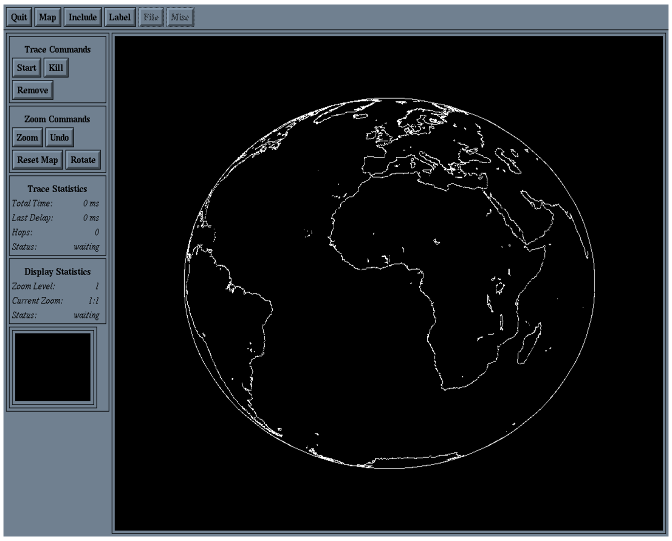
GeoTraceMan



The Globe

Computing Project 394
Three Dimensional Extensions to GeoTraceMan

Scott Pleitner
921626F
email: pleitner@cs.curtin.edu.au

Supervisors: Craig Farrell and Mike Schulze
email: {craig,mike}@cs.curtin.edu.au

Table of Contents

List of Figures

Abstract

GeoTraceMan is a network monitoring tool that aids in understanding the positions of hosts throughout the Internet. GeoTraceMan achieves this by drawing trace lines on a map of the world. These trace lines represent the path that Internet Protocol (IP) packets could take to get from one host to another.

GeoTraceMan also provides an appropriate delay statistic for each trace. GeoTraceMan's output is presented in a simple to comprehend and use X Windows interface.

The map can display trace information in either a flat (cylindrical projection) map or on a three dimensional globe. Additional map functionality includes zooming and rotation.

Preface

This preface is dedicated to thanking those people who helped me throughout this project. First and foremost to receive my thanks are my supervisors Craig Farrell and Mike Schulze. Without their help I would not have even attempted this project, let alone come close to finishing it. Craig Farrell also deserves a special mention for producing the majority of the DNS code, it is unfortunate it was not implemented.

Darren Brown deserves a large thank you for developing GeoTraceMan initially. His help during this project, especially when I did not understand some work he had previously completed, was essential. My thanks also goes to John Krivitsky for the help he offered early in the project when quad trees were being considered. It is a pity they were not implemented as John was an invaluable source of information in this area. Duke Tantiprasut also deserves to be mentioned for his understanding when I changed projects.

I would also like to thank Dr Geoff West and Peg Foo Siew for their help, both direct and indirect, with producing the necessary mathematics for the filter function used to covert the data from two to three dimensions.

A special thank you must be extended to Mazahir Songerwala for being the "beating post" during this project. My daily frustration would be taken out on Maz who was having enough trouble of his own, yet he still took my abuse.

This report is by no stretch of the imagination an explanation on how to use GeoTraceMan. It aims to provides a detailed insight into the design and implementation of GeoTraceMan from its origins to its release state. A manual page is available with the distribution of GeoTraceMan and a some day *real soon now*, users' and programmers' reference manuals will be provided. These documents should provide all of the information that is not supplied in this report.

1.0 Introduction

The Internet is a vast community of internetworked networks. As such, it can be quite difficult to understand, in a geographical sense, where each host resides. It is quite a simple task to discover the section of the Internet (subnet) that a host resides on, but this tells us nothing about its geographical location (an example is CUTNET, the Curtin University network, which has a subnet and host over 700 kilometers away from the majority of the network). GeoTraceMan is a tool whose aim is to provide the geographical location of any required host.

GeoTraceMan was developed by Darren Brown, a student at Curtin University of Technology, as a solution to this problem [Brown, 1993]. GeoTraceMan was designed to show, on a map of the world, where a host exists and a possible route packets travel to get there. This was achieved.

1.1 Netman

The Netman series of programs is a suite of network monitoring and management tools developed by the Department of Computing Science at Curtin University of Technology.

As a member of this suite of tools, the final goal of GeoTraceMan was incorporation into the Netman release. In its original state, GeoTraceMan was not robust enough, nor contained enough features to be included in the Netman release which included utilities such as *etherman*, *packetman* and *interman*.

Netman work (binaries only) completed to date is available via anonymous ftp from `ftp.cs.curtin.edu.au`. The matching source for the above binaries can be purchased from Curtin University of Technology. All inquiries should be sent via email to `netman@cs.curtin.edu.au` or via standard mail (snail mail) to:

Netman Development Group
School of Computing
Curtin University of Technology
GPO Box U 1987
Perth 6001
Western Australia

Netman was created by Mike Schulze, George Benko and Craig Farrell to provide "effective network diagnostics that are within the financial reach of network managers within small organisations" [Schulze et al, 1993]. To date, all of the Netman tools operate by promiscuous, passive monitoring of the network under study.

The first product released by Netman was *etherman*. This tool monitors ethernet traffic on the local ethernet network. It displays talking hosts in real-time and the amount of traffic over the network. The network statistics gathered by *etherman* are displayed graphically and can be stored in a textual format. Another tool currently under development by the Netman group, *analyser*, will take these statistics and produce a optimal network segmentation recommendation.

interman was the next tool developed by the Netman group. This monitored a network level protocol, specifically IP. It can be used to diagrammatically show connections, and the transport and higher layer protocols in use, between hosts on different networks.

Packet analysis in the form of *packetman* was the next tool created by Netman. It allowed packet streams to be captured and analysed in depth at a later date. By building in protocol descriptions, *packetman* was able to *decipher* packets and the protocols within. Greg Barron [Barron, 1993] deserves a large portion of the credit for the success of *packetman*. His *simple* addition of the NFS protocol built the underlying functionality that most protocols could be built on top of.

Another tool to come from the Netman group is *loadman*. This simple tool is an indicator of network load (bandwidth usage). *loadman* was developed as an implementation of the load-ring algorithm developed by Jeffrey Mogul [Mogul, 1990] [Mercer et al, 1994] for network load monitoring.

A new tool currently under development is *chargeman*. This is a network charging tool designed specifically to cater for the charging to be introduced by AARNET on January 1, 1995. While this tool is specific to Australian sites, the basis for it is packet charging based on source and destination locations.

2.0 Aims of this project

The main aim of this project was to extend GeoTraceMan. This extension was twofold:

1. To prepare GeoTraceMan for release.
2. To add a three dimensional globe option along with the flat map view of the world.

The first item on the list seemed simple. In reality, this turned out to be more difficult than expected. Full coverage of the changes made will be presented later.

The three dimensional globe option was first envisaged by Darren Brown, the previous developer of GeoTraceMan, as a possible future addition. This too became more difficult than it first seemed and will be covered later in this report.

A third aim that arose early in the project was to find a method of resolving a hosts geographical location. The current method of achieving this was to store the location of carefully selected sites in local cache files. It was decided that the Domain Naming System (DNS) [Mockapetris, 1987 (RFC1034 & RFC1035)] would be the best vehicle for achieving this. DNS is already implemented extensively throughout the Internet. The problems encountered and their solutions will be discussed in detail in section 7.0 Domain Naming System Resolution.

3.0 Development

The development of GeoTraceMan was performed in an *almost* planned manner, rather than the *ad hoc* method that befall most undergraduate projects. As such, many decisions were made early in the project that would normally be left until encountered. This aided in keeping the project on track as clearly defined, achievable, goals were set.

3.1 Development Platform

The majority of the GeoTraceMan development was performed on SUN workstations of various sizes. Throughout development, cross platform migration was considered extremely important, hence porting GeoTraceMan to other platforms should not require a great deal of effort. SUN IPX, IPC, SLC and ancient 4/110 workstations were used as the main development machines due to their availability. The SLC was mainly used for its monochrome display.

Other platforms that some stages of development were performed on include Silicon Graphics Indigo and Indy workstations and DEC 5000/240 and 5000/25 workstations. The multitude of architectures posed some previously unconsidered problems.

3.2 Development Considerations

For GeoTraceMan to be useful, it would have to be able to run on numerous different architectures, be able to display on the three main types of monitor (colour, gray scale and monochrome) and be able to update its display fast. These three problems were considered the most important aspects of the development.

3.2.1 Architectures

As mentioned above, the development of GeoTraceMan took place with multiple architectures in mind. This was catered for through the use of the UNIX *imake* utility. This utility allows machine specific `makefile`s to be created for the different architectures. By using the *xmkmf* front to the *imake* facility, a `makefile` could be generated that was not only architecture specific, but machine (software location) specific as well. This facility provided the locations of the X libraries and header files in such a way as to avoid having to exactly specify their locations then expecting them to be in the same location on all machines. Once the specific `makefile` is generated, GeoTraceMan is compiled using the standard *make* utility.

The fonts used in GeoTraceMan were the Hershey fonts [Hook, 1992]. These are the same fonts the rest of the Netman suite uses. This allowed users with other Netman tools to install GeoTraceMan with minimal adjustment to their system. By using a secondary font (ie. a non-machine specific font) it is less likely there will be problems associated with fonts not being available.

The choice of compiler was also based on the multiple architecture view. The standard SUN *cc* compiler was used as it is a solid implementation of a Kernighan and Ritchie C compiler [Kernighan & Ritchie, 1978]. Many platforms still do not have a bug free ANSI C [Kernighan & Ritchie, 1988] compiler, hence its lack of use. The GNU *gcc* ANSI C compiler was another possible choice, but due to some *unique* bugs while debugging code compiled with this compiler, it was not used.

Other platforms used for development include the afore mentioned Silicon Graphics and DEC workstations. As the byte ordering is different on DEC workstations to that of SGI and SUN workstations, the map data was required to be converted into the appropriate *endian* before it could be used. Rather than use two different, yet essentially identical data sets, the conversion was performed at run-time. Any data that required conversion was passed through a simple filter that performed the necessary byte swapping. It was felt the best method of achieving byte ordering independence was to use network byte ordering. Hence, all data that was a potential byte ordering problem was filtered through the byte ordering network functions, available in the C standard libraries.

3.2.2 Monitor Types

The three main types of monitors on modern workstations are:

- Full colour
- Gray scale
- Monochrome

GeoTraceMan was developed on these three monitor types specifically to allow it to be run on all of them. GeoTraceMan was initially written for a colour display. This proved to be quite acceptable for a gray scale display once the colour selection was arranged correctly. When first run on a monochrome display, most of the widgets became invisible due to the background and foreground colours both mapping to the same colour (black or white).

To overcome this problem, a check for the type of display was incorporated. This check was already performed by GeoTraceMan, it was simply not utilised. If a monochrome display was detected, a separate routine was called to initialise the required widgets as monochrome rather than colour. The majority of the application was acceptable in both colour and monochrome.

As there was no consistency to the remapping of colour widgets as monochrome, the routine used to perform this is a large routine of very simple, repetitive calls. It was not possible to initialise all of the widgets to, for example, black background and white foreground as this provided some *interesting* combinations. A better way to perform this ugly, but necessary task was not found, nor was one sort with any diligence.

3.2.3 Speed of Display

Speed of execution is a problem that most programmers face at some stage during project. Usually improvements in speed can only be achieved through a trade-off with either size or functionality.

The GeoTraceMan project was fortunate in this area as an increase in the functionality did not result in too great a decrease in the speed of execution. The slowest part of GeoTraceMan remained the reading and displaying of the map data. Many small changes were made that allowed the speed of display to be improved fractionally. Nothing could be done to decrease the amount of time required to read in the large amount of data.

The zoom functions were almost entirely redesigned to increase their speed, but once again, they too were limited by the speed at which such a large amount of data could be processed.

A particular concern over speed was brought to the fore when considering the three dimensional map display. It was felt the additional processing of the map data would result in a program that was too slow to be of any use on all but the fastest workstations. When this was actually coded, the speed of display was slower than the equivalent two dimensional display, as expected. It was not, however as slow as first feared. This resulted from the conversion process. When using globes, half of the data can be clipped, resulting in the need to process only half of the original data set.

3.3 Source Code Control

As with Darren Brown's original GeoTraceMan, this version utilised *RCS* (Revision Control System) [Tichy, 1991] to control different versions of the source code. This allowed versions to be kept whenever major changes were to take place.

The *SCCS* source code control system was another possible choice for revision control. It was not used as *RCS* was already in place with the existing GeoTraceMan code.

Source code control was deemed a waste of time and effort when first used. It required valuable time to learn and provided no immediate returns. It was utilised more fully as the need to backtrack through versions arose.

4.0 Background

As previously stated, Darren Brown was the initial developer of GeoTraceMan. As such, he performed the majority of the background research. His project report on GeoTraceMan is available for referral [Brown, 1993].

4.1 The Internet

The Internet is a vast community of computer networks linked together. The fact that no single governing body is required indicates the monumentous co-operation the Internet community has achieved. The main reason the Internet works as it does is a simple fact:- no *single* governing body runs or owns the Internet, everyone co-operates.

There is, however, a single body that administers the Internet. "The Internet is loosely administered by the Network Information Center (NIC) at the Stanford Research Institute (SRI), where documents and standards are archived, IP addresses are allocated, and network and domain names are registered." [Nemeth et al, 1989]

Even though a central administration for the Internet exists, there are no *Internet police* to enforce the policy of the NIC. This highlights the understanding, trust and co-operation of the Internet community even more.

4.1.1 Protocols Used In the Internet

For any network to operate successfully, all connected hosts must communicate using the same set of protocols. The Internet uses the TCP/IP (Transmission Control Protocol/Internet Protocol) suite of protocols. These protocols provide a standard way for all to communicate, even though they may not be directly connected.

The Internet uses numerous protocols to transmit control messages as well as data. It is a combination of data and control protocols that are utilised in GeoTraceMan.

4.1.1.1 Internet Protocol

The Internet Protocol (IP) is the main protocol used throughout the Internet. It acts as the outer shell for all other protocols. As stated in Comer (1991):

> The fundamental Internet service consists of a packet delivery system. Technically, the service is defined as an unreliable, best-effort, connectionless packet delivery system, analogous to the service provided by network hardware that operates on a best-effort paradigm. The service is called *unreliable* because delivery is not guaranteed. The packet may be lost, duplicated, delayed, or delivered out of order, but the service will not detect such conditions, nor will it inform the sender or receiver. The service

is called *connectionless* because each packet is treated independently from all others. A sequence of packets sent from one machine to another may travel over different paths, or some may be lost while others are delivered. Finally, the service is said to use *best-effort delivery* because the Internet software does not discard packets capriciously; unreliability arises only when resources are exhausted or underlying networks fail.

Packets that do not arrive at their destination host do not remain in the Internet forever. Each IP packet has a time to live (ttl) field that determines how long the packet has before a gateway can remove it from the network. When each packet arrives at a gateway the value of this field is decremented and checked. If it is zero, the packet is removed and an error message is sent to the source of the packet. If not, the field is theoretically decremented by the time that the datagram resides within a gateway. The packet is routed to the next gateway.

Each host in the Internet is identified by an IP number. An IP number is a 32 bit integer which uniquely identifies the network connection. IP numbers are often represented in dot notation. This involves dividing the IP number into its four octets, writing them as decimal numbers and separating them with decimal points. For example, the IP number of one of the GeoTraceMan development machine is 134.7.1.21. This number also maps onto the IP name of `beavis.cs.curtin.edu.au`. The IP number is normally one of three classes:

0	netid (7 bits)	hostid (24 bits)

Figure 1: Class A IP Address

1 0	netid (14 bits)	hostid (16 bits)

Figure 2: Class B IP Address

1 1 0	netid (21 bits)	hostid (8 bits)

Figure 3: Class C IP Address

There are two more classes of IP numbers (multicast and reserved), but the above three are the most common. Some reserved IP numbers are those with 255 or 0 as any part of it. These numbers are used for broadcast and masking.

4.1.1.2 Internet Control Message Protocol

Internet Control Message Protocol (ICMP) packets are control packets used in the Internet. This protocol "allows gateways to send error or control messages to other gateways or hosts" [Comer, 1991]. ICMP was originally designed for gateway to gateway control, but it can also be used to transmit such control messages between hosts.

ICMP packets are used to retrieve GeoTraceMan's trace data from the destination host. The data returned by these packets provides us with a method of calculating the delay statistics and the path from the source to the destination host.

Two ICMP packets that are of particular interest to GeoTraceMan are those containing the *Destination Unreachable* and the *Time Exceeded for Datagram* messages. These are used to calculate the delay and route statistics that GeoTraceMan provides. The statistics are calculated as the packets are received, hence a real-time comprehension of the delays that can occur in a packet switched network may be realised.

Even though ICMP packets are control packets, they are still subject to the same connectionless, unreliable delivery from which all Internet traffic suffers.

4.1.1.3 User Datagram Protocol

User Datagram Protocol (UDP) is a widely used protocol in the Internet. Comer (1991) susinctly summerised UDP:

> The User Datagram Protocol provides unreliable connectionless delivery service using IP to transport messages between machines. It adds the ability to distinguish among multiple destinations within a given host computer.

UDP is the primary mechanism that allows an application program to send datagrams to other application programs. UDP achieves this through the use of protocol ports. Each UDP packet contains both a source and destination port. UDP provides no error control, error correction or flow control. It is assumed that the application program will handle this. As UDP is encapsulated within an IP packet, it suffers from the same problems as IP.

Each UDP packet is usually treated as a totally self-contained entity. It can not be considered to be a part of a stream of packets as there is no guarantee the remaining packets will arrive correctly.

The main advantage of UDP is its small, eight byte, header. This makes for extremely efficient data transfer. The other major transport protocol in the Internet (TCP) has a header size of at least 24 bytes (depending upon options) and provides reliable end-to-end communications with flow control and error detection.

4.1.2 Traceroute

Traceroute [Jacobson, 1989] performs the task of tracing a route between hosts. This is achieved by attempting to force a ICMP TIMEEXCEEDED response from each gateway on the path to the destination host. The ICMP TIMEEXCEEDED message is produced when a gateway examines a packet with a time to live of zero.

Traceroute utilises a UDP packet as a probe datagram. The datagram is sent to port 33434 which is, hopefully, unused. Traceroute sends a probe packet to the first gateway with a time to live field of one. The gateway decrements this, and finding it to be zero returns a ICMP TIMEEXCEEDED message to the source. The second gateway is sent a probe packet with a time to live field of two. The time to live field is decremented by the first gateway, then the second gateway, which discovers this field to be zero. Once again the ICMP TIMEEXCEEDED message is returned to the source. This continues until a ICMP PORTUNREACHABLE message is returned (no process should be attached to this port number, hence no processing of the datagram should take place). This message indicates that the destination host has been reached.

From the above brief description, it can be seen that traceroute is a usful tool for tracking routes packets take to some, abitary, destination. Traceroute can also be used to indicate where gateways are potentially dropping packets.

At the time of writing this report, a new Traceroute was outlined in an RFC [Malkin, 1993]. This new version is more efficient than the current implementation. It generates only $n + 1$ packets rather than $2n$ packets, where n is the number of hops between the source and destination. Extra functionality is also provided in the new version which uses an ICMP message designed for this purpose.

4.2 Maps

The map data for GeoTraceMan was extracted from the CIA World Map Databank [Dellinger, 1989] and manipulated into its current form. The data was reduced from over 20 megabytes down to 6 megabytes via a specialised reduction created for this exact purpose. A smaller dataset would be preferable, but the smaller the size of the data, the less detail available. For a complete breakdown of how the maps were extracted and the problems encountered in this process, please refer to [Brown, 1993].

The possibility of using other data sources arose every time a problem with the current data source appeared. Upon review, this was not really an option due to the original difficulty in converting the current data set.

For example, when considering the quad tree approach, the original data was once again consulted. It was felt the best result would be attained by extracting from the largest data set possible, then reducing, as needed.

4.2.1 Problems With the Map Data

All data suffers from inaccuracy at some stage. The modified CIA World Map Databank was no different. The main problem encountered was the lack of joining vectors. Each run of vectors would join as expected, but there were always gaps, sometimes indistinguishable, sometimes quite large, between the runs of vectors. These gaps were introduced during the dataset reduction. Any vectors that crossed regions boundaries (see [Brown, 1993]) were not drawn. The difficulty that arose from this problem was an inability to produce a map with filled land masses. The gaps between the vector runs were inconsistent enough to preclude the use of a threshold value in a filling routine. In some situations, the dimensions of a land mass was less than the dimensions of a break in the vectors. A solution to this problem would be to rerun the map reduction process using a new algorithm that compensated for this via an edge intersection test. It will be addressed in the future.

The projection of the data provided another complication to this project. The original report stated the data was in Mercator's projection. The data itself contained statements which supported this. After numerous attempts at processing the data using this assumption, the possibility that the data may not be in this projection arose. Attempts were then made to process the data based on a cylindrical projection. This also proved unsuccessful. At the suggestion of Dr Geoff West, a *raw* processing attempt was made. This attempt was based on the assumption that the y coordinate was merely a deprojection of latitudes onto a cartesian plane. This was found to be the method used to produce the original data, hence it successfully deprojected the data.

4.3 X Windows

The X windowing system was the graphics platform utilised by GeoTraceMan. X is a networked graphical user interface which compliments the requirements of GeoTraceMan. X is the most common graphical user interface found on UNIX workstations, which is the target machine of GeoTraceMan.

"X is a distributed, intelligent, device independent, operating system independent, hardware independent windowing system."

[Personal communication with Andrew Marriot, Senior Graphics Lecturer, Curtin University of Technology]

4.3.1 Graphics Library

The VOGL (Very Ordinary Graphics Library) [Echidna, 1991] is used in GeoTraceMan in an attempt to simplify the coding process. This library is roughly based on the Silicon Graphics *GL* library [Silicon Graphics, 1990]. The library is public domain and has some public domain type bugs. The main bug that affected GeoTraceMan was VOGL's inability to handle more than one output window. The map drawing canvas was controlled solely by VOGL. The panner was to be implemented with VOGL controlling it, however VOGL's inability to handle multiple drawables would require standard X calls to be used.

By using an external library such as VOGL, the need for machine specific libraries is decreased. If *GL* was used for the Silicon Graphics version of GeoTraceMan, a total rewrite would be required for other platforms that did not support this library.

Rather than using an external graphics library such as VOGL, standard Xlib or Xt calls could have, potentially, been used. This would produce faster code, with no portability problems. The negative aspect of using these calls is the amount of extra code required to produce the same results. Once again, the trade-off between speed and functionality appears.

5.0 Quad Trees

Early in the development of the extensions to GeoTraceMan, the use of quad trees as a method for map storage was considered. It was decided that quad trees were not the solution required for the reasons outlined below.

5.1 Quad Tree Basics

Quad trees are a recursively decomposable data structure. This allows quad trees to store data at an increasing level of accuracy as the depth and size of the quad tree grows. This data can then be accessed at the appropriate level of accuracy when required. Figures 4 to 7 depict a region that is to be converted into a quad tree, the binary array representation of this area, the maximal blocks for the same area and finally the quad tree that would be produced. As can be seen, representing the original data in a binary array would take 64 bytes of storage. To store this data in a quad tree would require 96 bytes (assuming pointers and integers are both four bytes). This is a considerably larger dataset. The efficiency arises when displaying the data. To display the binary array requires the drawing of 26 rectangular regions. Using the quad tree, this can be displayed using 11 rectangular regions.

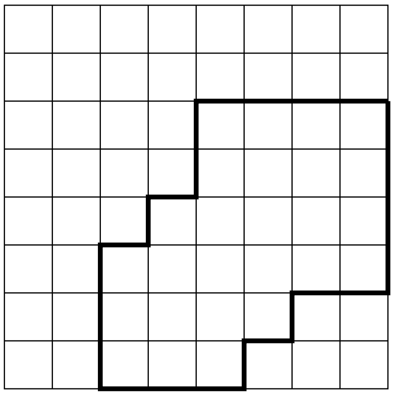


Figure 4: Original Region

The structure itself is based on divide and conquer problem solving methodologies. The natural division of the tree clearly shows this. Each branch of the tree can be subdivided, traversed and *conquered* with no regard for the branches above. Recursion is an excellent method for tree traversal. Each node with sub-nodes can be considered a parent can be treated as a separate tree, whether this node is the lowest level node with only four children or if it is a root node.

Quad trees are an excellent data structure when the data is to be viewed at multiple resolutions. If the resolution required is less than the overall depth of the tree, a large time savings can be made by stopping a tree search at the appropriate depth rather than recursing through the entire tree. As the majority of detail (and bulk) appears at

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0

Figure 5: Binary Array

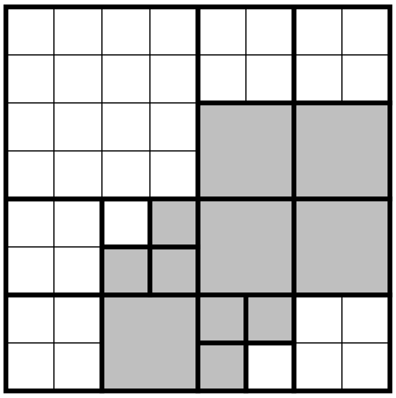


Figure 6: Maximal Blocks

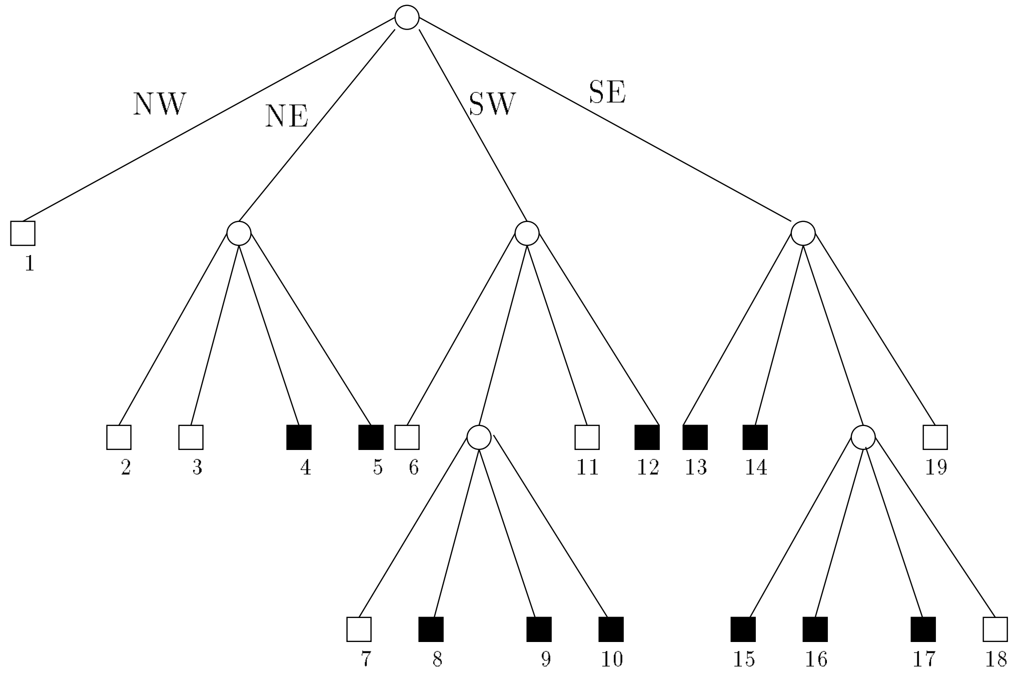


Figure 7: Quad Tree

the lower levels of the tree. It can be seen that considerable time savings are possible by limiting the depth of a search.

Clipping is also extremely efficient with quad trees. The branches of the tree not within the area to be displayed can be quickly pruned, hence never searched. This provides excellent time savings when searching a quad tree that has a know clipping boundary. If the data we are dealing with lies entirely within one quadrant at the highest level, three quarters of the tree can instantly be disregarded (depending upon the balance of the tree). By apply this logic to lower levels of the tree, large amounts of data can be ignored simply and efficiently when not required.

5.2 Quad Trees in GeoTraceMan

GeoTraceMan does not implement quad trees. Quad trees were considered a necessity for the display speed requirements of GeoTraceMan. After a lengthy deliberation period, the thought of using quad trees was replaced with the concept of maintaining the existing data set but manipulating it more efficiently.

The original map data extraction process took approximately four weeks. If the data was to be converted into quad tree data, it could be expected that approximately the same time period would have to be dedicated to producing such a tree. The original data was provided in cylindrical projection form. Through some simple logic, it was discovered while displaying the three dimensional globe only half of the data was used. This produced a time saving of approximately 50% (actually less than 50% due to extra calculations, but savings due to clipping were still significant).

The savings discovered in the current data set and the expense of producing a new set of maps resulted in a strong desire to keep the current data set. Although this decision did nothing to improve the performance of the flat map version of GeoTraceMan, it did allow more time to be dedicated to the two primary goals of the project: production of the three dimensional globe and preparing GeoTraceMan for release.

6.0 The Globe

A major aim of this project was to implement a three dimensional globe. This globe shows traces as arcs looping over the map between sites. The globe can be rotated and zoomed using exactly the same method as the flat map.

6.1 Globe Map Data

The data used in the globe representation of the earth is exactly the same as the flat map. This was done to decrease the amount of *almost* identical data required to run the program. Creating a different set of data to be used with the three dimensional globe would require either the modification of the current data set or the creation of a totally new version of the data, potentially as quad trees. It was decided that this modification could be performed in *almost* real time.

Even in the current flat map version of GeoTraceMan, problems existed with the map handling functions. There is a memory fragmentation problem that arises with the reading of the map data and is generally not as efficient as would be like. This, and other problems, will be discussed later, in section 9.0 Release of GeoTraceMan.

6.2 Globe Drawing

The drawing of the globe was done in an extremely similar method to the flat map. The main differences were the addition of extra clipping and a filtering function to provide the three dimensional look.

6.2.1 Clipping

Clipping was introduced to increase the speed at which data could be processed and displayed. It is possible to perform a 50% clipping. No matter which angle or rotation the globe is viewed at, half of the globe is obscured. The clipping extents are defined at execution time and are dependent on the current viewport. The clipping boundaries are always 90 degrees in all directions from the centre of the image.

Both end points of all vectors were required to be passed through the clipping routine. This produced a result that was expected, but not perfect. If one end point of the vector passed the clipping test and was to be drawn as a visible point and the other end point of the vector did not, the entire vector was not drawn. This is a less than ideal solution, but without the inclusion of a line intersection test this could not be overcome. The extra time taken to compute the clipping was much less than the time required to draw all vectors then perform the equivalent clipping by hidden line removal.

The clipping used had the effect shown in figure 8: Clipping a Two Dimensional Map. This shows the clipping applied to the flat map rather than to the three dimensional map for simpler understanding. As can be seen through comparison to figure 9: The Flat Map, the amount of saving is exactly 50%.

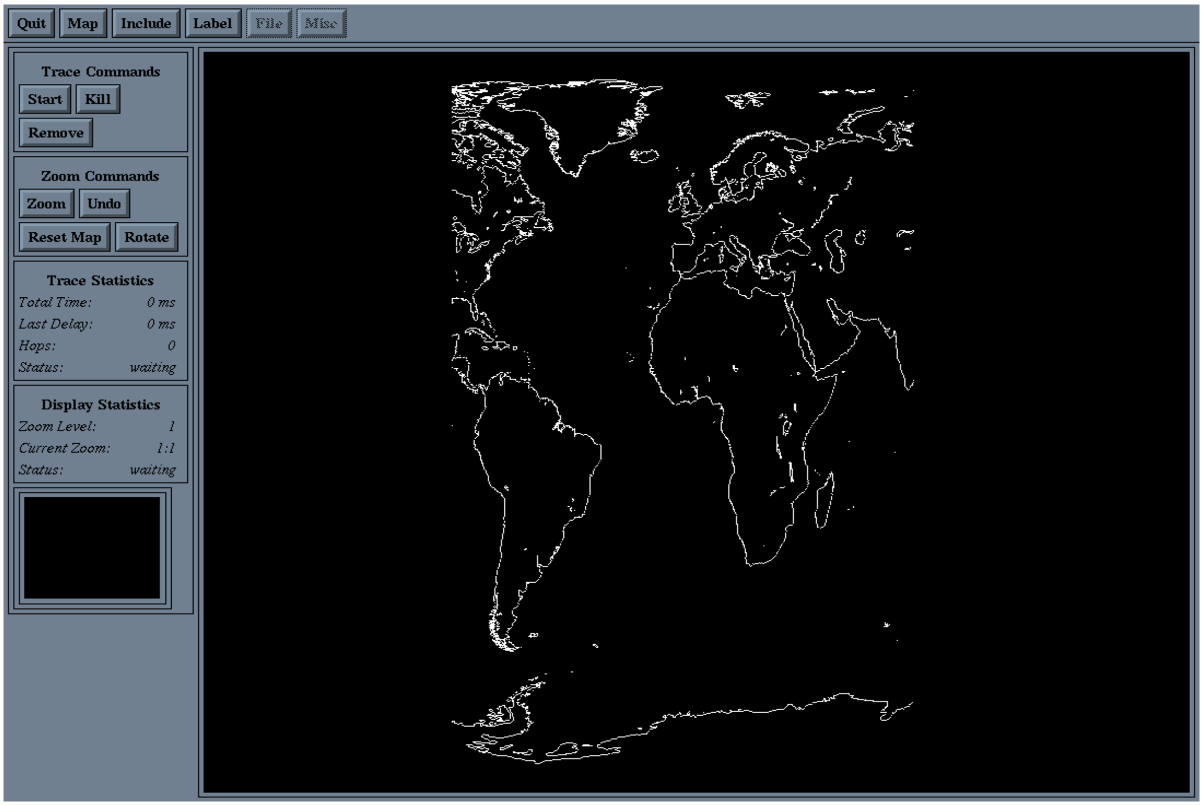


Figure 8: Clipping a Two Dimensional Map

6.2.2 Three Dimensional Drawing

The drawing of the GeoTraceMan map data in three dimensions proved to be difficult. Many attempts were made at a solution to this problem. Some of the attempts, both successes and failures, are outlined in this section.

6.2.2.1 The First Attempt - Warping

The first attempt at producing a three dimensional solution was through warping. The concept behind this was a simple non-linear function applied to all visible points. The mathematical formulae for this attempt was provided by Glen Downton, an honours student at Curtin University of Technology. Glen was working on a method of projecting a two dimensional object onto the back half of a sphere. This sounded

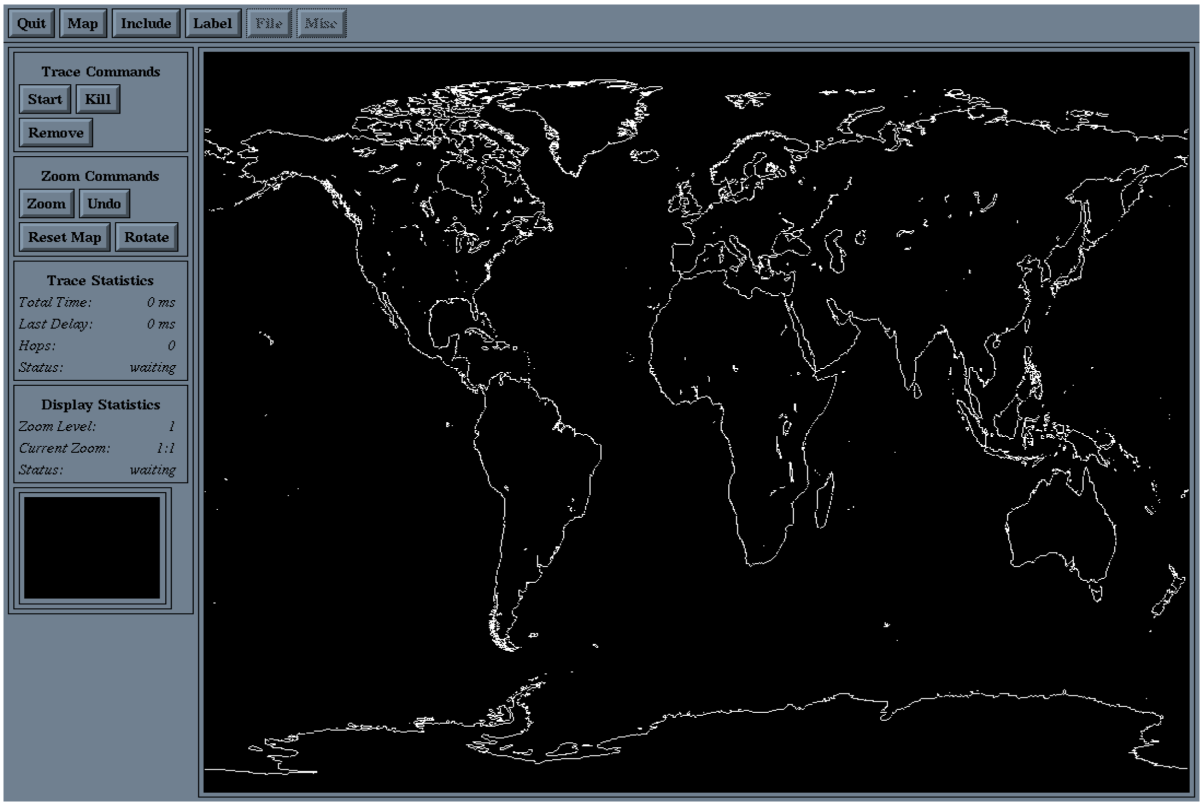


Figure 9: The Flat Map

familiar to the problem under study here. After producing code that encapsulated the mathematics produced by Glen, it was clear that this was not the required solution.

One advantage of this method was its speed. It was based entirely on ratios rather than the maths library. Due to this, a large amount of time was spent developing this method. As can be seen in figure 10: Warping to Three Dimensions, the result was close, but not close enough.

The implementation of this method highlighted some further problems with VOGL. The origin that VOGL used for all of its drawing is based in the lower left corner of the drawing pane. This method of warping expected the origin to be in the more natural place - the centre of the data. The simplest method to overcome this was to *shift* the data. The data, rather than the VOGL origin, was shifted as VOGL would still need to use the old origin to draw the data. The computation time required to use the *pushmatrix* and *popmatrix* calls of VOGL was unacceptably large for the amount of functionality it offered. After the conversion was performed, the data was shifted back and drawn.

To operate as a true filter function, shifting the data was logically *more correct* than shifting the VOGL origin. A filter function should be totally self-contained. It should be highly modular, accept data and produce results without requiring extensive setup before use. Shifting the data aided in producing such a filter function.

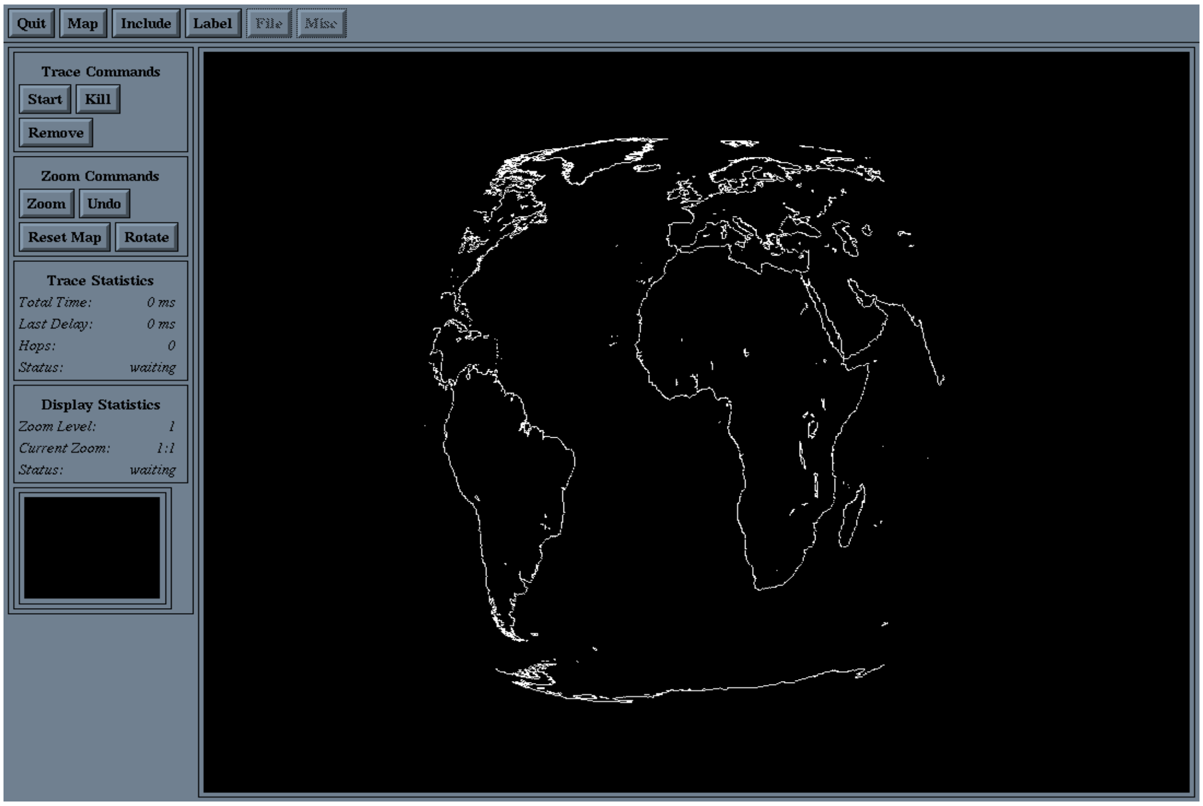


Figure 10: Warping to Three Dimensions

6.2.2.2 The Second Attempt - Reverse Mercator's

Based on the assumption that the data was in Mercator's projection, the next attempt at three dimensional displaying of the data was to perform a reverse Mercator projection. This required an understanding of how Mercator projection was produced. The most useful reference found for this was Elements of Cartography [Robinson et al, 1984]. The required mathematical formulae and C code were provided for the projection from latitude and longitude measures into the Mercator projection. It was a trivial task to reverse this mathematics to produce the desired effect.

As previously stated, data that was thought to be in Mercator projection was actually in a raw mapping from latitudes and longitudes. This attempt at producing the three dimensional globe proved the data was without a doubt not Mercator projection. Applying the mathematics derived from Mercator's original equations to the map data produced unexpected results. No data was displayed at all. When data was fed through the conversion function, the data produced was out of range.

The formula below outlines the process of converting a flat map Mercator projection into a three dimensional globe. As can be seen, it relies on the original data as well as the latitude and longitude of a position. This produces the three dimensional view of the globe.

```
x = x0 + scale * (lon - lon0)
y = y0 + scale * ln(tan(π/4 + lat/2))
```

6.2.2.4 The Third Attempt - The Solution

The basic algorithm that was implemented in GeoTraceMan to produce the three dimensional look for the globe is outlined below.

```
for each point displayable do
        alpha = point.x * mapwidth * 2 * π
        beta = point.y * mapwidth/2 * π/2
        point.x = radius * cos(beta) * sin(alpha) + mapwidth/2
        point.y = radius * sin(beta) + mapwidth/2
endfor
```

This algorithm performs two basic steps. It converts the two dimensional data into a three dimensional format and then places this three dimensional data on a two dimensional plane. The three dimensional data is temporarily stored as two angles (*alpha* and *beta*). These angles indicate the angle around the equator (longitude) and the angle towards either pole (latitude). As was discovered in previous failed attempts, the data was merely a mapping from these angle coordinates onto a cartesian plane, hence this remapping worked correctly. The displaying of this data on a two dimensional plane works by using the above simple conversion based on the trigonometric unit circle. The radius of the globe the data is to be displayed on is used to scale the data.

The efficiency of any algorithm that accesses the maths library must be studied due to the inability of most computers to perform fast floating point arithmetic. As can be seen from the above algorithm, efficiency has been considered. There are only three calls to the maths library, none of which could be avoided, nor simplified. A potential method to avoid using the maths library entirely would be to implement the *sin* and *cos* functions using a simplified Taylor's series, or possibly even a look-up table. This was avoided in order to maintain modularity and clarity. Clipping is performed before data is converted, hence the number of points requiring processing is decreased, further increasing the speed.

The results from applying the above algorithm can be seen in figure 12: Three Dimensional Solution.

6.2.2.3 A Bonus - The Mathematician's

At the suggestion of my supervisors, the advice of the School of Mathematics and Statistics was sort. The solution that they produced was not the required globe, but it was a valid projection. Mollweide's Projection [Bartholomew, 1978] may be kept as an

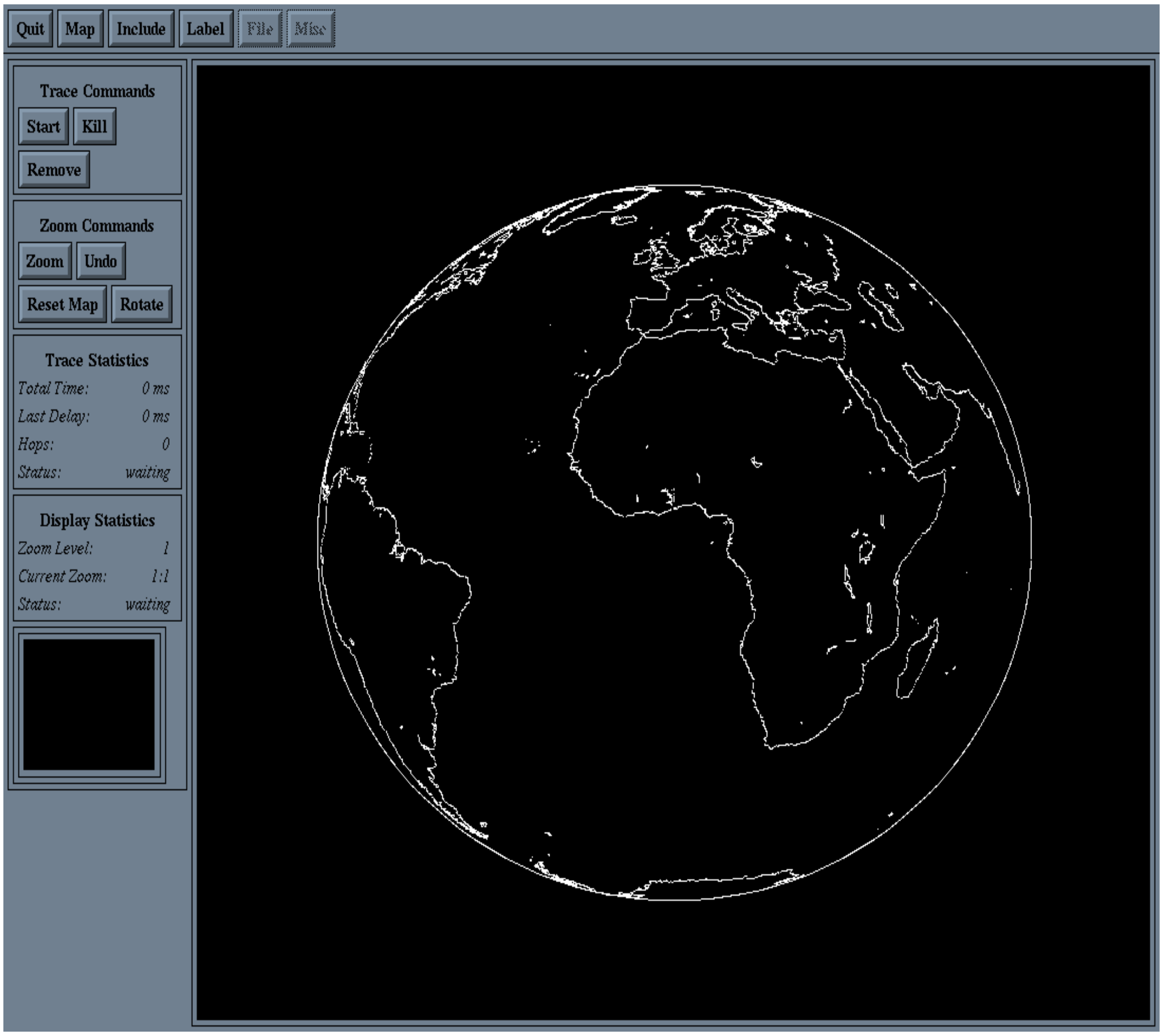


Figure 12: Three Dimensional Solution

option. This projection displays the entire world on the front of a three dimensional ellipse. This projection was seen as a means of displaying a trace in a three dimensions without having to rotate the globe in order to see the entire trace.

The algorithm used to produce this projection was:

```
for all points do
       alpha = point.x * mapwidth * π
       beta = 2 * arctan(point.y * mapwidth/2)
       point.x = radius * cos(beta) * sin(alpha) + mapwidth/2
       point.y = radius * sin(beta) + mapwidth/2
endfor
```

This algorithm is not as efficient as would be liked. Every point to be displayed (which with this algorithm is every single point) must be passed through three calls to the maths library. It is felt the decrease in the speed of this projection is acceptable due to decrease in the need for rotation. Figure 13: Mollweide's Projection shows the results of applying the above algorithm to the map data.

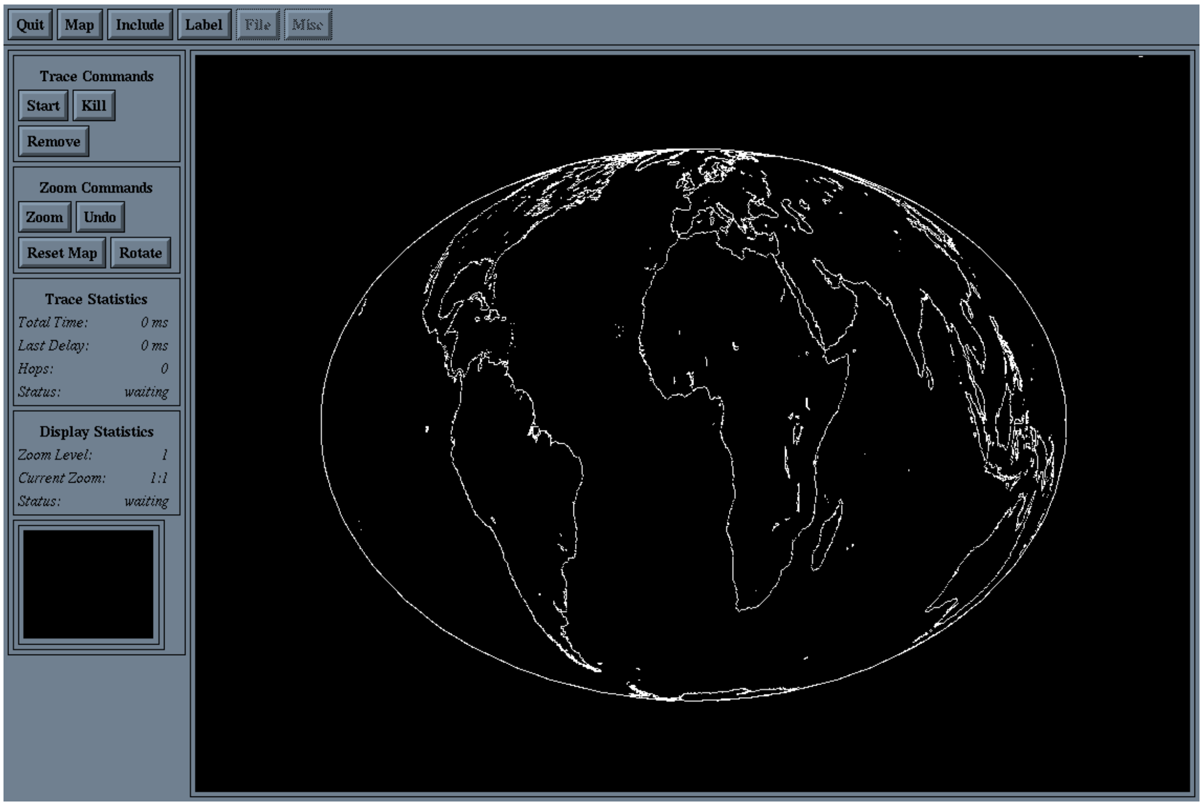


Figure 13: Mollweide's Projection

The advantages this projection exhibits are equi-distance along the equator and the maintenance of equal area, despite the deformation of shape near the poles. The main disadvantage from which it suffers is the deformation.

6.3 Trace Displaying

The purpose of GeoTraceMan is to display traces on a map of the world. In the flat map version, these trace line were merely straight lines connecting two points (sites) on the map. In the three dimensional version, arcs were to be used rather than straight lines. Figure 14: Two Dimensional Trace shows a trace on the two dimensional map from Curtin University, Perth, Western Australia (beavis.cs.curtin.edu.au) to a site in Helsinki, Finland (a part of EUNET-X25).

As can be seen from the above, straight lines are an excellent two dimensional method of depicting a trace. These lines are drawn directly between two end points. The path from Melbourne, Australia to Palo Alto, California is not, as the map indicates, over

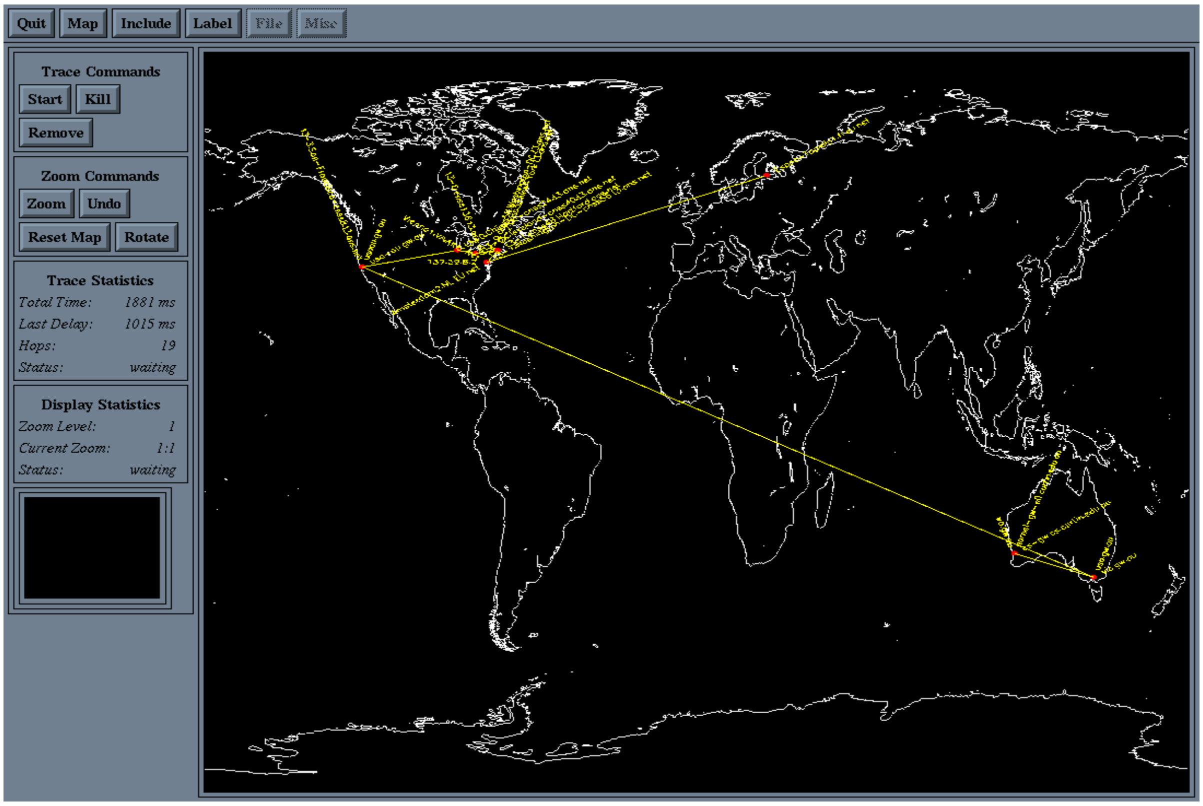


Figure 14: Two Dimensional Trace

Africa. It actually passes near Hawaii as Figure 15: A Rotated Two Dimensional Trace shows.

6.3.1 Three Dimensional Trace Displaying

As stated previously, the three dimensional trace was to be displayed using arcs instead of straight lines. These arcs were to extend beyond the globe in an attempt to make them look as though they were above the surface, rather than simply running along it. This should have provided a method for a trace line to start on the back half of the globe and finish on the front half of the globe. When the trace becomes visible, it should appear as though it originated on the other side of the globe rather than on the edge of the globe.

Difficulties arose in the attempt to produce a working model of the above theory. Using the same data conversion routine as previously outlined, if one point of the line was not displayable (on the other side of the globe), the entire line would not be displayed. Since only the end points were converted, the lines themselves remained straight. This was not acceptable. An example of a trace on a three dimensional globe is shown in figure 15: Straight Line Trace in Three Dimensions.

Rather than use straight lines, an attempt to use arcs was made. This attempt was

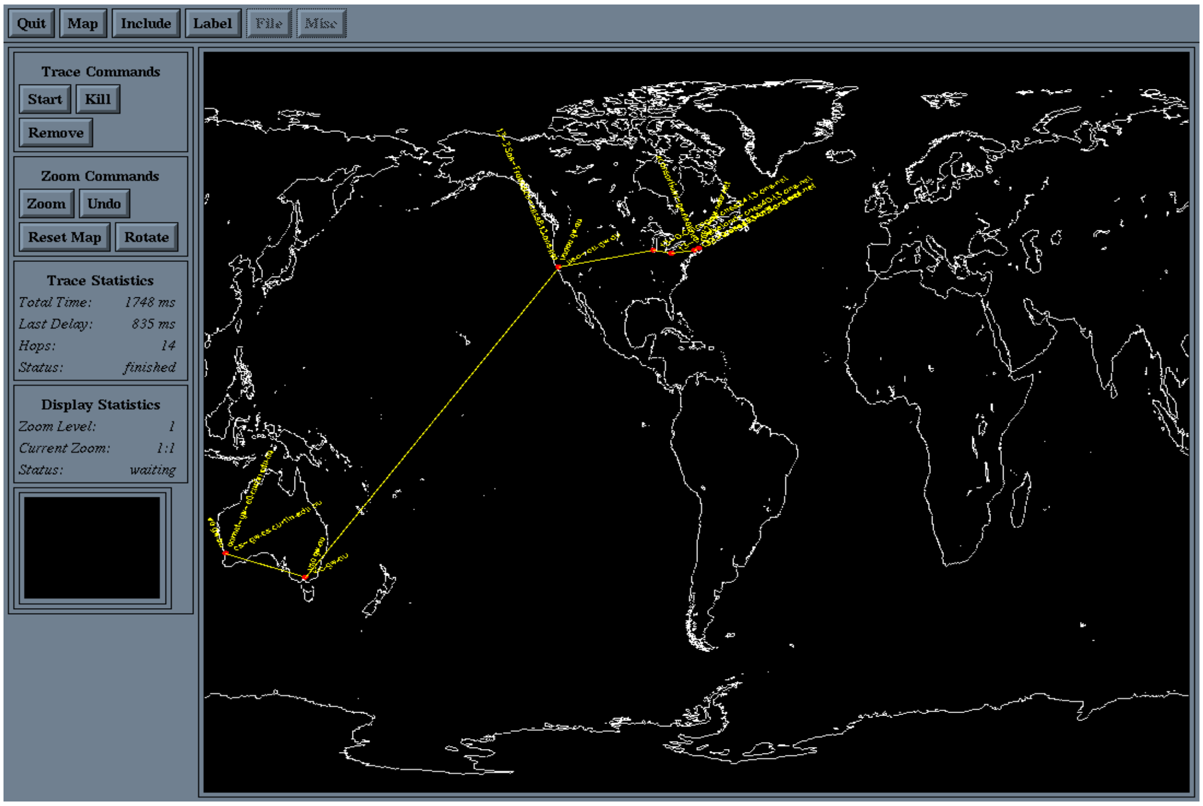


Figure 14: A Rotated Two Dimensional Trace

only partially attempted due to problems with drawing two dimensional arcs in three dimensional space. An arc at the equator would have to be flatter than an arc at the tropics, hence rotation of the arc in three dimensions would be required. Another problem that arose, as it did with most of the other methods, was hidden line removal. As an arc crossed from the back half of the globe to the front half, the trace line would have to become visible. VOGL provided no method of achieving this, so the idea of using arcs was discarded.

Another potential method of achieving the three dimensional look was to use B-Splines. This was not studied in depth as the author has no experience with their use. The problem of hidden line removal would still exist if B-Splines were used, although it appears as though it would be easier to cater for.

Dividing the line between two sites into small line segments was the third and final method studied to display a three dimensional trace. This process involved dividing the line into small segments. Both end points of the smaller line segments are processed by the two to three dimensional converter, producing a staggered effect through the line. Hidden line testing is now made easier, but producing a loop at the edge of the globe was not addressed. In concept, this method should have produced a usable three dimensional arc, in reality, it would produce this arc on the surface of the globe rather than *above* it. The production of this line would be extremely slow due to the large number of two dimension to three dimension conversions required ($n + 1$, where n is

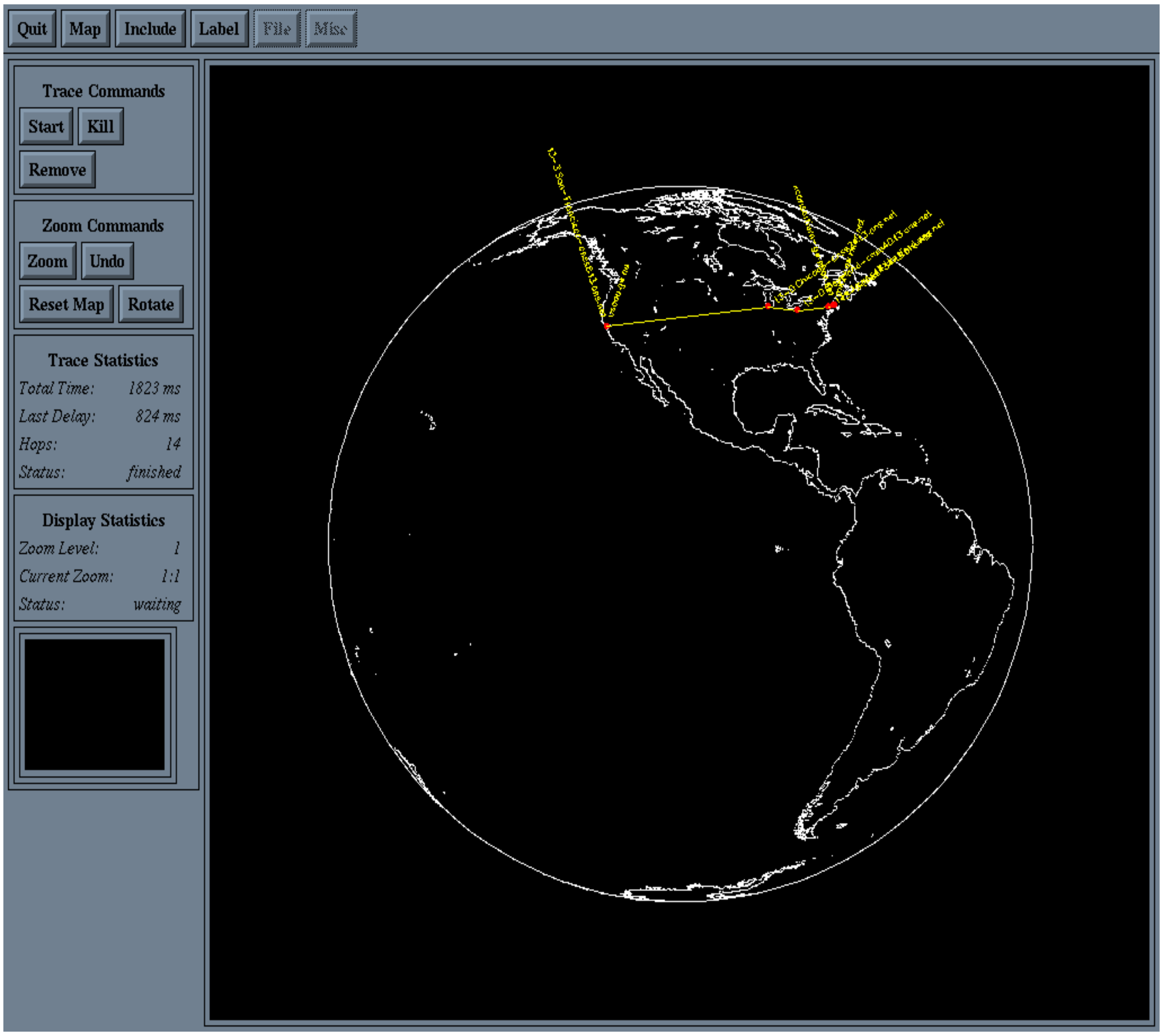


Figure 15: Straight Line Trace in Three Dimensions

the number of line segments).

6.4 Rotation

Rotation was previously only handled in the left and right directions on the flat map (east/west direction). This rotation was performed in 36 degree jumps, regardless of the current zoom. This has been maintained for this release, with plans to modify in the near future. Options considered included the ability to rotate as a percentage of the visible map and the ability to rotate the map in a north and south direction when possible.

Rotating the map as a percentage should have been simple to implement, but the design of the original data structures made this difficult. Due to time running short, this was not modified. This will be dealt with in the future, possibly by rewriting the structures used (mainly for other purposes).

The ability to rotate in the north and south direction would be a valuable addition. The three dimensional map should be able to rotate in the north and south direction at any stage. The flat map should only be able to rotate in a north and south direction when it is zoomed and the rotation would not extend the map beyond the poles (there can be no wrap-around in the north and south direction in a two dimensional map).

Another extension to the rotation functions of GeoTraceMan would be the ability to rotate the axes of the globe. To provide full freedom of rotation this should be included.

6.5 Zooming

Zooming is an important process in the displaying of any map related information. The ultimate example of zooming in maps is an atlas. Each section of the world is covered numerous times in an atlas at many different zooms showing different details. GeoTraceMan attempts to emulate the *natural* method of zooming, ie. showing more detail as the zoom increases.

The zooming function was implemented via a bounding box which can be drawn on the map canvas at any stage. This box is maintained as a square to avoid aspect ratio problems. The zoom is initiated when the user clicks and holds the mouse button when over the canvas. Once the box size is specified by dragging the mouse to the appropriate dimensions, the mouse button can be released. Once released, GeoTraceMan calculates and updates the appropriate map statistics and draws the new zoom.

6.5.1 Bounding Box

A bounding box is the most natural way for a user to specify a region of the map that they wish to inspect. The box indicates to the user, before the zoom is actually performed, the exact area that will be displayed.

As previously stated, the bounding box is maintained as a square rather than allowing the user to specify any rectangular shape. This stops problems of aspect where for example, the user attempts to zoom into the United States of America with a rectangle of length to width ratio of twelve to one. It was simpler to enforce the square bounding box than attempt to calculate an aspect zoom from a non-aspect bounding box.

The square aspect of the bounding box was produced using a simple test. The initial point from which the user started the zoom is stored. The box is drawn on the canvas as the user moves the mouse. The square is drawn using the larger of the x and y values as the side size of the square. The bounding box also has the ability to mirror itself in the x and y directions. The square aspect is maintained, even when the box mirrors across an axis.

The three dimensional map has the same bounding box functionality as the flat map model. The data received from the bounding box is still based in two dimensions. The zoom then drops back to the two dimensional map. To be able to perform this correctly, the bounding box dimensions have to be converted back to two dimensions.

6.5.2 Unlimited Undo

An *unlimited undo* feature was added into GeoTraceMan. This feature allows a user to *undo* zooms and other map operations. This is achieved through storage of appropriate map details on a stack. This data is pushed onto the stack when a zoom is performed, and popped off when an *undo* is required. The data contained on the stack includes all of the necessary information to recreate any combination of zooms, map inclusions and map rotations. When the *undo* option is required, the data at the top of the stack is popped off and stored as the current map data. When the map is updated, the newly popped data is used, "*undo*ing" the previous zoom operation.

The *undo* is intelligent, in the sense that it will clear the stack as soon as the map is reset. This allows the user to select the "Reset Map" option rather than having to traverse the entire stack, popping off the last map state. This can be seen as no longer *unlimited*, but it was felt there would be no need to store data once the user had reset all of the changes made. This decision was based on three discoveries made during the testing phase:

1. The *undo* feature is rarely used for more than approximately two *undo*'s per trace
2. Separate traces usually result in different areas of interest, hence *undo* information is rarely used to examine two geographically distant areas
3. The *Reset Map* option is used more frequently than the *Undo* option as after zooming more than one level, it is easy to become disoriented and a *Reset Map* was found to be the quickest method of "reorientation"

As can be seen from the above, the *Undo* option is mainly designed to be used in situations where a zoom was accidentally performed and the user does not wish to recreate their current map from the original state. The probability of this occurring has increased since zooming can always be performed by simply clicking on the map canvas. Originally, this was not available. This provides a fast and efficient method of achieving extra functionality which is "naturally" obvious to the user.

6.5.3 Zooming in Three Dimensions

Three dimensional zooming was to be a simple extension from flat map zooming. This proved to be an unfounded assumption. The theory of the three dimensional zoom was to drop back to the two dimensional map once the bounding box was defined. The first noticeable problem was a lack of reverse conversion for the bounding box coordinates. This produced a two dimensional zoom that was grossly out of aspect. A solution that was not implemented as of the date of this report was to convert the coordinates of the bounding box back into two dimensions then determine the zoom, while maintaining aspect.

Even though three dimensional zooming was not considered vital to the successful completion of this project, some consideration was given to it. The basic process of achieving this was to calculate the starting point of the zoom and produce the other coordinates of the bounding box from this in two dimensional coordinates.

6.6 City Inclusion

In order to obtain a point of reference, an option to include city names was offered. When selected, this option displayed a series of city names on top of the currently displayed map. As with the maps, the level of detail (number of cities) shown is dependent upon the zoom. At higher levels of zoom, more cities are displayed. The exact number of cities displayed at each level of zoom is dependent upon the number of cities found in the cities files and the location of these cities.

Displaying the contents of the cities files is performed in a similar manner to the coastline maps themselves. At a certain zoom level, the second cities file is read in and at a greater zoom, the third. These zoom levels are hard coded. The optimal solution would be to provide these zoom levels as user setable resources with fallback default values.

As multiple files are used to hold city data, the possibility of the same city existing in multiple files arises. If the location of any two cities is identical, or relatively close, the second will be clipped by the algorithm outlined below.

```
if (zoomlevel == 1) (step 1)
        draw all cities in cities file 1
if (zoomlevel == 2) (step 2)
        perform step 1
        for all cities already displayed (citycoord) do
                if (abs(xcoord-LVL2XBUFFER) ≤ citycoord.x)
                        if (abs(ycoord-LVL2YBUFFER) ≤ citycoord.y)
                                drawcity
if (zoomlevel == 3)
        perform step 2
        for all cities already displayed (citycoord) do
                if (abs(xcoord-LVL3XBUFFER) ≤ citycoord.x)
                        if (abs(ycoord-LVL3YBUFFER) ≤ citycoord.y)
                                drawcity
```

If the distance between the two cities is too great, they are not clipped by this algorithm and both will be drawn. It is not possible to perform clipping on any field but location. Many city names are duplicated across countries, states and even within a state.

A linked list of city locations is maintained to provide a method to check the location of any city to be added to the map. This linked list is traversed whenever a city name is to be displayed. If it passes the test (signified in the above algorithm), it is appended to the list.

6.7 Map Inclusion

Another means of maintaining a point of reference is the overlaying of different thematic maps. Options included in GeoTraceMan are rivers and political boundaries overlays. These maps are in identical format to the coastlines map, hence are displayed via the same mechanism as the coastlines. Due to this, the three dimensional displaying of this data is also performed in the same manner as for the coastline map.

The rivers overlay depicts the major rivers of the world. This, generally, does not help the user unless they are zoomed into an area with a major river as a feature. The political boundaries provide a better solution to this problem. These boundaries include the boundaries of the world's major countries and the state boundaries within the United States of America.

The original version of GeoTraceMan offered map inclusion.

6.8 Atlas Map Widget

The Atlas Map Widget is a commercial widget and toolkit that displays the world in three main projections, orthogonal (three dimensional), cylindrical and Mercator. This widget provides zooming, rotating, map inclusion and city inclusion. The main advantage this widget has over GeoTraceMan is its speed. Its disadvantage is its cost:- $US25,000 to be able to re-sell any tool using it.

The Atlas Map widget provided a target. The ultimate goal of the three dimensional additions to GeoTraceMan was to produce a finished product with the flair of the orthogonal projection in the Atlas Map widget. As can be seen from the figure below, the Atlas Map widget is a full featured production level widget. If purchased, it comes with the widget itself and the toolkit which offers extra functionality such as the inclusion of totally different sets of data at different zoom levels, the inclusion of pixmap images on the map surface and the drawing of lines over the surface of the map (regardless of projection).

The demonstration program shown above utilises some of the features of this widget. This program is available from `ftp.uu.net:/vendor/3rdPlanet`. SunOS 4.1.x, IRIX 4.0.x, HP/UX 9.0.x and IBM AIX versions are available.

Throughout this project, the Atlas Map widget was referred to as the "correct" answer. Anything less than comparable was considered an unsuccessful attempt.

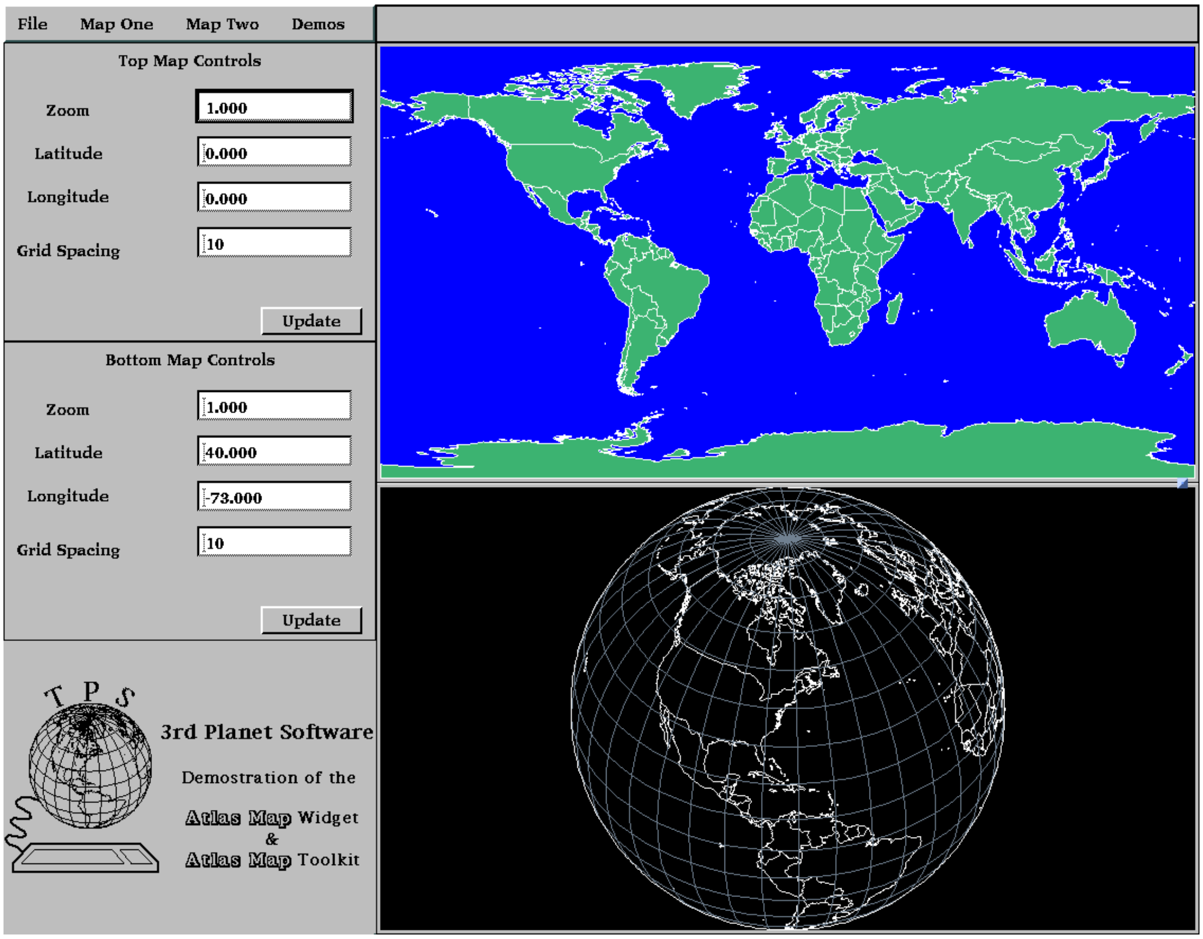


Figure 16: Atlas Map Widget

7.0 Domain Name System Resolution

"The Domain Name System is a distributed database. This allows local control of the segments of the overall database, yet data in each segment are available across the entire network through a client-server scheme. Robustness and adequate performance are achieved through replication and caching." [Albitz & Liu, 1992].

The two arms of DNS are the *name servers* and the *resolvers*. The name servers store the information about segments of the network. They answer requests from the resolvers.

Each domain name specifies its position in the distributed database. This position is determined by the recursive decomposition of the subnets that the host in question resides on. For example, `beavis.cs.curtin.edu.au` can be decomposed into belonging to the `.au` (Australian) subnet, the `.edu` (education facility) subnet, the `.curtin` (Curtin University of Technology) subnet and finally the `.cs` (computing science) subnet.

For further information on DNS, the book *DNS and BIND* [Albitz & Liu, 1992] is recommended reading.

7.1 Requests For Comments

As with all Internet standards, DNS was accepted through the series of reports known as *Requests for Comments*. The main RFC's dealing with DNS are:

- RFC 1032 - Domain administrators guide
- RFC 1033 - Domain administrators operations guide
- RFC 1034 - Domain names - concepts and facilities
- RFC 1035 - Domain names - implementation and specification
- RFC 1101 - DNS encoding of network names and other types

These RFC's discuss the concepts behind DNS and it's implementation. They provide all of the necessary information to install and maintain a DNS based site.

7.2 DNS Basics

The information contained in the DNS is stored in Resource Records (RR). The data held in these RR's is entered by the local systems administrator. The data is of an exact format as defined by the RFC's. An example entry for the DNS could be:

```
beavis  IN      A       134.7.1.21
        IN      MX      0       marsh
        IN      MX      10      cupid.curtin.edu.au.
        IN      HINFO   SUN-SLC SunOS-4.1.3
```

This shows the address (the "A" field) of the machine `beavis` to be 134.7.1.21. The mail exchange point for `beavis` is `marsh`, then `cupid` if `marsh` is unavailable. The last entry is the HINFO field. It contains two strings indicating the machine and the operating system.

7.3 DNS Work at Curtin University

DNS lookup was always to be implemented into GeoTraceMan. As this project was proceeding fairly smoothly, it was felt that this would be an ideal time to incorporate this feature.

A major part of DNS lookups is the host being looked up. For this purpose Craig Farrell and Mike Schulze modified the 4.3 BSD version of *named*, the name server daemon, and ran it on the Department of Computing Science network name server. After a few small problems, a version of *named* was produced which was stable and responded as expected to all requests, including those requiring geographical positioning (GPOS) information. As a further test, the same code was supplied to a neighbouring university, the University of Western Australia, and run on their library network. This also passed all tests by responding as expected.

To test the name servers, versions of the most popular DNS lookup software, such as *nslookup*, *dig* and *dnsquery*, were produced. It was from this software that the final GPOS query was constructed. This query was to be incorporated into GeoTraceMan.

As a result of the successful modification and implementation of the DNS, the code was submitted to Paul A. Vixie, the source code controller for BIND (Berkeley Internet Name Domain). Paul Vixie was working on a similar concept, the location (LOC) resource record (RR).

The basic functionality of GPOS and LOC is the same. They both can be used to provide geographical location by means of a DNS query.

7.3.1 GPOS Basics

GPOS was designed to suit an application, namely GeoTraceMan. This affected the implementation of this Resource Record. GPOS was kept as simple by the use of strings rather than a bit encoded record. The use of strings allowed, in essence, infinite precision (up to the length of the string).

The data contained in GPOS was:

- Latitude - the latitudinal position of the host
- Longitude - the longitudinal position of the host
- Altitude - the altitude of the host above or below sea level

The ease of implementation was also considered a strong point of GPOS. The systems administrator was simply required to enter the longitude, latitude and altitude of their site as accurately as they desired.

The draft RFC for the experimental GPOS RR can be found in Appendix A.

7.3.2 LOC Basics

LOC was created by Paul Vixie and Christopher Davis as a possible method of resolving host data for Brian Reid's UUCP Mapping Project (software available from `ftp.uu.net`). As such, LOC was designed with this project in mind.

The LOC Resource Record encodes all of its data in a specific record format. This format includes such fields as:

- Precision - how accurate the supplied data is
- Size - the physical size of the host
- Latitude - the latitudinal position of the host
- Longitude - the longitudinal position of the host
- Altitude - the altitude of the host

As this data is to be stored in a record, there has to be a conversion between the method of input (either decimal degrees or degrees/minutes/seconds) and the transferable (integer) format.

7.3.3 GPOS versus LOC

All Internet standards should be discussed with the Internet community at large. This ensures they are the best possible standards for the job. This is one reason the Internet has become the success it is. Being informed by Paul Vixie he was working on a similar experimental RR created such a discussion. Over a period of two weeks, both the GPOS and the LOC experimental standards were modified enormously.

There were three main arguments the GPOS team had with LOC:

1. The redundancy of the precision and size fields
2. The difficulty of entering data
3. The lack of "infinite" precision

If numbers entered were not very precise, the application querying the host would have no choice but to use the data anyway, making the field redundant. The size field was introduced to allow network size to be recorded. This field also allows the entering of microscopic hosts, which is nonsensical. As the military does not allow access to data with accuracy greater than plus or minus ten metres, this field loses meaning.

Data would have to be entered in a format that could be understood by the average systems administrator. The format LOC was suggesting was non-trivial to encode. It required the location to be stored as fractions of seconds in order to remove the limitation DNS lacking a floating point type.

The lack of "infinite" precision was created by the need to fit the data into a predefined type, the 32 bit integer. This created a major deficiency in LOC, a trade-off between accuracy and coverage. If the data was to be highly accurate, the area of coverage would be decreased, yet if the area of coverage was to be expanded, the accuracy would be decreased. To allow for "nanocomputers" (many of which can fit into a thousandth of a degree second), the LOC RR had to disallow satellites in geostationary orbit.

The main arguments the LOC team had with the GPOS implementation were:

1. The size required to store the data
2. The need for parsing of the data
3. Is there a need for "infinite" precision?

The maximum possible size that a GPOS entry could be is 256 bytes. This compares with the 18 bytes for LOC. On an average case, the space used by GPOS is approximately (depended upon precision required) 21 bytes. As can be seen, LOC requires less space than even the average GPOS entry.

With floating point data stored in strings, the need for parsing appears. One goal of DNS was to limit parsing. The GPOS string was required to be parsed as three floating point numbers.

The third argument the LOC team had with the GPOS implementation was based on a need for "infinite" precision. They saw no real need for this and pointed out that it increases complexity. To take advantage of a latitude with, for example, 120 decimal places, a new parser must be written (the *scanf* family can not cope with this precision). The application that parses such a number must also perform any mathematics using specially designed formulae as there is no standard type for a 120 decimal place real number. The LOC team also raised the point of obtaining data to this degree of accuracy, not to mention entering it into the DNS.

7.3.4 The Solution

The solution to the LOC versus GPOS debate was decided with the withdrawal of the GPOS experimental standard. GPOS was withdrawn due to pressures from Paul Vixie and the LOC team. The inability of any members of the GPOS team to travel to the IETF (Internet Engineering Task Force) meeting decreased the chance of GPOS begin accepted as a standard. The IETF perform the task of determining which draft RFC's become accepted standards.

7.4 Cache File Resolution

The current method of resolving the location of a site is through the use of external cache files. Three levels of caching exists:

1. Known backbones
2. Local data
3. Known networks

The backbones file is the first consulted. This contains the geographical location and IP address of all known backbones. A backbone is considered any major site in the major networks (such as NFSNET, JANET and AARNET). If the site is not found in this file, the local data is searched. This file is used to contain hosts that are not in the same geographical location as the rest of the network. This can be used for hosts which are, for example, connected via a dial-up line. The final search is in the networks file. This contains the network address and locations of known networks. After these three levels of caching if the location of a site is still unknown, it is displayed at the same location as the last site. If this happens to be the first site in the trace, it is displayed at 0,0 (near Africa).

7.5 Implementation in GeoTraceMan

One question arose before the attempt to incorporate DNS lookup into GeoTraceMan: what to do if we don't resolve a name? The previous method of maintaining cache files solved this.

The DNS lookup was always an *option* in GeoTraceMan, hence, cache files would always be kept. If the DNS lookup was not able to resolve the GPOS entry for a particular site, GeoTraceMan would fall back to the cache file method of lookup already implemented.

The actual code for the DNS lookup that was added to GeoTraceMan was mostly written by my project supervisor, Craig Farrell. Only small modifications were required to produce the needed result. This programming feat was performed by utilising the source code to the 4.3 BSD version of BIND.

As the GPOS entry was withdrawn, there is limited chance DNS lookup will be included in the final version of this project. When the LOC RR becomes an accepted Internet standard, it will be implemented into GeoTraceMan.

7.6 Update on GPOS

The GPOS experimental standard was officially accepted as an RFC on November 3, 1994. As was previously stated, the draft RFC was withdrawn by the GPOS team. It is unknown how this was overlooked. The future of LOC is unclear now that GPOS is an official standard. If an implementation of GPOS is submitted, it *has* to be included in BIND, partially ensuring the success of DNS lookups in GeoTraceMan.

If LOC becomes an accepted standard, it too will be implemented in GeoTraceMan. The likelyhood of this occurring is unknown.

The next release for GeoTraceMan will provide DNS query facilities. This may include solely the GPOS query or both GPOS and LOC. It is undecided if this situation should be exploited.

8.0 Other Modifications

The development of GeoTraceMan was not a single task project. Many small tasks had to be completed to make GeoTraceMan a reliable and usable tool. The majority of these modifications seem trivial when studied individually, but when combined together, they create a substantial part of the project.

Most of the modifications discussed in this section are directly related to the user interface and associated functionality.

8.1 Kill Trace

An important feature of any utility that spawns off sub-processes is the ability to kill those sub-processes. GeoTraceMan now provides this facility. The process that GeoTraceMan spawns is a modified version of Van Jacobson's *traceroute* [Jacobson, 1989]. The data obtained by *traceroute* is passed back to GeoTraceMan via an unnamed pipe.

When a trace is started, the process identifier (PID) of the spawned trace process is stored. When the *Kill* button is selected, this process is killed with the SIGKILL signal. The value of the process identifier is then set to negative one (-1). This is an invalid process identifier, so any attempts made to kill a process with this identifier will fail. PID 0 is the swapper daemon on most systems, hence was not chosen. As GeoTraceMan has to be run as root, this could have created a security hole. SIGKILL is sent to the appropriate process it can not be caught, blocked or ignored.

Other events that occur when a the trace process is killed include the closing of the socket and pipe descriptors for the trace and setting the trace status widget to its *waiting* state.

A "feature" of the *Kill* option is the ability to display multiple traces. If a trace is killed before it completes (and not removed), another trace can be started. This second trace will display over the previous trace as if it did not exist. For all practical uses, this is correct; the first trace no longer exists. If any map functions are performed, the original trace is not redisplayed, limiting the usefulness of this "feature".

As multiple traces were to be added, functionality of the *Kill* option was not improved. In order to correctly cater for multiple traces, potentially simultaneously, this would have to be redesigned.

8.2 Panner and Scroll Bars

Once again, the panner and scroll bars were not implemented. Some consideration was given to the development of the panner, but it was not implemented due to a lack of time. The scroll bars that were previously visible, but not used, were removed in an attempt to implement the rotator buttons described below.

The concept behind the implementation of the panner widget was to provide a method of constant reference. The user would always know where they were in the world via a quick glance at the panner. When the user is zoomed in, a box would be drawn on the panner widget indicating the current extents of the zoom. Functionality of the panner should include the ability to zoom by drawing a bounding box on the map displayed in the panner widget and the ability to move the current zoom box on the panner widget by dragging it with the mouse.

The panner widget could be implemented using the GeoTraceMan icon as the map. The size of the panner was initially designed for the incorporation of this icon.

8.3 Rotators

The possibility of using rotator buttons instead of scroll bars arose early in this project. It was felt to be more natural when used in combination with the aforementioned panner widget. The rotator buttons were to be placed directly below the panner widget to allow easy and logical access. These buttons would consist of a set of four, allowing rotation in the north, south, west and east directions with a simple click of the mouse.

The buttons were to be attached to the standard rotation functions. These would require modification to allow rotation in a north and south direction, as well as incremental rotation when zoomed in.

The rotator buttons were not implemented due to a lack of time. The implementation should be a simple process requiring the creation of the four command widgets (using bitmaps to display the arrows indicating the direction of rotation) and attaching callback procedures to these buttons to perform the required rotation.

8.4 Trace Loading and Saving

The loading and saving of traces allows the use of GeoTraceMan without having to provide the root access required to start a trace. A trace could be saved and later loaded to show the path that packets were taking at the time of the original trace. This can be used for comparison purposes, or even as a teaching aid.

A potential security hole arises with the ability to save traces. As a trace must be executed as root, the saving of a trace provides root access to the file system. There is no method to overcome this without limiting either the ability a trace or save a trace. A trace can be saved over any file, destroying the previous contents. Loading a trace does not encounter this problem as the file is only read. If does not contain appropriate data, nothing is displayed, hence, security is maintained.

Loading and saving of traces was not fully implemented. It was felt that the incorporation of multiple traces should be added before this was studied. To allow for multiple traces, the basic trace data structures of GeoTraceMan would require total redevelopment. This was considered to be beyond the bounds of this project, but will be attempted in the immediate future.

Trace saving and loading was developed to the extent of creating the file selection box and processing the request therein. The actual input to and output from a file is not performed.

It was decided of ASCII text file rather than binary files would be used when eventually implemented. As can be seen from the difficulties that arose when porting to a platform with a different *endian*, the use of binary files can be troublesome. Both the loading and saving of traces pops up the same file selection box, mostly stolen from the Free Widget Foundation [Totty, 1994].

8.4.1 Free Widget Foundation

"The Free Widget Foundation (FWF) is a collection of no-cost, freely redistributable X graphical user interface widgets" [Totty, 1994]. The collection currently numbers 58 widgets of varying descriptions and functions. The widget that was utilised for in GeoTraceMan was the *FileSelector* widget. This widget can be seen in Figure 17: The FWF File Selector Widget below.

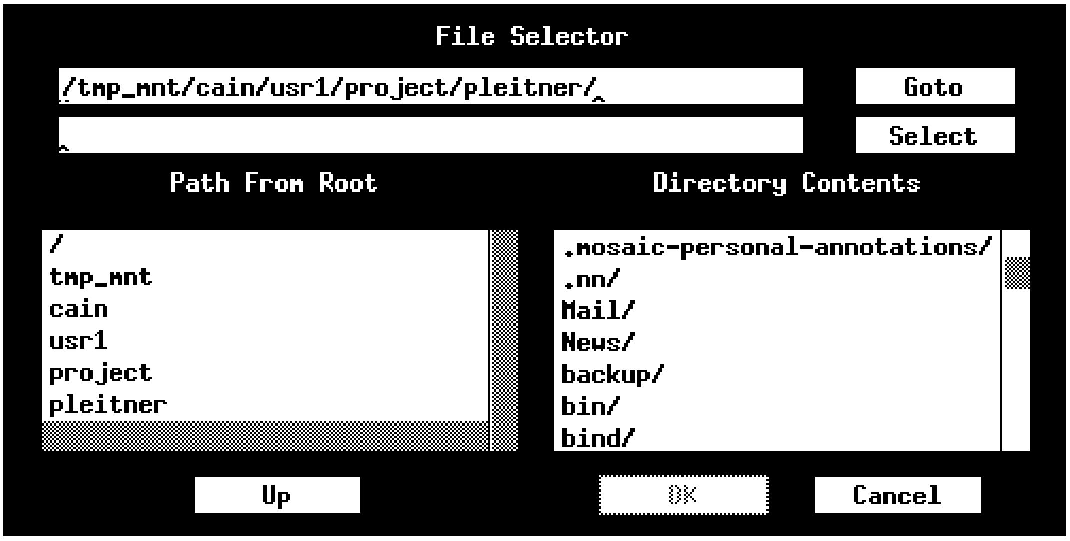


Figure 17: The FWF File Selector Widget

The functionality of the widget, shown in the above figure, is similar to most file selection widgets. It was chosen above all others for its simplicity of implementation. Very little work was required to produce a library from this widget which could be linked into GeoTraceMan. Only a handful of subroutine calls were then required to set up the entire widget. The data returned from the widget was the path of the file and the file name itself. Through a simple string concatenation, the complete path to the file could be obtained and the file opened for reading or writing as required.

The widget was eventually removed from GeoTraceMan for the reasons outlined in section 8.4 Trace Loading and Saving.

8.5 Postscript Dumps

The postscript dump option was not included in this version of GeoTraceMan. The possibility of this being a project for another undergraduate student prevented it from being attempted here. The basics for the dump could be extracted from other Netman tools such as *etherman* and *interman*. The size of the postscript produced from these tools is approximately fourteen kilobytes for a screenful of data. As can be seen from the size, this is fairly efficient (compressed) postscript. It would be expected that the postscript produced by GeoTraceMan would be larger than this due to the extra complexity displayed.

8.6 Continuous Rotation

After the rotation of the three dimensional globe was successfully implemented, the ability to continually rotate the globe was desired. This was achieved through the use of an Xt call designed to execute a procedure whenever the event loop is idle. The rotation is extremely slow due to the amount of data passed through the two to three dimension conversion routine.

The continuous rotation is implemented through a command button and a popup dialog. The dialog is used to receive input indicating the direction to rotate.

9.0 Release of GeoTraceMan

The release of GeoTraceMan was always a major aim of this project. This proved to be the major learning point of the entire project. Work which was *almost* correct was no longer acceptable (as it usually was for assignments). The code produced was required to work all of the time for all possible inputs. Testing was not option, but a requirement.

9.1 Testing

Testing is a vital part of development of all software, particularly software that is destined for release. The testing of GeoTraceMan was not a single phase of development, it was performed as each modification and addition were made. The main aim of testing was for validification of the code, ie. the code does what it is supposed to do. Efficiency was tested via other means.

True testing and validification of software is a concept introduced in Software Engineering. It provides us with two objectives: [Sommerville, 1992]

1. The discovery of defects in the system.
2. The assessment of whether or not the system is usable in an operational situation.

These strict guidelines were not used to test GeoTraceMan. A simplified version of this testing was used. The logic of modifications made were tested using simple human understanding (how logical are humans anyway?) and in a running state with both real world data and data that would be likely to introduce unforeseen errors.

9.1.1 Errors Encountered

The testing of GeoTraceMan provided an almost endless supply of errors. Some of the more *interesting* and important errors will be discussed. All of these errors were fixed, with the exception of the memory fragmentation problem. This turned out to be unavoidable.

9.1.1.1 Start Trace Address Entering

To start a trace, the user must click on the *Start* button. This pops up a dialog box requesting the location of the host to trace to. Initially, the user had to enter the Internet name of the host they wished to trace to. If the Internet number was entered, the program would block waiting for *traceroute* to reply, even though it had yet to be spawned. This was corrected by allowing the user to enter either Internet names or numbers. If required the conversion was performed.

9.1.1.2 No Map Display

When compiled with the *-g* option (to include debug information), GeoTraceMan worked as expected (although slightly slow due to inclusion of the symbol table). When a preliminary release was being prepared, the debug option was converted into an optimise option, *-O2*. The code compiled successfully, yet when run no map data was displayed. The optimiser option was then changed to *-O* and GeoTraceMan was recompiled. Once again, no map data was displayed. As a last resort, optimisation was turned off completely. The map data now displayed. It is assumed incompatibilities exist between the SUN optimising compiler and the VOGL graphics library.

This problem has no solution, other than leaving off all optimisations. Another potential incompatibility tested was dynamic versus static linking. No problems were encountered with this.

9.1.1.3 Illegal Instruction Generated

At one stage during the development, an "Illegal Instruction" was being generated. This appeared for no known reason. The code in which it appeared had not been modified for approximately two weeks. It was assumed to be a compiler problem as when GeoTraceMan was compiled with the *-O2* switch instead of the *-g* switch, the problem disappeared. This, however, recreated the problem outlined in section 9.1.1.2 No Map Display.

The solution to this problem was a restoration of code from two days previous (when the "Illegal Instruction" was not plaguing GeoTraceMan). This alleviated the problem and little work and even less time was lost due to the use of RCS (see section 3.3 Source Code Control).

9.1.1.4 No Hops

When no hops were required to complete the trace, a *core* dump occured. This was due to the nature of the trace structure used. As no hops were required, no trace data is saved. GeoTraceMan still attempts to calculate statistics. This results in an attempt to access memory not under the control of this process, generating a sementation violation. This was corrected by inserting a check for the number of hops. If this count was zero, a popup message was displayed indicating no sites were encountered. The trace is not performed in these circumstances.

9.1.2 Testers

The developers of all software understand the true capabilities of their product, hence, tend to (subconsciously of course) test around potential bugs. External testers are required to expose these potential bugs. Numerous people were involved in the testing of GeoTraceMan, most of whom were *conned* into it. Many of these were friends and members of staff at Curtin University. Testers from Curtin that deserve a particularly mention include:

- Mazahir Songerwala
- Geoff West
- Jarrod Douglas

These people helped in numerous other areas of this project, hence are rewarded with a mention here.

As this tool was to be used within the Internet, testing would have to be performed outside Curtin University to ensure that GeoTraceMan would function correctly at sites with different setups to Curtin. Remote testers provided feedback that would have otherwise been unattainable. This also allowed the cache files to be updated before the final release as the section of the Internet seen from Curtin is very different to the section of the Internet seen from a host at, for example, Chicago.

External testers provided feedback indicating the usefulness of GeoTraceMan outside Curtin University. An external tester that deserves particular thanks is Carl Brewer from the University of Western Australia.

9.1.3 The Ultimate Test

The ultimate test of any piece of software is the mass market. A preliminary version of GeoTraceMan was released on 23rd of September, 1994. This release contained the functionality of the final release minus the three dimensional map option. The aim of this preliminary release was to receive some feedback on the usefulness of the tool, bug reports and updated cache files.

The first two weeks of the preliminary release proved the most useful with over 300 people *ftp*ing the binaries and approximately 150 *ftp*ing the source. The response to the release was confusing at first. Very little email was received pertaining to GeoTraceMan. Eventually feedback was received, mostly relating to difficulties in running the tool. These problems were mainly due availability of the hershey fonts used and differing versions of the resolver library (libresolv.a).

9.2 Memory Leaks

Most large programs suffer from some memory leakage. GeoTraceMan was no different. Memory leakage can be considered as any memory that is allocated and never used or deallocated. This can be compounded if the memory is unreachable, ie. no pointer points to the block of memory. This does not allow the memory to be deallocated, even if this is desired. This usually occurs through the mismanagement of pointers.

Purify [Pure Software Inc., 1993] was used at the suggestion of John Krivitsky. This proved to be enormously helpful. Purify clearly indicated all memory accesses and highlighted the illegal and possibly incorrect ones. When GeoTraceMan was first run through Purify, it produced over 2 megabytes of text output regarding memory accesses. Most of this data was repeated accesses that were legal, hence the introduction of the *-firstonly* switch, which reports an access the first time it occurs only. This reduced purifies report to a more manageable size. The majority of this data was still legal accesses. There was, however, some memory leaks present. The total memory leaked in GeoTraceMan is just over six kilobytes. This was not corrected as a rewrite of the map handling functions of GeoTraceMan was planned and this would simply duplicate effort.

A large number of the illegal accesses (not leaks) recorded by purify were "uninitialized memory read" errors. An alarmingly large number of these errors were found to be in the system and X libraries, hence were not repairable. The illegal accesses that were not produced by external libraries were not corrected due to the imminent rewrite.

9.3 Memory Fragmentation

GeoTraceMan is memory hungry. The map data requires approximately 7 megabytes and a large portion of this is required to be in memory. When the zoom level changes (see section 6.5 Zooming), the map of the required zoom level is read from disk into memory. Before this can happen, the memory that the previous map was using is freed. Due to the nature of the data structure that contains the map data, many small allocations and deallocation occur. This causes memory fragmentation.

9.3.1 UNIX Memory Management

Most UNIX operating systems work in a very similar manner for allocation and deallocation of memory. A process has its own memory allocation pool. When memory is required by this process (such as a call to *malloc*) this memory pool is first checked. If it contains enough memory to satisfy the request, the memory is allocated from the pool. If the pool does not contain enough memory to satisfy the request, the operating system steps in and increases the size of the memory pool.

Memory deallocation is performed in the reverse order. The memory is made available to the processes memory pool on calls to functions such as *free*.

Subsequent memory allocations should take place from the process's own memory pool. If the memory pool is fragmented (no single contiguous block is large enough to satisfy the allocation request) the operating system must step in and increase the size of the pool for this request to be fulfilled. Eventually memory will be exhausted and allocations will fail.

Most of the information for this section was obtained from the UseNet newsgroup `comp.lang.c`.

9.3.2 Fixing the Fragmentation

The only plausible method available to fixing the fragmentation occurring in GeoTraceMan is to perform memory management for the operating system. As the specific needs of GeoTraceMan are understood, we can provide a specific solution that could decrease, if not stop, the fragmentation. This solution would involve the use of *malloc* for the small, non map data related allocations, including the use of *malloc* within the libraries, and the use of a special form of management for the map data. This memory management would involve the use of a stack with a guard page. Memory is allocated from the stack and when the guard page is reached, another page is requested from the operating system. There is still no guarantee that this method would produce anything other than extra overhead. Contiguous memory is not guaranteed.

Another possible, but not plausible, method to perform memory management was to allocate memory through the use of the *brk* and *sbrk*. These two functions can create a large problem when a library function wishes to allocate memory. As the above functions directly modify the processes allocation pool, library functions have an inconsistent pool to allocate from. This can cause errors ranging from none (everything works fine) to annihilation of other data in memory. This is not considered an acceptable risk.

A third possibility was considered. A large piece of memory could be initially allocated and all of the map data read into this. As map operations are performed, they duplicate the required sections of data and operate on this. When the processing is complete, the newly copied block of memory can be deallocated. The next map operation reads from the original piece of memory again. This speeds up the application by decreasing the amount of disk access. The fragmentation is avoided by allocating large contiguous blocks of memory.

9.4 Final Release

The final release of GeoTraceMan was not accomplished in this project. The conclusion outlines some sections that need reviewing and possibly creating to prepare GeoTraceMan for final release. The final release should be a stable, reliable and useful program. GeoTraceMan is only part way to achieving this goal.

Before the final release, a major rewrite of the map reading and displaying functions will be needed. This is the slowest section of the entire program. As this is also one of the most frequently executed code segments, it should be made one of the most efficient, both in terms of speed and space. A rewrite could also remove the memory fragmentation problem outlined in section 9.3 Memory Fragmentation.

10.0 Conclusions

All project should provide some answers at their completion. This project, even though it was only an undergraduate *educational* project, is no different. The release of GeoTraceMan was an important aim in this project, and it is from this aim the most was learnt.

10.1 The Future of GeoTraceMan

GeoTraceMan has great potential. The natural future of any tool developed in limited time is extended development. Some areas that should be developed further are briefly discussed below. This is a limited selection of possible extensions that could be added. As was stated in the original GeoTraceMan report, "With any program, it is always difficult to know when to stop coding" [Brown, 1993].

10.1.1 Postscript Dumps

As was previously stated, postscript dumps were not incorporated into this project. The possibility that another undergraduate project could be made from this stopped its development in this project. Producing postscript would require the map data to, effectively, be drawn again on a postscript drawable rather than an X drawable. The process to achieve this has not been studied.

Other Netman tools include postscript dumps as a part of their functionality. Initially, it appears as though it would be relatively simple to reuse this code in GeoTraceMan. This is unlikely due to the map data displayed on the GeoTraceMan drawable. To be efficient, the postscript should be produced in a single pass rather than a pass for the coastline data, a pass for the political boundaries, rivers and cities, and yet another pass for the trace itself. The size and time required to process such postscript would be unacceptable.

10.1.2 Multiple Traces

The ability to display multiple traces on the same map would be useful. This could be used to display the deviations between two traces. It could also be used to indicate any deviations between the same trace performed at different times to show routing decisions as they occur.

The possibility of multiple traces was first mentioned in [Brown, 1993]. Multiple traces have been studied since this report and it was felt the best method of achieving this was to recreate the data structures used to hold trace information. Since the map data structures were to be rewritten for efficiency and functionality, it was felt that the trace structures should also be rewritten with the same aim.

The current method of storage of trace information is too static to be useful for multiple traces. The structure needs to be dynamic, both in terms of the allocation of multiple traces and the number of hosts in each trace.

10.1.3 Trace Saving and Loading

As was mentioned earlier, the ability to load and save traces was not fully implemented in this version of GeoTraceMan. The possibility of including multiple traces halted the development of file functions.

If multiple traces were included in GeoTraceMan, the ability to store trace data when multiple traces were displayed would have to be studied. Possible solutions to this would be to save all of the traces, or just the last trace to a trace file. The best solution would be a query system which asks the user which trace(s) they wish to save. This would be fairly simple to implement once the rest of the file handling system was in place.

Another addition in the field of file input and output would be to save and load the map status. A zoom that the user finds particularly useful could be saved and restore this any time of their choosing. An extension to this could be macros for preset map states.

10.1.4 Panner Widget

The panner widget is still not implemented. It seems this is always avoided. As stated in section 8.2 Panner and Scroll Bars, the panner was aimed at providing the user with constant reference as to where they are in the world. This is especially useful when zoomed in. The implementation of the panner is not merely a coding exercise. Some method of maintaining synchronization between the map canvas and the panner widget would have to be created. Without this the panner widget would be mostly useless. It would simply be another method of specifying a zoom.

The interaction required between the two would have to be at the highest possible level. When the zoom was updated via the panner, the zoom would have to be updated on the map canvas and vice versa. If the user were dragging the bounding box drawn on the panner, GeoTraceMan would have to wait until the user has selected the new zoom (by dropping the drag) before updating. The map redrawing rate is simply not fast enough to try and keep up with a user dragging a bounding box.

10.1.5 List of Sites Touched

A popup list of sites touched by a trace could be a useful extension. This popup could contain information received by the trace, such as the delay to each site. At present, GeoTraceMan only displays the delay information from the source to the destination site. This popup could also be used to contain other information. Such information could include the IP name and number (for those sites that the background *traceroute* did not resolve) and the information returned by the *whois* command.

The data for this popup could be obtained in the background. Spawning extra processes to gather this information would not noticeably increase the load GeoTraceMan places on the source machine.

A further extension to this could be the inclusion of SNMP statistics. This is discussed in the next section, 10.1.6 SNMP Statistics. The most relevant SNMP provided data for this popup would be the system contact.

10.1.6 SNMP Statistics

A further extension to the above popup list could be the addition of SNMP (simple network management protocol) queries. SNMP is becoming a widely used protocol within the Internet. It is beyond the scope of this report to fully explain SNMP. Suggested reading is [Comer, 1991]. Chapter 25 of this book provides full coverage on this subject.

The ability to query a site via SNMP could provide excellent insight into the operation of a particular gateway or host. This could extend GeoTraceMan from simply a tool that monitors a wide area network to a tool that can be used to partially control such a network.

The data offered should be selectable by the user. The selection process should be presented as a tree structure indicating the different groups in the MIB. The breakdown of each group should be offered as subgroups, finally down to queriable variables. It is debatable if setting SNMP variables should be included in a tool such as this. If sites have not secured their SNMP agents against such, a large amount of damage can be caused. If it were included, it would be added to the list of functions that only the superuser should have access to.

10.1.7 Converting the Map to a Widget

The acquisition of the Atlas Map Widget produced a desire to achieve similar performance and functionality with GeoTraceMan. If the GeoTraceMan map canvas was converted to a separate widget, the possibility of more efficient mapping could be realised.

If this were converted into a widget, it would most likely be released to the public domain. The possibility of others in the Internet working on this widget is high. Any modifications and improvements made would find their way back into GeoTraceMan. Obviously, this improves GeoTraceMan.

Converting the mapping functions of GeoTraceMan into a widget would provide a method of including the basic maps into other Netman tools. The charging tool currently under production, *chargeman*, could use this widget. It would only require the basic functionality of displaying and drawing lines on the map (the tool is yet to be released so it will not be discussed in this report). There are also plans to add an optional interface to *interman* that includes this widget.

This widget could also control a child widget:- the panner. This would provide a solution to the synchronisation problems outlined above. The control of the panner widget would be performed internally, hence the implementation of both the map and panner widgets would be made trivial.

10.1.8 Cyclic Route Detection

The inclusion of a cyclic route detector would enable simpler spotting of possible routing problems. By keeping a better record of the sites encountered during a trace, the detection of cyclic routes is possible. If a site is encountered more than once, a cyclic route (or at least a routing problem) has occurred. The second time the site is touched, the displaying of the name could be converted to a different colour to indicate a problem has occurred. Another possible action is a popup box indicating the routing problem. Other routing problems could be discovered via the SNMP queries.

If the SNMP set operation is allowed, it would be possible for GeoTraceMan to correct routing errors it finds.

10.1.9 Multiple Displays

The ability to display multiple zooms in multiple windows was suggested by an undergraduate student at Curtin University of Technology who may be extending GeoTraceMan. Each window could provide a different zoom with different map operations performed on it. This would allow the user to manipulate the maps in a more professional manner.

Creating the map as a widget would simplify this process. Each separate window could have a separate instance of the widget. They could all share trace data, but each could have a local copy of the map data.

The functionality of the multiple displays can be likened to that of *ghostview* [Theisen, 1989]. When a zoom is selected, it is displayed in a separate window. The second window can have all of the basic functions performed on it. Both windows are active at all times.

10.1.10 Improved Trace Displaying

The displaying of trace data should be improved. The current method is acceptable, but in situations where too much data is to be displayed around the one site, it overwrites itself. A method to overcome this should be sort.

Other problems with the trace displaying is the length of site names. Often, the names of sites close together become unreadable as they overlap. This is merely aesthetics, but the production of software destined for release should address issues such as this. The popup list of sites in the trace may alter how sites are displayed on the map.

10.1.11 New Version of Traceroute

As was discussed briefly in section 4.1.2 Traceroute, a new version of *traceroute* was discussed in RFC 1393 [Malkin, 1993]. The implementation of this new *traceroute* would improve the efficiency of GeoTraceMan, but the possibility of not receiving any trace data limits its use. If a routing error is encountered, no trace data will be returned. Due to this limitation, it is unlikely this *traceroute* will replace the original.

10.3 Conclusions From This Project

The main conclusion that was obtained from this project is related to time. There is never enough time to do everything. As can be seen from this report, the possible extensions outweigh the actual achievements. This is an excellent method of deflating the ego.

The three dimensional extensions to GeoTraceMan were suitable for an undergraduate project. It was enough work to create large amounts of stress and many *all-nighters*, yet it was still completed in the time allotted. It was not completed to my satisfaction, but soon will with the continued development outlined in this report.

A lesson learnt from this project is knowing whom to ask at what stage. Much of the work in this project would not have been completed without the help of others. Consultation with experts in the field is invaluable.

The preliminary release of GeoTraceMan provided some useful insights into the workings of the Internet. The process of making software available for *ftp* was explored when GeoTraceMan was added to the Netman distribution tree. Advertising the release was also explored.

This project achieved its aims. As such, it can be considered successful. This success is mostly due to others who helped, I simply combined their knowledge and produced code from it.

11.0 References

Albitz, P. & C. Liu, DNS and BIND, O'Reilly and Associates, Inc., California, 1992

Barron, G., Packetman v1.1, Technical Report, Department of Computer Science, Curtin University of Technology, 1993

Bartholomew, J., Bartholomew Atlas of the World, John Bartholomew & Son Ltd, Edinburgh, Scotland, 1978

Brown, D., Geographical Trace Route v1.0, Technical Report, Department of Computer Science, Curtin University of Technology, 1993

Burrough, P.A., Principles of Geographical Information Systems for Land Resource Assessment, Oxford Science Publications, New York, 1991

Castleman, K.R., Digital Image Processing, Prentice Hall Englewood Cliffs, New Jersey, 1979

Comer, D.E., Internetworking with TCP/IP, Volume I, Principles, Protocols, and Architectures, Prentice Hall, Englewood Cliffs, New Jersey, 1991

Dellinger, J., README file supplied with the CIA World Map Databank, Stanford University, 1989

Echidna, E, VOGL - Very Ordinary Graphics Library, Department of Engineering Computing Resources, University of Melbourne, 1991

Everhart, C. (Transarc), L. Mamakos (University of Maryland), R. Ullmann (Prime Computer), P. Mockapetris (ISI), New DNS RR Definition, RFC 1183, October, 1990

Farrell, C., M. Schulze, S. Pleitner, D. Baldoni, DNS Encoding of Geographical Location, RFC 1712, November, 1994

Finkelstein, A. & D.H. Salesin, Multiresolution Curves, Proceedings from SIGCOM 1994, Department of Computer Science and Engineering, University of Washington, 1994

Halsall, F., Data Communications, Computer Networks and Open Systems, Addison-Wesley, Wokingham, England, 1992

Hook, D.G., The *hershey fonts* Manual Page, 1992

Jacobson, V., The *traceroute* Manual Page, 1989

Kernighan, B.W, D.M. Ritchie, THE C PROGRAMMING LANGUAGE, Prentice Hall, 1978

Kernighan, B.W, D.M. Ritchie, THE C PROGRAMMING LANGUAGE: ANSI C, Prentice Hall, 1990

Maling, D.H., COORDINATE SYSTEMS AND MAP PROJECTIONS, George Philip & Son Limited, London, 1973

Malkin, G., TRACEROUTE USING AN IP OPTION, RFC 1393, Xylogics Inc., January, 1993

Mercer, J., C. Farrell, M. Schulze, ACCURATE NETWORK LOAD MONITORING: LOADING, Technical Report No. 1, Department of Computer Science, Curtin University of Technology, 1994

Mockapetris, P., DOMAIN NAMES - CONCEPTS AND FACILITIES, RFC 1034, ISI, November, 1987

Mockapetris, P., DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, RFC 1035, ISI, November, 1987

Mockapetris, P., DNS ENCODING OF NETWORK NAMES AND OTHER TYPES, RFC 1101, ISI, April, 1989

Mogul, J., EFFICIENT USE OF WORKSTATIONS FOR PASSIVE MONITORING OF LOCAL AREA NETWORKS, Proceeding of the 11th Symposium on Operating Systems Principles, ACM SIGOPS, Austin, Texas, November, 1987

Nemeth, E., G. Snyder and S. Seebas, UNIX SYSTEM ADMINISTRATION HANDBOOK, Prentice Hall, Englewood Cliffs, New Jersey, 1989

Pure Software Inc., THE *purify* MANUAL PAGE, 1993

Robinson, A.H., R.D. Sale, J.L. Morrison, P.C. Muehrcke, ELEMENTS OF CARTOGRAPHY, 5TH EDITION, John Wiley & Son, New York, 1984

Schalkoff, R.J., DIGITAL IMAGE PROCESSING AND COMPUTER VISION, John Wiley and Sons, New York, 1989

Samet, H., THE DESIGN AND ANALYSIS OF SPATIAL DATA STRUCTURES, Addison-Wesley, Reading, Massachusetts, 1989

Schulze, M., G. Benko, C. Farrell, HOMEBREW NETWORK MONITORING: A PRELUDE TO NETWORK MANAGEMENT, Department of Computing Science, Curtin University of Technology, 1993

Sedgewick, R., ALGORITHMS, Addison-Wesley, Reading, Massachusetts, 1988

Silicon Graphics Inc., GRAPHICS LIBRARY REFERENCE MANUAL, *C* EDITION, DOCUMENTATION VERSION 4.0, Silicon Graphics Inc., Mountain View, California, 1990

Sommerville, I., SOFTWARE ENGINEERING, Addison-Wesley, Wokingham, England, 1992

Theisen, T., THE *ghostview* MANUAL PAGE, 1989

Tichy, W.F., RCS - A SYSTEM FOR VERSION CONTROL, Department of Computer Science, Perdue University, West Lafayette, Indiana, 1991

Totty, B., FREE WIDGET FOUNDATION RELEASE, Department of Computer Science, University of Illinois at Urbana-Champaign, 1994

Appendix A - Draft GPOS RFC

DNS Encoding of Geographical Location

Craig Farrell, Mike Schulze, Scott Pleitner and Daniel Baldoni
Department of Computer Science
Curtin University of technology
Perth, Western Australia.

August 1994.

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

This document defines the format of a new Resource Record (RR) for the Domain Naming System (DNS), and reserves a corresponding DNS type mnemonic and numerical code. This definition deals with associating geographical host location mappings to host names within a domain. The data shown in this document is fictitious and does not necessarily reflect the real Internet.

1. Introduction

It has been a long standing problem to relate IP numbers to geographical locations. The availability of Geographical location information has immediate applications in network management. Such information can be used to supplement the data already provided by utilities such as whois[Har85], traceroute[VJ89], and nslookup[UCB89]. The usefulness and functionality of these already widely used tools would be greatly enhanced by the provision of reliable geographical location information.

The ideal way to manage and maintain a database of information, such as geographical location of internet hosts, is to delegate responsibility to local domain administrators. A large distributed database could be implemented with a simple mechanism for updating the local information. A query mechanism also has to be available for checking local entries, as well as inquiring about data from non-local domains.

2. Background

The Internet continues to grow at an ever increasing rate with IP numbers allocated on a first-come-first-serve basis. Deciding when and how to setup a database of geographical information about internet hosts presented a number of options. The uumap project [UU85] was the first serious attempt to collect geographical location data from sites and store it centrally. This project met with limited success because of the difficulty in maintaining and updating a large central database. Another problem was the lack of tools for the checking the data supplied, this problem resulted in some erroneous data entering the database.

2.1 SNMP:

Using an SNMP get request on the sysLocation MIB (Management Information Base) variable was also an option, however this would require the host to be running an appropriate agent with public read access. It was also felt that MIB data should reflect local management data (eg "this" host is on level 5 room 74) rather than a hosts geographical position. This view is supported in the examples given in literature in this area [ROSE91].

2.2 X500:

The X500 Directory service[X.500.88] defined as part of the ISO standards also appears as a potential provider of geographical location data. However due to the limited implementations of this service it was decided to defer this until this service gains wider use and acceptance within the Internet community.

2.3 BIND:

The DNS[Mock87a][Mock87b] represents an existing system ideally suited to the provision of host specific information. The DNS is a widely used and well-understood mechanism for providing a distributed database of such information and its extensible nature allows it to be used to disseminate virtually any information. The most commonly used DNS implementation is the Berkeley Internet Name Domain server BIND[UCB89]. The information we wished to make available needed to be updated locally but available globally; a perfect match with the services provided by the DNS. Current DNS servers provide a variety of useful information about hosts in their domain but lack the ability to report a host's geographical location.

3. The GPOS RR

The geographical location is defined with the mnemonic GPOS and type code 24.

GPOS has the following format:

<owner> <ttl> <class> GPOS <decimal-northing> <decimal-easting>

A decimal format was chosen to specify geographical locations for reasons of simplicity. This also guarantees a concise unambiguous description of a location by enforcing two compulsory numerical values to be specified.

The owner, ttl, and class fields are optional and default to the last defined value if omitted. The decimal_northing is a decimal number ranging from -90 to 90 specifying a hosts latitude on the globe. For example Perth, Western Australia is located at 32ˆ 7‘ 19" south of the equator which would be specified as -32.68820. The decimal easting is a number ranging from -180.0 to 180.0 specifying a hosts geographical longitude. For example Perth, Western Australia is located at 116ˆ 2' 25" east of 0ˆ longitude which would be specified as 116.86520.

The valid GPOS record for a host in Perth Western Australia would therefore be:

GPOS -32.68820 116.86520

There is no limit imposed on the number of decimal places.

4. Master File Format

Each host requires its own GPOS field in the corresponding DNS RR to explicitly specify its geographical location. If the GPOS field is omitted, a DNS enquiry will return no position information for that host.

Consider the following example:

```
; Authoritative data for cs.curtin.edu.au.
;
@       IN      SOA     marsh.cs.curtin.edu.au. postmaster.cs.curtin.edu.au.
                (
                        94070503        ; Serial (yymmddnn)
                        10800           ; Refresh (3 hours)
                        3600            ; Retry (1 hour)
                        3600000         ; Expire (1000 hours)
                        86400           ; Minimum (24 hours)
                )

                IN      NS      marsh.cs.curtin.edu.au.

marsh           IN      A       134.7.1.1
                IN      MX      0       marsh
                IN      HINFO   SGI-Indigo IRIX-4.0.5F
                IN      GPOS    -32.68820 116.86520
ftp             IN      CNAME   marsh

lillee          IN      A       134.7.1.2
                IN      MX      0       marsh
                IN      HINFO   SGI-Indigo IRIX-4.0.5F
                IN      GPOS    -32.68820 116.86520

hinault         IN      A       134.7.1.23
                IN      MX      0       marsh
                IN      HINFO   SUN-IPC SunOS-4.1.3
                IN      GPOS    -22.68820 116.86520

merckx          IN      A       134.7.1.24
                IN      MX      0       marsh
                IN      HINFO   SUN-IPC SunOS-4.1.1

ambrose         IN      A       134.7.1.99
                IN      MX      0       marsh
                IN      HINFO   SGI-CHALLENGE_L IRIX-5.2
                IN      GPOS    -32.68820 116.86520
```

The hosts marsh, lillee, and ambrose are all at the same geographical location, Perth Western Australia (-32.68820 116.86520). The host hinault is at a different geographical location, 10 degrees north of Perth (-22.68820 116.86520). For security reasons we do not wish to give the location of the host merckx.

Although the GPOS clause is not a standard entry within BIND configuration files, most vendor implementations seem to ignore whatever is not understood upon startup of the DNS. Usually this will result in a number of warnings appearing in system log files, but in no way alters naming information or impedes the DNS from performing its normal duties.

5. Security Considerations

Once information has been entered into the DNS, it is considered public.

6. Author's Addresses

postal: Craig Farrell, Mike Schulze, Scott Pleitner and Daniel Baldoni
Department of Computer Science
Curtin University of technology
GPO Box U1987 Perth,
Western Australia.

email: craig@cs.curtin.edu.au
mike@cs.curtin.edu.au
pleitner@cs.curtin.edu.au
flint@cs.curtin.edu.au

7. References

[ROSE91] Rose M., "The Simple Book: An Introduction to Management of TCP/IP-based Internets", Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[X.500.88] CCITT: The Directory - Overview of Concepts, Models and Services", Recommendations X.500 - X.521

[Har82] Harrenstein K, Stahl M.K. and Feiner E.J., "NICNAME/WHOIS" RFC 812, March 1982.

[Mock87a] Mockapetris P.V., "Domain Names - Concepts and Facilities", RFC1034, Nov 1987.

[Mock87b] Mockapetris P.V., "Domain Names - Implementation and specification", RFC1035, Nov 1987.

[FRB93] Ford P.S, Rekhter Y. and Braun H-W. "Improving the Routing and Addressing of IP", IEEE Network Vol7. No 3. pp 11-15 May 1993.

[VJ89] Jaconsen V., The traceroute(8) manual page. Lawrence Berkeley Laboratory, Berkeley, CA, February 1989.

[UCB89] University of California, "BIND: Berkeley Internet Name Domain server", 1989.

[UU85] UUCP Mapping Project, Software available via anonymous ftp from ftp.uu.net., 1985.

Appendix B - Draft LOC RFC

```
Network Working Group                                           C. Davis
Request for Comments: LOCN                             Kapor Enterprises
Updates: RFCs 1034, 1035                                        P. Vixie
                                                       Vixie Enterprises
                                                              T. Goodwin
                                                                   PIPEX
                                                            I. Dickinson
                                                 University of Warwick
```

A Means for Expressing Location Information in the Domain Name System

Status of this Memo

This memo defines a new DNS RR type for experimental purposes. This RFC describes an Experimental Protocol for the Internet community and does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Introduction

This RFC defines the format of a new Resource Record (RR) for the Domain Name System (DNS), and reserves a corresponding DNS type mnemonic (LOC) and numerical code (XXX to be assigned XXX). The definition is experimental.

This RFC assumes that the reader is familiar with the DNS [RFC 1034, RFC 1035]. The data shown in our examples is for pedagogical use and does not necessarily reflect the real Internet.

RDATA Format

```
       MSB                                           LSB
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      0|   VERSION  |    SIZE    | HORIZ PRE |  VERT PRE |
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      2|                   LATITUDE                     |
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      4|      LATITUDE        |        LONGITUDE        |
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      8|                   LONGITUDE                    |
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     10|                   ALTITUDE                     |
       +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
     12|      ALTITUDE        |
       +--+--+--+--+--+--+--+--+
```

(octet)

where:

VERSION Version number of the representation. This must be zero. Implementations are required to check this field and make no assumptions about the format of unrecognized versions.

SIZE The size of the described entity, in log10(meters), expressed as a signed 4-bit integer. For hosts, this will often be zero (i.e. 1m). Larger entities, such as multinational companies' domains, will use larger values. The largest possible value is 7 and the smallest possible value is -7.

HORIZ PRE The horizontal precision of the data, in log2(meters), expressed as an unsigned 4-bit integer. This is treated as an "accurate to within N meters" value. For example, if the latitude and longitude have been determined using differential GPS, this value would be 0 (i.e. 1m). If survey maps and building blueprints are used, the value might be 3 (i.e. 8m). If standard values for the metropolitan area are used, the value could be as high as 15 (32.7km).

VERT PRE The vertical precision of the point described by the (LATITUDE, LONGITUDE, ALTITUDE) tuple, in log2(meters), expressed as an unsigned 4-bit integer. This is treated as an "accurate to within N meters" value.

LATITUDE The latitude of the center of the sphere described by the SIZE field, expressed as a signed 24-bit integer, most significant octet first, in 24ths of a second of arc, rounded away from the equator. Positive numbers represent locations north of the equator.

LONGITUDE The longitude of the center of the sphere described by the SIZE field, expressed as a signed 24-bit integer, most significant octet first, in 12ths of a second of arc, rounded away from the prime meridian. Positive numbers represent locations east of the prime meridian.

ALTITUDE The altitude of the center of the sphere described by the SIZE field, expressed as an unsigned 24-bit integer, most significant octet first, in units of 3 meters, from a base of 3,000,000m below median sea level, rounded upward. This measurement is relative to the [WGS 84] reference spheroid used by GPS (semimajor axis a=6378137.0, reciprocal flattening rf=298.257223563).

Master File Format

The LOC record is expressed in a master file in the following format:

```
<owner> <TTL> <class> LOC ( d1 m1 s1 {"N"|"S"} d2 m2 s2 {"E"|"W"}
                            hp["m"] alt["m"] vp["m"] siz["m"] )
```

(The parentheses are used for multi-line data as specified in [RFC 1035] section 5.1.)

where:

```
d1:        [0 .. 90]       (degrees latitude)
d2:        [0 .. 180]      (degrees longitude)
m1, m2:    [0 .. 59]       (minutes latitude/longitude)
s1, s2:    [0.00 .. 60-~0] (seconds latitude/longitude)
hp, vp:    "1" | "2" | "4" | "8" | "16" | "32" | "64" | "128" |
            "256" | "512" | "1024" | "2048" | "4096" | "8192" |
            "16384" | "32768" (horizontal/vertical precision)
alt:       [-3000000 .. 47331648] BY 3 (altitude in meters)
siz:       "0.0000001" | "0.000001" | "0.00001" | "0.0001" |
            "0.001" | "0.01" | "0.1" | "1" | "10" | "100" |
            "1000" | "10000" | "100000" | "1000000" |
            "10000000" (size in meters)
```

Suggested Uses

Some uses for the LOC RR have already been suggested, including the USENET backbone flow maps, a "visual traceroute" application showing the geographical path of an IP packet, and network management applications that could use LOC RRs to generate a map of hosts and routers being managed.

Optional Search Algorithm

If an application wishes to have a "fallback" behavior, displaying a less precise or larger area when a host does not have a LOC RR, it should use the algorithm in this section. This fallback behavior is OPTIONAL and may need to be modified based on the specific requirements of the application involved.

If the application is working from an IP address (as a "visual traceroute" application might be), it should map the address to a name using the IN-ADDR.ARPA namespace, then check for a LOC RR associated with that name. If there is no LOC RR for the name, [RFC 1101] should be followed to find a network or subnet name associated with the IP address. If such a name exists, it should be checked for a LOC RR, which can be used as an approximation of the host's location.

If the application has a host name, rather than an IP address (as the USENET backbone flow maps do), it should check for a LOC RR associated with the name. (CNAME records should be followed as for any other RR type.) If there is no LOC RR for the name, all A records associated with the name (if any) should be checked as above.

Note that MX-only hosts must have a LOC RR associated with their name, as no A records will exist.

Sample Conversion Routines

XXX needs to be written XXX

Applicability to non-IN Classes

The LOC record is defined for all RR classes, and may be used with non-IN classes such as HS and CH. The semantics of such use are not defined by this memo.

References

[RFC 1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, USC/Information Sciences Institute, November 1987.

[RFC 1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, USC/Information Sciences Institute, November 1987.

[RFC 1101] Mockapetris, P., "DNS Encoding of Network Names and Other Types", RFC 1101, USC/Information Sciences Institute, April 1989.

[WGS 84] United States Department of Defense; DoD WGS-1984 - Its Definition and Relationships with Local Geodetic Systems; Washington, D.C.; 1985; Report AD-A188 815 DMA; 6127; 7-R-138-R; CV, KV;

Security Considerations

High-precision LOC RR information could be used to help plan a penetration of physical security, leading to potential denial-of-machine attacks.

Authors' Addresses

Christopher Davis
Kapor Enterprises, Inc.
238 Main Street, Suite 400
Cambridge, MA 02142

Phone: +1 617 576 4532

Email: ckd@kei.com

Paul Vixie
Vixie Enterprises
116 Stanley Street
Redwood City, CA 94062

Phone: +1 415 361 1937

Email: paul@vix.com

Tim Goodwin
PIPEX Ltd
216 The Science Park
Cambridge CB4 4WA
UK

Phone: +44 223 250250

Email: Tim.Goodwin@pipex.net

Ian Dickinson
Computing Services
University of Warwick
Coventry CV4 7AL
UK

Phone: +44 203 524217

Email: vato@csv.warwick.ac.uk

Appendix C - GeoTraceMan Manual Page

```
NAME
        geotraceman

SYNOPSIS
        geotraceman [-xrm resource]

DESCRIPTION
        GeoTraceMan (or Geographical Traceroute) is an X based utitlity which
        uses the Traceroute(8) program to generate host locations, and then
        plots these locations onto a map of the Earth (Mercator projection).
        Due to Traceroutes's need to open a raw socket, GeoTraceMan can only be
        used by root for route analysis (or bu using setuid).  All users can
        make use of the mapping functions as well as the displaying of
        previously saved traces.

OPTIONS
        -xrm resource
                Use this resource.

GRAPHICAL DISPLAY
        GeoTraceMan's display can be divided into a number of key areas:
                - Menu Options
                - Trace Command Buttons
                - Map Command Buttons
                - Statistics Boxes
                - Panner and Rotator Buttons
                - Main Display

        Menu Options:
                Quit - causes the application to exit nicely
                Map - includes functions such as map rotation, map resetting
                      and the selection of 3 dimensional map display
                Include - toggles the map include options: rivers, political
                          boundaries and cities
                Label - toggles the trace host labels.  The label can be a
                        combination of the hosts IP number, name or its
                        numerical position in the trace
                File - allows the saving or loading of a trace to or from a
                       file
                Misc - toggles the DNS lookup option as well as allowing a
                       postscript dump of the currently displayed trace

        Trace Command Buttons:
```

```
          Start - spawns Traceroute(8)
          Kill - kills the current trace
          Remove - removes the displayed trace from memory as
                   well as the screen

        Map Command Buttons:
          Zoom - Redundant.  Allows the user to zoom by drawing
                 a bounding box on the map
          Undo - undo the previous zoom operation.  Unlimited
          Reset Map - resets the entire map.  Same as the menu
                      option

        Statistics Boxes:
          Trace Statistics - displays statistics as they are
                             gathered about the current trace
          Map Statistics - display statistics about the current
                           state of the map

        Panner and Rotator Buttons:
          Not yet functional.  The panner is to be used to display the
          current position on the map when zoomed.  The rotator buttons
          are to be used to allow rotation without having to use the menu

        Main Display:
          This is where the world map and traces are displayed.  A
          bounding box can be drawn at any time by pressing and holding
          the left mouse button and dragging it to the required size.
          Releasing the mouse button then performs the zoom.

FILES
        ./cities.file.*         Names of the cities to be displayed at the
                                differing levels of zoom
        ./maps/gtm_*.*          Map data files.
        ./GeoTraceMan.ad        Application defaults file for GeoTraceMan.  It
                                is not required, but can be used to customise
                                GeoTraceMan.

SUPPORTING LIBRARIES
        X11 and Xtoolkit Libraries by MIT X Consortium
        Xaw3d Library by Kaleb Keithley, NASA Jet Propulsion Labs
        EzMenu Library by MIT X Consortium, Dirk Grunwald and George Ferguson

SEE ALSO
        traceroute(8), interman(1), etherman(1), packetman(1), x(1)
```

BUGS

This application is currently under development. The are a number of problems which are not worth going into here as many will be fixed by the time it is released.

Yes there are bugs. Please send any reports to netman@cs.curtin.edu.au