwood High School in 1999. Majoring in Computer Science, Gene has a minor in mathematics. He has been a member of the varsity Swimming Team all four years at the University of Maine, and was elected as captain of the men's team for his senior year. Upon graduation in May of 2003, Gene plans on pursuing a professional career in Computer Science.