A Distributed Approach to Passively Gathering End-to-End Network Performance Measurements

A Thesis
Presented to
The Academic Faculty

by

Charles Robert Simpson, Jr.

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2004

A Distributed Approach to Passively Gathering End-to-End Network Performance Measurements

Approved by:

Professor George F. Riley, Advisor

Professor Henry L. Owen

Professor John A. Copeland

Professor Constantinos Dovrolis
(College of Computing)

Date Approved: 7 April 2004

To my loving parents,

Bobby and Sandra Simpson,

without whose love, support, and encouragement I would have never made it this far.

And to my sister,

Jodi (Dee Dee),

who first taught me how to sit through class (and the value of education).

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. George Riley, for his help and guidance in the completion of this project. I would also like to thank my committee for their diligent and thoughtful review.

The love and support given to me by my family has always been wonderful and this thesis, as with much throughout my life, would not have been possible without them.

I have also received lots of support from my lab-mates, especially Dheeraj Reddy, throughout this time, and they deserve many thanks!

My friend and roommate Julian Grizzard has also been very helpful, especially with those late-night ponderings.

Finally, I would like to thank everyone else who has helped with the successful completion of this thesis, including everyone who participated in the code review.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

NETI@home is an open-source software package that collects network performance statistics from end-systems. It has been written for and tested on the Windows, Solaris, and Linux operating systems, with testing for other operating systems to be completed soon. NETI@home is designed to run on end-user machines and collect various statistics about Internet performance. These statistics are then sent to a server at the Georgia Institute of Technology, where they are collected and made publicly available. This tool gives researchers much needed data on the end-to-end performance of the Internet, as measured by end-users. NETI@home's basic approach is to sniff packets sent from and received by the host and infer performance metrics based on these observed packets. NETI@home users are able to select a privacy level that determines what types of data are gathered, and what is not reported. NETI@home is designed to be an unobtrusive software system that runs quietly in the background with little or no intervention by the user, and using few resources.

CHAPTER 1

INTRODUCTION

The Internet has become an increasingly essential part of life for almost everyone. To accommodate this growth and increased interest, much research needs to be completed on the behavior of the Internet and Internet users as well as the overall performance of this vast global network. In response to this need, network measurements have become a hot topic in Internet research.

The area of network measurements has recently become a major focus for those who wish to have an understanding of the traffic patterns of the Internet. The analysis of these measurements has led to better models for traffic flows, file sizes, burst sizes, and many other complex characteristics of the Internet. Most of the sampling techniques for this data have come either from active measurements (for example, pinging) or from localized passive measurements (for example, sniffing). It has been documented that active measurements introduce bias into these measurements, and many claim that this bias is so great that some measurements collected are not representative of actual Internet traffic [2][16]. As for the passive measurements that have been conducted, they are only able to analyze a small percentage of the Internet and cannot give a good representation of the end-user experience as they are only collected from a small number of hosts. It is important to be able to collect measurements from the perspective of the end-user because such a perspective gives an excellent insight into the "real" use of the network. Thus, there is a need for large-scale, end-to-end, passive network measurements.

Another major issue for the large-scale collection of passive end-to-end network measurements is the privacy of the end users. Any collection done on an end-user's system must not invade their privacy whatsoever. Should their privacy be infringed upon, at the very least it would spell the end for such a large-scale collection system.

This thesis introduces the NETI@home (NETwork Intelligence) software package, a distributed approach to passively gathering end-to-end network performance measurements. NETI@home was written to capture the experience of regular users of the Internet, addressing the need for end-to-end network measurement data. It is designed to run on end-user computers with a variety of operating systems and to require little or no intervention by the user. NETI@home collects an assortment of network measurements from these machines and then sends the results to Georgia Tech for analysis.

The following thesis first presents some related work and other background information relevant to the NETI@home project. Next, motivation for a system such as NETI@home is given. Then, an overall description of the NETI@home software is presented, followed by a detailed description of the statistics collected by the NETI@home software. Implementation details follow the description of the statistics. A discussion of the distribution of the NETI@home software is given in Chapter 7. Finally, a brief discussion of the currently collected data is given followed by the conclusion and areas of future work. The appendices present a listing of the files comprising the NETI@home software package, followed by their associated copyright license.

CHAPTER 2

BACKGROUND AND RELATED WORK

There are essentially two methods of gathering network measurements: active and passive measurement. Active measurements [19][5] involve interacting with the network to make measurements, usually by sending probe packets. Passive measurements [22][6][13] do not inject any traffic into the network; they merely monitor traffic on the network and infer measurements from the observed traffic. Many studies have compared active and passive measurements [2][16] with both methods having advantages and disadvantages.

Active measurements have been used since the early days of the Internet. Some popular active measurement tools are ping and traceroute [9]. While active measurements provide meaningful data in some cases, there are many measurements that cannot be feasibly made using active techniques, such as accurate statistics on packet loss [2]. Active measurements can also introduce a large amount of bias into the measured system, since they actually inject, and thus interact with, the measured traffic.

Vern Paxson has proposed the use of a National Internet Measurement Infrastructure (NIMI) [19], which will actively collect data from many points strewn across the Internet. However, if this infrastructure can ever be put in place, it will still be unable to give an entirely accurate representation of the end-user experience, and it suffers from the unwillingness of Internet service providers (ISPs) to install such measurement devices on their networks.

Passive measurements, on the other hand, have the goal of minimally affecting the measured network. The most popular passive measurement tools are sniffer based tools such as tcpdump [11] and Ethereal [4], which actually sample every packet that is seen on the link (in promiscuous mode) or every packet that is sent from or received by the host that is sniffing (without promiscuous mode). Using passive measurements, one is able to see the actual users' experiences, if the passive measurements are made at the end host. Other

forms of passive measurements observe the network at a point that is between the two end hosts. Two such proposed systems are OC3MON [22] and IPMON [6]. In addition, many studies have analyzed Internet traffic from a variety of points, studying different metrics such as round-trip times (RTTs), available bandwidth, packet loss, and various aspects of protocols such as TCP (receiver window sizes, throughput, time to live values, etc.). Internet Service Providers (ISPs) also collect many network measurements that would be useful to the research community for analysis [22][6]. However, most ISPs are reluctant to release such information since it could potentially expose problems in their own networks [18].

To our knowledge, there have been no studies on the large-scale, distributed collection of passive end-to-end network measurements such as that of NETI@home.

The distributed approach to the collection of network measurements was inspired by the SETI@home project [20], NETI@home's namesake. Although SETI@home does not deal with network measurements (it actually looks for signs of intelligent life in the universe), it was one of the first programs to rely on regular users of the Internet to perform a data related function and then report back to a central server. Since its introduction, SETI@home has become extremely popular. NETI@home hopes to capitalize on this popularity and novel technique for the collection of network measurements.

CHAPTER 3

PROBLEM STATEMENT

The main goal of this thesis is the construction of a software system to collect network measurements from end-users on the Internet. These measurements are to be collected passively and are to be collected from "normal" Internet users so that their use of the Internet can be analyzed.

Effort should be devoted to protecting the privacy of the users of the software system. User privacy should be of utmost importance and users should be assured that their privacy is respected. At the same time, research potential should be maximized.

A large variety of network measurements should be collected for the most commonly used Internet protocols. The software system should minimally affect the user and the user's system, so that it will have little impact on the collected measurements and not frustrate users.

To facilitate the collection of a large number of measurements, this system should have as large a user base as possible. To this end, the software system should run on multiple platforms, especially the most common platforms used by Internet end-users. The system should also run in the background, requiring little or no intervention on the part of its users, and using few resources. Effort should also be made to distribute such a system to further increase the amount of measurements gathered. Motivation for participation should be considered.

Once these measurements are collected, they are to be sent to Georgia Tech for analysis. The collected data should also be made public, so that other network researchers will have access to this valuable data. The collection method should also be scalable.

Further, this system should be easily upgraded to allow for bug fixes, new measurements, and changes in existing measurement collection techniques.

In response to these needs, the NETI@home software has been created - a passive,

end-to-end network measurement system.

CHAPTER 4

DESCRIPTION OF NETI@HOME SOFTWARE

NETI@home is written in the C++ programming language and is built on top of the open-source Ethereal [4] software package. Ethereal is a well-written network protocol analyzer (sniffer) that is becoming the de facto network sniffer standard. A sniffer is software that captures the raw packets observed by the host machine on which it is running. Once Ethereal sniffs a network packet, the packet is then sent to the NETI@home software for analysis. The packets are sorted into bidirectional flows, as defined in [3]. These packets are *not* collected with the network interface(s) in promiscuous mode, therefore only packets sent from and received by the user's system are analyzed. Packets are not collected in promiscuous mode to eliminate the possibility of duplicate measurements, to respect the rights and privacy of others, and to guarantee the collection of end-to-end measurements.

Once NETI@home analyzes a specified number of flows, the data is compressed using the zlib compression library [15]. This compressed data is then sent to a server, `neti.ece.gatech.edu`, here at the Georgia Institute of Technology (Georgia Tech) via TCP. This server is carefully monitored and will be upgraded if necessary. A duplicate copy of the binary data sent is stored on the user's system in a log file. The user can observe what data is actually transmitted using the included NETILogParse program. The collected packets at Georgia Tech are placed into a publicly accessible database. Researchers are then able to sort the data via a web interface to look for specific data and/or trends. This provides the much needed end-to-end passive measurements that researchers have been wanting.

NETI@home is copyrighted (copylefted) under the GNU General Public License (GPL). The GNU GPL specifies that NETI@home and its derivatives will remain free, open-source software. NETI@home is copyrighted under the GNU GPL so that users and researchers will be able to examine the source code for NETI@home and make suggestions for improvements,

determine exactly how the measurements are collected, and recognize that user privacy is respected. The GNU GPL further specifies that any software created using code from NETI@home also must be copyrighted under the GNU GPL. The full text of the GNU GPL is available in Appendix B for convenience.

4.1 Ethereal

NETI@home uses the open-source Ethereal network protocol analyzer to collect packets for analysis. Ethereal is built using the pcap library [10] and has an active and dedicated developer base. NETI@home uses Ethereal's ability to sniff packets in real-time with little impact on the user's system. Should Ethereal be found to be unsuitable, another network protocol analyzer can easily be used, such as tcpdump [11], or a custom sniffer can be written using the pcap library [10].

Ethereal has two basic modes of operation, a graphical user interface and a command line interface, called tethereal. NETI@home uses the tethereal interface to reduce the amount of strain on the end-user system. Options are also passed to tethereal so that it only captures packets which use the TCP, UDP, ICMP, or IGMP protocols, collects packets from all interfaces except the loopback interface, and places the network interface(s) in non-promiscuous mode.

Ethereal is available from their website, located at `http://www.ethereal.com/`.

4.2 Privacy Levels

NETI@home users are able to specify a privacy level that determines which statistics are collected and reported. Currently, there are three privacy settings: high, medium, and low.

At the highest privacy setting, neither the source, destination, nor any multicast IP addresses are recorded. While the vast majority of users would not want to conceal such information, it was thought that it would provide the maximum respect to the users' privacy to include such a setting. However, should this level be selected, information will be lost concerning the location of nodes on the network and much research potential is also lost.

Medium privacy, the default setting, allows NETI@home to collect all statistics with

the exception of the source and destination IP addresses, which are trimmed to contain only the network portion of their addresses. The network portion is defined by what class the IP address is assigned to, either A, B, or C. While class-based IP addresses are largely historical, this is a reasonable guideline for the trimming of the addresses. Thus, the individual computers running NETI@home cannot be identified, but the overall network to which they belong can be identified. This will aid in various studies, particularly those related to the topology of the Internet and the performance between the networks. Users may wish to use this setting if they do not want to be identified by their IP address or if they do not want the particular systems with which they communicate to be identified, for instance, certain web servers.

The low privacy setting allows NETI@home to collect all statistics, including all IP addresses. While some users may not want this information collected, NETI@home users are strongly encouraged to select this privacy setting as it will greatly benefit Internet research.

The privacy level is specified in a human readable configuration file, `neti.conf`. This file resides in the `/etc` directory on all systems except those running Microsoft Windows operating systems. On these Windows systems, the user is able to use the NETITray application to manipulate the configuration file. The NETITray application is further described in the next section.

4.3 NETITray

For users of the Microsoft Windows operating systems, NETI@home includes a program entitled NETITray to ease the configuration of NETI@home. While running NETITray, Windows users are able to right-click on a NETI@home icon in the tray area of the Windows taskbar to select between four options: Properties, About, Stop NETI@home, and Exit, as seen in Figure 1 and Figure 2.

By selecting the Properties menu item, users are able to modify two aspects of the operation of the NETI@home software: the maximum log file size and the privacy level. Users are able to select between one kilobyte and ten megabyte log file sizes, which determines

Figure 1: Screenshot of the NETITray application in the Windows taskbar. By right-clicking on this icon, a user is presented with the NETITray menu, as shown in Figure 2.

Figure 2: Screenshot of the NETITray menu. From this menu, users are able to select between the Properties window (Figure 3), the About window, an option to halt NETI@home's collection of measurements, and an option to close the NETITray application.

the maximum size of the log file stored on the user's system. The default log file size is one megabyte. Users are further able to select between the high, medium, and low privacy levels. The default privacy setting is the medium privacy level. A screenshot of the Properties window is shown in Figure 3.

The About menu item displays a caption briefly describing NETI@home including copyright and version information.

The Stop NETI@home menu item causes NETI@home to halt its gathering of measurements and close. The NETITray application also closes.

The Exit menu item closes the NETITray application without halting the collection of

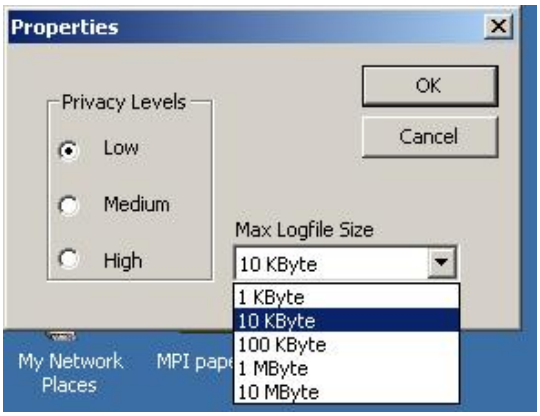

Figure 3: Screenshot of the NETITray properties window. From this window, users are able to select their desired privacy level and maximum log file size.

measurements.

4.4 Installation

NETI@home is available for many different operating systems, and, as such, has a few different methods of installation. For users of Microsoft Windows operating systems, NETI@home is available for installation in the form of a self-extracting executable. This executable was created using the InstallShield Express [8] software package, however, future versions of NETI@home will rely on the open-source and more robust Nullsoft Scriptable Install System (NSIS) [23]. For Redhat Linux systems and compatibles, NETI@home is available in the form of an RPM file. For all other systems, NETI@home is available in tarball format. Included in the tarball are configuration scripts generated by the GNU autoconf and automake tools [7], to ease configuration and installation. Typically for these other systems, one must only execute three commands (`configure`, `make`, and `make install`).

4.5 NETILogParse

The NETI@home software package includes the NETILogParse application for users who are curious or paranoid about the data that is transmitted to the server at Georgia Tech. As binary data is sent to Georgia Tech, a duplicate copy is stored on the user's machine in the form of a log file. Using the NETILogParse application, the user is able to view exactly what is contained in this binary data and further verify that their privacy is protected.

4.6 NETIMap

One of the most important goals of the NETI@home project is to have a large installed user base. To encourage end users to run the NETI@home software a program, written in Java, is included. This program, titled NETIMap, when run in conjunction with the core NETI@home software, maps each remote IP address contacted to a graphical display of the world. This plot allows users to visualize where each remote host with which they communicate is located geographically. A screenshot of the NETIMap application in use is shown in Figure 4.

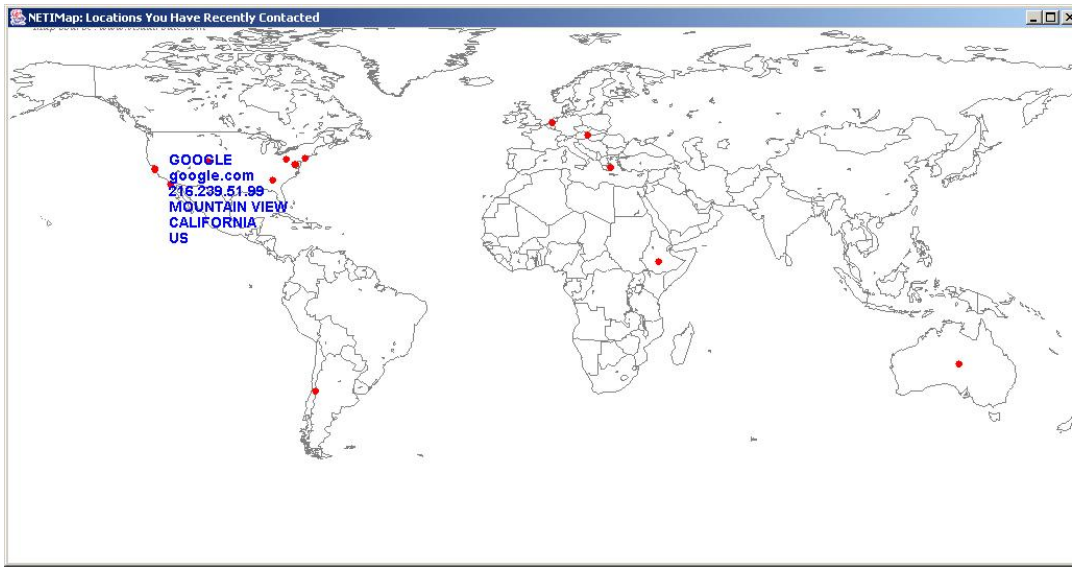

Figure 4: Screenshot of the NETIMap application. In this screenshot several locations have already been contacted and graphed. The mouse cursor is also hovering over the dot representing a connection to `http://www.google.com/`, displaying the information for that website. This information includes the name, DNS entry, IP address, city, state, and country associated with the website, all retrieved from CAIDA's NetGeo [17] database.

To resolve IP addresses into latitude/longitude coordinates, the NetGeo [17] database from CAIDA is used. It should be noted that this database is not entirely accurate. The coordinates returned by the NetGeo database should be viewed as approximations and are based on the records in whois databases, which can be outdated or misleading [17]. Finally, this program is written in Java for maximum portability and for compatibility with the NetGeo database.

CHAPTER 5

NETWORK STATISTICS COLLECTED

NETI@home collects many different statistics from end-users, in accordance with CAIDA specifications [3]. These statistics focus on four transport-layer protocols on the Internet: the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), the Internet Control Message Protocol (ICMP), and the Internet Group Management Protocol (IGMP), as well as their underlying network-layer protocols (such as the Internet Protocol, IP). No application layer data is collected, to maintain user privacy.

The following sections discuss the statistics that are collected by NETI@home on a per-flow basis. These statistics are also summarized in Table 1, Table 2, Table 3, and Table 4 for convenience.

5.1 Transmission Control Protocol

TCP is one of the most common and widely used protocols on the Internet. It is used for reliable end-to-end Internet connections for applications such as world wide web (WWW) traffic, secure shell (SSH) traffic, and file transfer protocol (FTP) traffic. A bidirectional TCP flow is defined by a local IP address and port and a remote IP address and port and represents a communication channel. The following sections briefly discuss the statistics that are collected by NETI@home for TCP flows. These statistics are also summarized in Table 1 for convenience.

5.1.1 IP Addresses

NETI@home is able to collect the source and destination IP addresses for TCP connections directly from the TCP packets.[1] IP addresses uniquely identify an Internet host, much like a telephone number uniquely identifies a telephone.

[1]The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting

5.1.2 Ports

NETI@home is able to collect the source and destination TCP ports directly from the TCP packets. The collection of TCP ports aids researchers in determining what type of application is transmitting the data, such as web (port 80), email (port 25), or ssh (port 22) traffic.

5.1.3 Times

NETI@home records the times that the TCP flow is established and closed. These times are precise to the level of detail allowed by Ethereal, which, in turn, is as precise as libpcap and the user's system allows. Typically, the level of precision is on the order of microseconds.

5.1.4 Checksums

NETI@home is able to determine if an IP or TCP Checksum is correct.[2] A checksum is used to determine if the packet has been corrupted during transmission.

5.1.5 Packets Sent and Received

NETI@home maintains a counter of the number of packets that are sent from the host on which it is running and the number of packets that are received by the host on which it is running.

5.1.6 Bytes Sent and Received

NETI@home maintains a counter of the number of data bytes that are sent from the host on which it is running and the number of data bytes that are received by the host on which it is running.

5.1.7 Number of Acknowledgment Packets

NETI@home maintains a counter of the number of packets it sees with the acknowledgment flag set.

[2]This statistic does not work correctly on systems that perform checksum offloading, such as Windows 2000

5.1.8 Number of Duplicate Acknowledgment Packets

NETI@home determines that duplicate acknowledgment packets have occurred if it sees two sequential acknowledgment packets with the same acknowledgment number. Duplicate acknowledgment packets usually occur as a result of loss.

5.1.9 Number of Triple Duplicate Acknowledgment Packets

NETI@home determines that a triple duplicate acknowledgment has occurred if it sees three sequential acknowledgment packets with the same acknowledgment number. Triple duplicate acknowledgments occur as a result of extreme loss and cause TCP to reduce the amount of information that is sent at a time (by reducing its congestion window).

5.1.10 Number of URG Flags

NETI@home maintains a counter of the number of packets that it observes with the URG flag set. URG flags are used by TCP to specify that a packet contains urgent data.

5.1.11 Number of PUSH Flags

NETI@home maintains a counter of the number of packets that it observes with the PUSH flag set. PUSH flags are used by TCP to notify the receiver to "push" all buffered data to the application.

5.1.12 Number of ECNECHO Flags

NETI@home maintains a counter of the number of packets that it observes with the ECNECHO flag set. ECN is early congestion notification, and this flag is used by TCP to indicate ECN ability and to notify the receiver of congestion.

5.1.13 Number of CWR Flags

NETI@home maintains a counter of the number of packets that it observes with the CWR flag set. The CWR flag works in conjunction with the ECNECHO flag. The CWR flag is set in response to receiving a packet with the ECNECHO flag set. The CWR flag indicates to its receiver that appropriate action has been taken in response to the ECNECHO flag (and thus congestion).

5.1.14 SACK Permitted

NETI@home records if either the sender, receiver, or both the sender and receiver indicate SACK permitted. SACK (selective acknowledgment) is a mechanism used by TCP to acknowledge nonsequential data packets with a single acknowledgment packet.

5.1.15 Minimum, Maximum, and Average Advertised Window Sizes

NETI@home records the minimum, maximum, and average advertised window sizes for both the receiver and the sender. The advertised window size can be directly observed in the TCP packet header and indicates the amount of data a host can receive at a given time.

5.1.16 Minimum, Maximum, and Average TTL Values

NETI@home records the minimum, maximum, and average TTL (time to live) values for both the receiver and the sender. The TTL values can be directly observed in the IP packet header and are used to prevent lost packets from continuously being routed around the network.

5.1.17 Number of Packet Retransmissions

NETI@home maintains a counter of the number of packet retransmissions that it observes. A *retransmission* is defined as a packet with a sequence number less than the highest sequence number that has been observed thus far.

5.1.18 Number of Bytes Retransmitted

NETI@home records the number of bytes that are retransmitted, due to packet retransmissions.

5.1.19 Number of Inactivity Periods

NETI@home maintains a counter of the number of inactivity periods that it observes. An *inactivity period* is defined to be a period in which no packets are sent or received for at least 200 milliseconds. This statistic is a rough estimate of a TCP timeout.

5.1.20 Minimum, Maximum, and Average Round Trip Times

NETI@home records the minimum, maximum, and average round trip times that it estimates. Round trip time estimation is made by keeping a list of all packets for a flow. When an acknowledgment arrives, if its acknowledgment number is equal to one of the packets' sequence numbers plus that packet's length and that packet has not been retransmitted, the round trip time is estimated to be the difference between the time the acknowledgment arrives and when the original packet was sent. However, if the original packet had its SYN flag set (or FIN), then the acknowledgment number should equal its sequence number plus one.

As a final note, this method for round trip time estimation only works if NETI@home is running on the machine from which data is being sent.

5.1.21 Number of Packets Received In and Out of Order

NETI@home maintains counters of the number of packets that are received in-order and out-of-order. A packet is defined to be *in-order* if its sequence number is equal to the sum of the sequence number and length of the packet received immediately before it, and out-of-order otherwise.

5.1.22 Number of Fragmented Packets

NETI@home maintains a counter of the number of fragmented packets that it observes. Fragmented packets are indicated by either the more fragments flag in the IP packet header or by a nonzero fragment offset value in the IP packet header. Fragmented packets are usually a result of a router's need to fragment a large packet that it can only handle in smaller pieces.

5.1.23 Number of Don't Fragment Flags

NETI@home maintains a counter of the number of packets that it observes with the don't fragment flag set.

5.1.24 Maximum Segment Sizes

NETI@home records the maximum segment size (MSS), if reported, for the TCP flow. The MSS is sent from the receiver to the sender at the beginning of the connection to indicate the maximum size of TCP segment that can be received by the receiver.

5.1.25 Number of Rejected Connection Requests

This statistic is not measured directly as is the case with the other statistics, but is rather inferred. A connection is deemed rejected if the TCP three-way handshake procedure fails.

5.1.26 Flow Termination

NETI@home records the method by which the flow was terminated. A flow can either be idle-closed, RST-closed, or FIN-closed.

A connection is defined to be *idle closed* if there has been no activity in the flow (no packets sent or received) for 64 seconds, as per [3].

A connection is defined to be *reset closed* if a packet is sent or received with the RST flag set. NETI@home also records which host sent the packet with the RST flag set, either the local or remote host.

A connection is defined to be *FIN closed* if a packet is sent or received with the FIN flag set. NETI@home also records which host sent the packet with the FIN flag set, either the local or remote host.

5.2 User Datagram Protocol

UDP is another one of the most widely used protocols on the Internet. It is used for unreliable, or timely, end-to-end Internet connections for applications such as Internet telephony (VoIP) and multimedia streaming. A bidirectional UDP flow is defined by a local IP address and port and a remote IP address and port and represents a communication channel. The following sections briefly discuss the statistics that are collected by NETI@home for UDP flows. These statistics are also summarized in Table 2 for convenience.

Table 1: TCP Statistics

Network Performance Statistics
Source IP Address
Destination IP Address
Source Port
Destination Port
Time the Connection was Established
Time the Connection was Closed
Number of Bad IP Checksums
Number of Bad TCP Checksums
Number of Packets Sent
Number of Packets Received
Number of Bytes Sent
Number of Bytes Received
Number of Acknowledgment Packets
Number of Duplicate Acknowledgment Packets
Number of Triple Duplicate Acknowledgment Packets
Number of URG Flags
Number of PUSH Flags
Number of ECNECHO Flags
Number of CWR Flags
Sender SACK Permitted
Receiver SACK Permitted
Sender Minimum Window Size
Receiver Minimum Window Size
Sender Average Window Size
Receiver Average Window Size
Sender Maximum Window Size
Receiver Maximum Window Size
Sender Minimum TTL Value
Receiver Minimum TTL Value
Sender Maximum TTL Value
Receiver Maximum TTL Value
Number of Packet Retransmissions
Number of Bytes Retransmitted
Number of Inactivity Periods
Minimum Round Trip Time
Average Round Trip Time
Maximum Round Trip Time
Number of Packets Received In-Order
Number of Packets Received Out-of-Order
Number of Fragmented Packets
Number of Don't Fragment Flags
Sender Maximum Segment Size
Receiver Maximum Segment Size
Number of Rejected Connection Requests
Method of Flow Termination

5.2.1 IP Addresses

NETI@home is able to collect the source and destination IP addresses for UDP connections directly from the UDP packets.[3] IP addresses uniquely identify an Internet host, much like a telephone number uniquely identifies a telephone.

5.2.2 Ports

NETI@home is able to collect the source and destination UDP ports directly from the UDP packets. The collection of UDP ports aids researchers in determining what type of application is transmitting the data, such as DNS (port 53) or various streaming multimedia traffic.

5.2.3 Checksums

NETI@home is able to determine if an IP or UDP Checksum is correct.[4] A checksum is used to determine if the packet has been corrupted during transmission.

5.2.4 Number of Packets Sent and Received

NETI@home maintains a counter of the number of packets that are sent from the host on which it is running and the number of packets that are received by the host on which it is running.

5.2.5 Minimum, Maximum, and Average Packet Sizes

NETI@home records the minimum, maximum, and average packet sizes for both the receiver and the sender. The packet size can be directly observed in the UDP packet header.

5.2.6 Number of Fragmented Packets

NETI@home maintains a counter of the number of fragmented packets that it observes. Fragmented packets are indicated by either the more fragments flag in the IP packet header or by a nonzero fragment offset value in the IP packet header. Fragmented packets are

[3] The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting

[4] This statistic does not work correctly on systems that perform checksum offloading, such as Windows 2000

Table 2: UDP Statistics

Network Performance Statistics
Source IP Address
Destination IP Address
Source Port
Destination Port
Number of Bad IP Checksums
Number of Bad UDP Checksums
Number of Packets Sent
Number of Packets Received
Average Size of Received Packet
Average Size of Sent Packet
Minimum Size of Received Packet
Minimum Size of Sent Packet
Maximum Size of Received Packet
Maximum Size of Sent Packet
Number of Fragmented Packets
Number of Don't Fragment Flags
Time of the First Packet
Time of the Last Packet

usually a result of a router's need to fragment a large packet that it can only handle in smaller pieces.

5.2.7 Number of Don't Fragment Flags

NETI@home maintains a counter of the number of packets that it observes with the don't fragment flag set.

5.2.8 Times of the First and Last Packets

NETI@home records the times of the first and last packets observed in a UDP flow. These times are precise to the level of detail allowed by Ethereal, which, in turn, is as precise as libpcap and the user's system allows. Typically, the level of precision is on the order of microseconds.

5.3 Internet Control Message Protocol

ICMP is used for managerial purposes on the Internet. ICMP messages are sent when a packets TTL value expires, which is useful for tracerouting, as well as other times such as

when an Internet host in unreachable. A bidirectional ICMP flow is defined by a local IP address and a remote IP address and is closed after an inactivity period of 64 seconds. The following sections briefly discuss the statistics that are collected by NETI@home for ICMP flows. These statistics are also summarized in Table 3 for convenience.

5.3.1 IP Addresses

NETI@home is able to collect the source and destination IP addresses for ICMP connections directly from the ICMP packets.[5] IP addresses uniquely identify an Internet host, much like a telephone number uniquely identifies a telephone.

5.3.2 Checksums

NETI@home is able to determine if an IP or ICMP Checksum is correct.[6] A checksum is used to determine if the packet has been corrupted during transmission.

5.3.3 Number of Fragmented Packets

NETI@home maintains a counter of the number of fragmented packets that it observes. Fragmented packets are indicated by either the more fragments flag in the IP packet header or by a nonzero fragment offset value in the IP packet header. Fragmented packets are usually a result of a router's need to fragment a large packet that it can only handle in smaller pieces.

5.3.4 Number of Don't Fragment Flags

NETI@home maintains a counter of the number of packets that it observes with the don't fragment flag set.

5.3.5 ICMP Type

NETI@home is able to directly record the ICMP type from the ICMP packet header. This field indicates what type of ICMP message the packet represents.

[5]The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting

[6]This statistic does not work correctly on systems that perform checksum offloading, such as Windows 2000

Table 3: ICMP Statistics

Network Performance Statistics
Source IP Address
Destination IP Address
Number of Bad IP Checksums
Number of Bad ICMP Checksums
Number of Fragmented Packets
Number of Don't Fragment Flags
ICMP Type
ICMP Code

5.3.6 ICMP Code

NETI@home is able to directly record the ICMP code from the ICMP packet header. Some ICMP messages (dependent on their ICMP type) have ICMP codes to indicate further information.

5.4 Internet Group Management Protocol

IGMP is used for multicast communications on the Internet. Hosts join a multicast group and each group is assigned a multicast IP address. Thus, when an Internet host wishes to contact an entire multicast group, all they must do is send a packet to a multicast IP address.

NETI@home IGMP statistics are the only statistics not collected on a per-flow basis. Each IGMP packet is recorded and reported by NETI@home. The following sections briefly discuss the statistics that are collected by NETI@home for IGMP packets. These statistics are also summarized in Table 4 for convenience.

5.4.1 IP Addresses

NETI@home is able to collect the source, destination, and multicast IP addresses for IGMP connections directly from the IGMP packets.[7] IP addresses uniquely identify an Internet host, much like a telephone number uniquely identifies a telephone. Multicast IP addresses are assigned to IGMP groups and are used to address the entire multicast group.

[7]The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting

5.4.2 Checksums

NETI@home is able to determine if an IP or IGMP Checksum is correct.[8] A checksum is used to determine if the packet has been corrupted during transmission.

5.4.3 IGMP Version

NETI@home is able to directly record the IGMP version from the IGMP packet header. This field indicates the version of the IGMP protocol used by the packet.

5.4.4 IGMP Type

NETI@home is able to directly record the IGMP type from the IGMP packet header. This field indicates what type of IGMP message the packet represents.

5.4.5 Number of Packets Sent and Received

NETI@home maintains a counter of the number of packets that are sent from the host on which it is running and the number of packets that are received by the host on which it is running.

5.4.6 Number of Fragmented Packets

NETI@home maintains a counter of the number of fragmented packets that it observes. Fragmented packets are indicated by either the more fragments flag in the IP packet header or by a nonzero fragment offset value in the IP packet header. Fragmented packets are usually a result of a router's need to fragment a large packet that it can only handle in smaller pieces.

5.4.7 Number of Don't Fragment Flags

NETI@home maintains a counter of the number of packets that it observes with the don't fragment flag set.

[8] This statistic does not work correctly on systems that perform checksum offloading, such as Windows 2000

Table 4: IGMP Statistics

Network Performance Statistics
Source IP Address
Destination IP Address
Multicast IP Address
Number of Bad IP Checksums
Number of Bad IGMP Checksums
IGMP Version
IGMP Type
Number of Packets Sent
Number of Packets Received
Number of Fragmented Packets
Number of Don't Fragment Flags
Maximum Response Time
Time of the First Packet
Time of the Last Packet

5.4.8 Maximum Response Time

NETI@home records the maximum response time for an IGMP packet. The response time is the amount of time between receiving (or sending) a packet and then sending (or receiving) a packet.

5.4.9 Times of the First and Last Packets

NETI@home records the times of the first and last packets observed in an IGMP flow. These times are precise to the level of detail allowed by Ethereal, which, in turn, is as precise as libpcap and the user's system allows. Typically, the level of precision is on the order of microseconds.

5.5 Miscellaneous

NETI@home also collects a few miscellaneous statistics which are not directly related to a particular transport-layer protocol. However, the collection of these statistics may greatly benefit Internet research. The following sections briefly discuss these miscellaneous statistics. These statistics are also summarized in Table 5 for convenience.

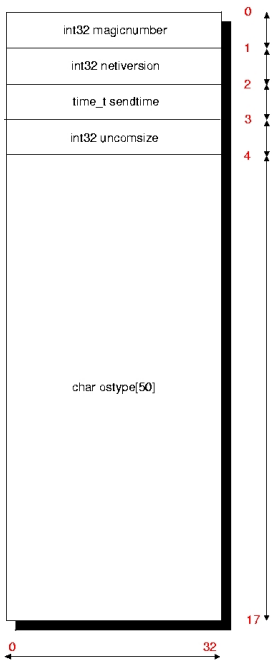

Figure 5: Graphical representation of the header of a NETI@home packet. This header is at the start of all data reported back to the NETI@home server by each user.

5.5.1 NETI@home Version

Each host reports the version of NETI@home they are running as they report their statistics. The version number will aid as protocol statistics are updated, new protocols are added, and as problems are fixed. The NETI@home version number is reported in the NETI@home packet header, as shown in Figure 5.

5.5.2 Operating System

Each host also reports the operating system they are running as they report their statistics. Knowledge of a user's operating system will aid in understanding how each implementation of the various network protocols vary. For all systems other than those running Microsoft operating systems, the GNU autoconf tool [7] is used to determine the operating system type at build time. For systems running Microsoft operating systems, special functions provided with the Windows SDK are used to determine the type and version of the operating system used. The operating system is reported in the NETI@home packet header, as shown in Figure 5.

Table 5: Miscellaneous Statistics

Network Performance Statistics
NETI@home Version
Operating System
Send Time
Arrival Time

5.5.3 Arrival and Send Times

The *Send Time* is the time the data was sent to Georgia Tech, as recorded by the user's system. The *Arrival Time* is the time the data arrived at Georgia Tech. By observing both the send and arrival times, major timing discrepancies between Georgia Tech and the user's system can be accommodated. This will be most beneficial when analyzing connection start and end times. These times are precise to the level of detail allowed by Ethereal, which, in turn, is as precise as libpcap and the user's system allows. Typically, the level of precision is on the order of microseconds. The send time is reported in the NETI@home packet header, as shown in Figure 5.

CHAPTER 6

IMPLEMENTATION OF NETI@HOME SOFTWARE

Initially it was thought that NETI@home should function as a Linux kernel module, with similar device level software written for other systems. However, this route had many disadvantages including non-portability, bias introduced by the individual operating systems, and the difficulty of writing such software. The network sniffer approach was chosen because it clearly addresses these issues, with minimal effort. Unfortunately, some internal measurements are lost using the sniffer approach, such as the TCP congestion window sizes. The Ethereal network analyzer [4] was chosen due to its popularity, power, availability on many platforms, and open-source model.

In the hopes of assuring quality and bug-free software, an informal code review was performed before the initial release of the NETI@home software. This code review consisted of approximately ten participants, all from the Georgia Tech ECE department. Each participant was given a complete copy of the NETI@home source code, a description of each file in the code much like that in Appendix A, and a comment sheet. In addition to the code review, the NETI@home source code is available from the NETI@home website, `http://neti.gatech.edu/`, in the hope that others will review the code and make suggestions.

6.1 NETI@home Client

The NETI@home client forms the core of the NETI@home software package. Packets sniffed by Ethereal are piped to the NETI@home client where they are sorted into bidirectional flows based on their protocol, source and destination IP addresses, and source and destination port numbers (for the TCP and UDP protocols). Once these packets are sorted into their respective flows, the corresponding measurements are continuously calculated. After approximately 300 flows are fully analyzed, the data is compressed and transmitted

to Georgia Tech (`neti.ece.gatech.edu`) using TCP.

The NETI@home client is written in the C++ programming language due to C++'s portability and performance. The C++ Standard Template Library is also used for similar reasons and for convenience. Data compression is performed using the zlib compression library [15]. Portable data types are also defined so that all data collected will be similar regardless of operating system.

In all versions of the NETI@home client, the file `neti.conf` is used to specify the maximum log file size and privacy level. The NETI@home client code reads this file and then uses the settings appropriately. To implement class-based privacy levels, the class A, B, or C netmask is combined with the IP address using a bitwise AND, dependent on the class of the IP address.

The NETI@home client also includes the NETILogParse application. The NETILogParse application is written in the C++ programming language and is used to parse the log file on the client's machine. The results of parsing the log file are printed to the standard output by NETILogParse.

6.1.1 Windows

The implementation of NETI@home for the Microsoft Windows operating systems was substantially more difficult than for other operating systems. Instead of using the standard Berkeley API, all socket programming in the NETI@home client had to be done using the Winsock library. Further, the `in_addr` struct, used to specify IP addresses, is somewhat different on Windows systems and the `inet_aton` function, used to convert IP addresses from dotted decimal notation to binary form, is nonexistent. These problems are but a handful of those encountered while implementing the NETI@home client for Windows.

To start NETI@home, as well as the NETILogParse and NETIMap applications, batch files were written. Thus, when a user selects an application via the Windows Start menu, the corresponding batch file is executed, which in turn executes the appropriate program. The NETIMap batch file also checks for the presence of either Sun's Java or Microsoft's `jview`, which are used to run the NETIMap application, as it is written in Java. The batch

file used to start the core NETI@home software passes the appropriate options to tethereal and then pipes the output of tethereal to the NETI@home executable.

To cause the collection of measurements to begin as soon as possible, regardless of whether a user is logged in or not, the NETI@home software runs as a Windows service. Running as a service in the Windows 95, Windows 98, and Windows Me operating systems is trivial, all that must be done is to have the RunServices registry key point to the appropriate batch file. However, in the Windows NT, Windows 2000, and Windows XP operating systems, having NETI@home run as a service requires specialized service software. This difficulty was compounded with the fact that NETI@home cannot be called directly, but must be called with a batch file as tethereal must first be executed and then piped to NETI@home. Thus, a separate program, NETIService, had to be written to handle this task. A screenshot of the NETIService program running is shown in Figure 6.

To aid the manipulation of the `neti.conf` file, as well as to remind users that the NETI@home program is running and to allow them to halt NETI@home, the NETITray application was written. The NETITray application proved somewhat difficult to implement as it is designed to only run in the tray area of the Windows taskbar. When a user selects the properties window, the current settings are read from the `neti.conf` file. Should a user change the settings and select 'OK,' the new settings are written to the `neti.conf` file and the NETI@home application is notified of the change in settings, to allow the new settings to be used immediately. Interprocess communication between the NETITray application and the NETI@home core software is accomplished using Windows events. Two such events are implemented, one for notifying the core NETI@home software of changes in the `neti.conf` file and one for instructing the NETI@home core software to close, when a user selects this option from the NETITray menu.

Finally, to allow users to install the NETI@home software package for the Windows operating systems, a self-extracting executable was written using the InstallShield Express suite [8]. Two such executables were written, one for the Windows 95, 98, and Me operating systems and one for the Windows NT, 2000, and XP operating systems. Two different executables were required due to the different ways these operating systems implement

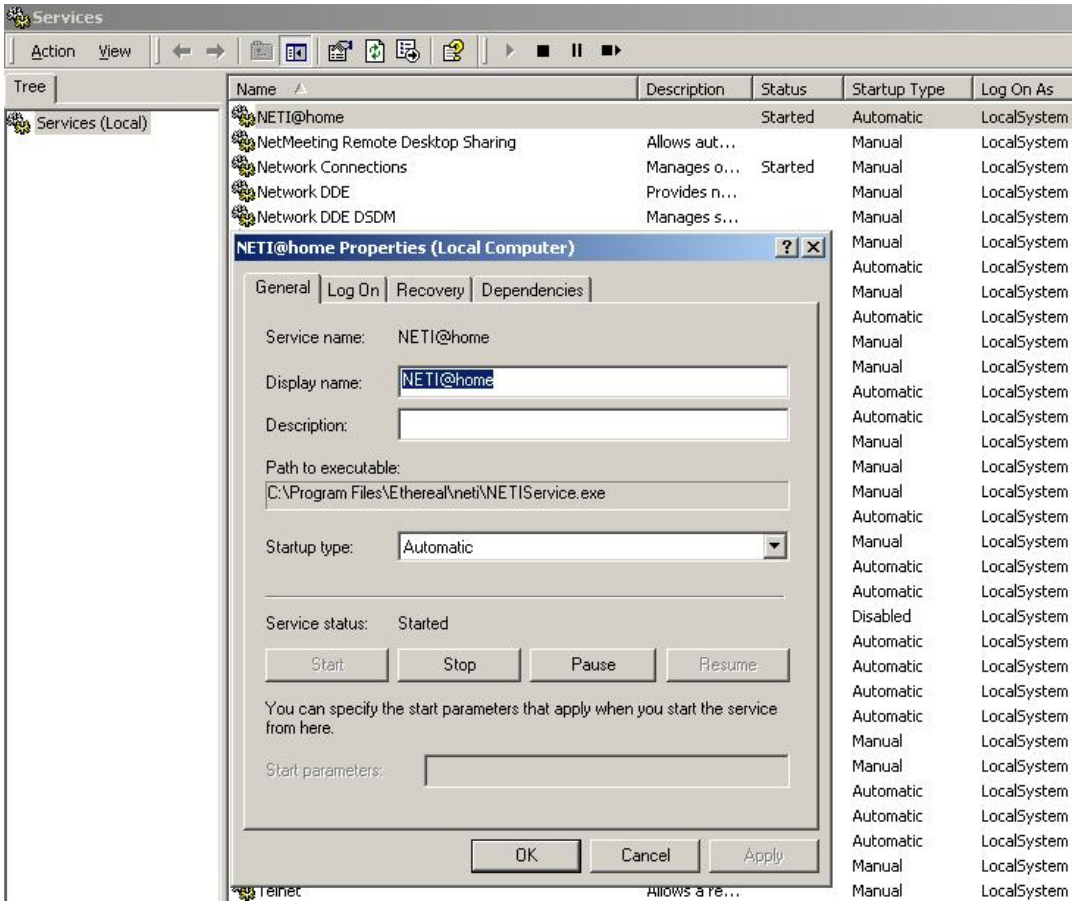

Figure 6: Screenshot of the NETIService application displayed in the Windows Services Control Panel applet (background). The properties of the NETIService application are also displayed (foreground). NETIService allows NETI@home to start at boot time and run whether or not a user is logged in on systems running Microsoft Windows NT, 2000, or XP.

services. The InstallShield Express suite allows the Windows registry to be edited, the NETI@home software to be installed in the proper location, the Add/Remove Programs entries to be added, and a Start menu folder for NETI@home to be created, among other things. In the future, the Nullsoft Scriptable Install System (NSIS) [23] will be used as it is an open-source alternative to InstallShield. Self-extracting executables created using NSIS are configured via a powerful scripting language.

6.1.2 Linux, Unix, and Others

Implementation of NETI@home for Linux, Unix, and other operating systems was significantly easier than for the Windows operating systems. NETI@home was written using POSIX functions and was tested on various systems including Redhat Linux 7.3, Redhat Linux 8.0, Redhat Linux 9.0, Gentoo Linux, and Sun's SunOS 5.8, to name a few.

To start the core NETI@home software, a shell script, NETIStart, was written which first calls tethereal with the appropriate options and then pipes the output of tethereal to the NETI@home software. This script resides in the `/sbin` directory, as only users with root access can start NETI@home to protect user privacy and due to the fact that NETI@home uses the tethereal sniffer, which could constitute a security issue.

To cause the collection of measurements to begin as soon as possible, regardless of whether a user is logged in or not, a script was written to allow NETI@home to run as a daemon. This script, usually installed in the `/etc/init.d` directory structure, allows NETI@home to run as a daemon and accept the standard start, stop, status, and restart commands.

For installation and configuration, the GNU autoconf and automake tools [7] were used. Two files had to be written to tell autoconf and automake exactly how to configure the installation scripts, `configure.in` and `Makefile.am`. The scripts generated also determine the operating system, CPU type, and vendor of the machine for reporting back to Georgia Tech.

For Redhat Linux systems and compatibles, NETI@home is also available as an RPM package. To specify how the RPM should be created a specification file, `neti-1.0-1.spec`,

was written. The resulting RPM file allows users to install NETI@home, query the package, and uninstall the package all with simple one line commands.

6.2 NETI@home Server

To collect the incoming client data at Georgia Tech, a simple TCP server program was written in the C programming language. This server accepts client connections and then decompresses the received data and writes that data to a file. The C programming language was chosen due to its performance, especially in the presence of high amounts of traffic. To validate that the data is in fact NETI@home data, a magicnumber, 4021980, is included in the NETI@home packet header, which is checked by the server. The structure of the NETI@home packet header is shown in Figure 5. The server accepts connections on TCP port 557. NETI@home clients contact the server via the DNS name `neti.ece.gatech.edu`, which currently points to `enzo.ece.gatech.edu`. Should another system be needed, the `neti.ece.gatech.edu` DNS name can easily be pointed to another server, providing scalability. The NETI@home server software is designed to be robust and fault-tolerant as it is a crucial element of the data collection process.

Another parsing program was written in the C++ programming language to allow the data collected by the NETI@home server to be parsed and displayed. This program is nearly identical to the NETILogParse program described previously, with the exception that the data parsed is not collected from one user, but from all NETI@home users.

To distribute the information collected by the NETI@home project, a website is being created using the PHP scripting language and MySQL database package. Researchers will then be able to visit the associated website and manipulate the NETI@home data using a user-friendly interface. Researchers will also be able to download the raw data if they choose.

6.3 NETIMap

The NETIMap application is written in the Java programming language. The Java programming language was chosen due to its portability, especially for GUI applications. Further,

as many Windows systems do not have Sun's Java Virtual Machine installed and merely have Microsoft's built-in `jview`, NETIMap is written to be compatible with Java version 1.1. Although Java version 1.1 is now fairly outdated, it is the latest version supported by all versions of `jview`.

To convert IP addresses into latitude/longitude coordinates, the NetGeo database [17] from CAIDA is used. CAIDA provides a NetGeo client API, which is written in Java, yet another reason NETIMap is written in Java. NETIMap must be executed while the core NETI@home software is running as IP addresses are collected by the core software. Once a remote IP address has been collected by the core software, it is sent to the NETIMap application via UDP loopback sockets. NETIMap listens for the remote IP addresses on UDP port 1557. Loopback sockets were chosen for interprocess communication because they are available on all operating systems and work with both C++ (NETI@home core software) and Java (NETIMap).

CHAPTER 7

DISTRIBUTION OF NETI@HOME SOFTWARE

One major goal of the NETI@home project is to have the NETI@home software installed in many academic computer labs across the world as well as on many end-user systems in homes and offices. As the number of users increases, so does the value of the collected data. NETI@home is currently available for the Linux and Solaris operating systems, in RPM and tarball format, and for the Windows operating system in the form of a self-extracting executable. All distributions are available from the NETI@home website located at `http://neti.gatech.edu/`. In addition to these distributions, testing of distributions for other platforms such as the Macintosh operating system are currently in progress. To use NETI@home, one must also install the Ethereal network analyzer [4], which is available from the Ethereal website, `http://www.ethereal.com/`.

Since NETI@home is copyrighted (copylefted) under the GNU General Public License (GPL) (see Appendix B for the full license), the source code is also available from the NETI@home public website, `http://neti.gatech.edu/`. Being an open-source project, NETI@home users do not have to worry about the possibility of so-called "spyware," as they are free to see exactly how NETI@home works, and make suggestions if they wish.

7.1 SourceForge

A project on the SourceForge website (`http://www.sourceforge.net`) has been created to distribute the NETI@home software. SourceForge is an excellent resource for open-source software development. SourceForge hosts open-source software projects at no charge and provides many services such as web space, CVS servers, a Compile Farm to test the compilation of software on several different platforms, and plenty of advertisement. SourceForge's popularity, and the popularity of many of the projects that it hosts, help to advertise the other, lesser known projects on its site. According to [1], SourceForge is ranked number 258

out of the top 100,000 websites in the world in terms of traffic. Thus, by using SourceForge, NETI@home can attract a larger audience than would be possible otherwise.

Currently, the NETI@home project at SourceForge consists of the current NETI@home file releases, a CVS repository, and a redirection webpage. SourceForge provides many powerful mirrors for the possibility of large numbers of simultaneous downloads, with each mirror in a strategic geographic location. The CVS repository allows developers to have a centralized, archived location for code development. The NETI@home project web space, `http://neti.sourceforge.net`, consists of a redirection page that points to the Georgia Tech NETI@home website, `http://neti.gatech.edu/`. SourceForge also provides the NETI@home project with forums for the reporting of software bugs and other software related communication. Finally, SourceForge maintains various statistics on the NETI@home project such as the number of downloads, the amount of development activity, and the number of webpage views.

7.2 `neti.gatech.edu`

NETI@home's presence on the World Wide Web is located at `http://neti.gatech.edu/`. This website informs visitors of NETI@home's purpose, the statistics that are collected by NETI@home, NETI@home's privacy levels, installation and uninstallation instructions, as well as many other topics of interest to visitors. From this website, users are able to download the NETI@home software while at the same time verifying NETI@home's legitimacy as a Georgia Tech project. The webserver which hosts this website is maintained by Georgia Tech's Office of Information Technology (OIT) and should be able to handle large volumes of Internet traffic.

To further verify the legitimacy of the downloaded files, the NETI@home website contains `md5sum` hashes of each NETI@home file. Thus, if a user suspects that the file they downloaded is not the official release of NETI@home they are able to compare the `md5sum` hash of their file to the official hash on the website.

7.3 Publicity

As previously stated, one of the major goals of the NETI@home project is to have a large installed user base, to increase the amount and variety of measurements collected. To this end, the promotion of NETI@home is an ongoing undertaking. The first major accomplishment of this task was the publication of an article describing NETI@home in the 2004 Passive and Active Measurement Workshop [21]. Continuing efforts are being made toward NETI@home related publications in other research conferences and journals. However, to attract a more mainstream audience, ongoing efforts are being made by the Georgia Tech Office of Institute Communications and Public Affairs to have NETI@home mentioned in several mass-media publications.

CHAPTER 8

BRIEF DISCUSSION OF COLLECTED DATA

This chapter contains a brief discussion of results obtained from NETI@home clients as well as some statistics on NETI@home's use. Further analysis of NETI@home results will be performed in the future. These results are from data collected from January 12, 2004 to March 31, 2004.

As of March 31, 2004, there are approximately ten active NETI@home users. These users have reported measurements collected from 131520 TCP flows, 192697 UDP flows, 3557 ICMP flows, and 2663 IGMP flows. The average size of a report from a single user was 5404 bytes, compressed, and 37294 bytes once decompressed. The minimum size of a report from a single user was 2234 bytes (19264 once decompressed) and the maximum was 45764 bytes (646964 once decompressed). These users contacted the NETI@home server an average of 13 times per day, with a maximum contact frequency of 57 per day and a minimum of once in a single day. Finally, the total size of the aggregated, decompressed data collected is approximately 40 megabytes.

CHAPTER 9

CONCLUSION

9.1 Summary of Results

NETI@home is an open-source software package designed to run on end-user systems to collect various network performance measurements. All measurements collected by NETI@home are collected passively, require little or no intervention on the part of the user, and use relatively few resources. User privacy has been protected via a configurable privacy level and can be verified by examining the freely available source code or examining the log file with the included NETILogParse application. These measurements are collected for some of the most common transport-layer protocols on the Internet: TCP, UDP, ICMP, and IGMP. The NETIMap application is also included with the NETI@home software package to further encourage participation in the NETI@home project by users. NETI@home also runs on many different operating systems to provide maximum availability. Finally, data gathered by the NETI@home client software is reported to Georgia Tech in a scalable manner, where it will be made publicly available to aid researchers in the on-going quest to better the global Internet.

9.2 Future Work

In the future, many more features will be added to NETI@home to aid researchers. For example, it would be extremely useful to collect the size of the TCP congestion window, which would allow researchers to monitor the TCP congestion control algorithm. However, to collect the size of the congestion window, as well as many other statistics, will require a lower-level application, such as a kernel module, or some other novel technique. Some other new features under consideration are: collecting the bandwidth available to the end user, collecting the round trip time according to the TCP three-way handshake [12], adding additional transport-layer protocols to the current four that are studied, and collecting

traceroutes from the user to selected destinations. Depending on the user's privacy level, the traceroute will either not be recorded, will be trimmed, or it will be fully recorded. The traceroute was not initially included as part of NETI@home because it might tax users' systems as well as the network itself. However, using the NETIMap program, traceroute functionality will be provided. When a user clicks a specific host on the map, another window will appear and a graphical traceroute will be performed. Since the traceroute is explicitly performed by the user, it should not be considered taxing to the user's system or the network, and it truly represents a passive technique. This data will also be sent to the server at Georgia Tech to add to the amount of measurements that are collected.

As a further effort to maximize research potential while at the same time respecting the privacy of the users, NETI@home privacy settings will constantly be analyzed and improved upon. Currently, research efforts are underway to create a system to allow each NETI@home user to anonymize their collected IP addresses by hashing these addresses into unique identifiers. Such hashing would maintain the uniqueness of the IP addresses while at the same time providing anonymity. It is also desired that such a hashing procedure maintain the IP prefix, or at least that IP addresses with similar prefixes have similar hashed prefixes, to allow them to be grouped accordingly.

As previously mentioned, future releases of the NETI@home software package for Windows operating systems will be distributed as self-extracting executables created with the Nullsoft Scriptable Install System [23]. NSIS is a much more powerful, free, open-source alternative to the currently used InstallShield Express suite [8].

Research is also currently underway to integrate a sniffer into the NETI@home source code and thereby eliminate the need for a third-party sniffer such as Ethereal. This research is being performed by Juan Carlos Gonzalez, as part of the Intel Opportunity Scholars Program, under the direction of myself and Dr. Riley. Full integration of the sniffer into the NETI@home source code will have many advantages including the integration of the NETITray and NETIService applications into the core source code as well as the elimination of overhead associated with piping tethereal's output into the NETI@home software.

Also as part of the Intel Opportunity Scholars Program and under direction of myself

and Dr. Riley, Crystal Wrenn is making various improvements to the NETIMap application. These improvements include advanced graphical techniques such as zooming and implementation of traceroute functionality.

Finally, the NETI@home online database will receive many enhancements and analysis will be performed on the collected data.

APPENDIX A

FILE LISTING

The following sections contain lists of the files that constitute the NETI@home software package. Each of these files, and thus the NETI@home source code, are available from the NETI@home website, located at `http://neti.gatech.edu/`, with the exception of the server software.

A.1 Core

The following files constitute the "core" NETI@home files, that is, the files that are directly responsible for the collection of the network measurements, and the reporting of those measurements to Georgia Tech.

- `convert_tad.c`: This file was taken from glibc and is largely unmodified. It provides the `strptime` function, used to parse times, since it is not included on many systems.
- `convert_tad.h`: This file was taken from glibc and is largely unmodified. It provides the `strptime` function, used to parse times, since it is not included on many systems.
- `flows.h`: This file describes the flow structures and classes used by NETI@home.
- `logparse.cpp`: This program parses the log file created on the local machine and constitutes the majority of the NETILogParse application. It is very similar to parse.cpp.
- `neti.conf`: This is the configuration file for NETI@home. It sets the maximum log file size and the privacy setting. In Windows, this file is modified by the NETITray application, on other systems it resides in the `/etc` directory. It is included in the source code distribution to set the defaults: a medium privacy level and a maximum log file size of one megabyte.

- `neti.cpp`: This is the main file for collecting the network measurements and reporting them back to the NETI@home server. Packets sniffed by ethereal are passed to this program for processing.

- `neti.h`: This is the main header file. It contains the statistics that are collected as well as various other important information.

- `neti_netof.c`: This is a helper file that contains simple networking functions.

- `neti_netof.h`: This is the header file for the above helper file that contains simple networking functions.

- `parse.cpp`: This is a server-side program for parsing the file that is created by the NETI@home server. It is very similar to logparse.cpp.

- `server.c`: This is the server program for NETI@home that runs on the machine pointed to by the DNS entry `neti.ece.gatech.edu`. It collects the data, decompresses it, and places it into a log file.

A.2 NETIMap

The following files are used to create the NETIMap application.

- `NetGeoClient.java`: This file is unmodified by myself. It was obtained from CAIDA to access their NetGeo database, which converts IP addresses into latitude/longitude coordinates [17].

- `NETIMap.java`: This program, when run, will map IP addresses onto a world map. The IP addresses are sent to NETIMap from neti.cpp.

- `world100.gif`: This file contains an image of an Alber/Lambert projection world map in GIF format. This image is used as the map for the NETIMap application and was obtained from CAIDA, courtesy of VisualRoute (`http://www.visualroute.com`).

A.3 *NIX

The following files are specific to Linux, Unix, and systems other than those running Microsoft Windows operating systems. These files are responsible for starting NETI@home at boot time, configuration, and installation.

- `neti.script`: This script allows NETI@home to run as a daemon so that it can run regardless of whether or not a user is logged in and at boot time.
- `configure.in`: This file is used by GNU autoconf [7] to create a configure script, used to determine if a system has all the appropriate functions for NETI@home to operate and the locations of these functions. The configure script also determines the operating system of the machine on which it is run.
- `Makefile.am`: This file is used by GNU automake to create makefiles, which are used to compile NETI@home.
- `neti-1.0-1.spec`: This is the specification file used to create RPM packages.
- `neti.7`: This is the man page for NETI@home.
- `NETIStart`: This file starts NETI@home on systems other than Windows. It first starts tethereal with the appropriate options and then passes tethereal's output to NETI@home.

A.4 Windows

The following files are specific to systems running Microsoft Windows operating systems. These files are responsible for starting NETI@home at boot time, starting the various NETI@home applications, and other Windows-specific tasks.

- `go.bat`: This file starts NETI@home on Windows systems. It first starts tethereal with the appropriate options and then passes tethereal's output to NETI@home.
- `gpl.rtf`: This file is the GNU GPL (see Appendix B for the full license) in rich text format. It is displayed during installation on systems running Microsoft Windows

operating systems and users must explicitly agree to its terms and conditions before installation will complete.

- `logparse.bat`: This file runs the NETILogParse application on Windows systems.
- `map.bat`: This file runs the NETIMap program on Windows systems. It also checks for the existence of various Java programs, such as Microsoft's `jview` and `wjview` and Sun's `java` and `javaw`.
- `neti.ico`: This file contains the NETI@home graphical icon, which is used to indicate NETITray's presence in the tray area of the Windows taskbar as well as to represent the NETI@home applications in the NETI@home group of the Windows Start menu.
- `NETIService.cpp`: This is a program that allows NETI@home to run as a service in Windows NT, 2000, and XP and constitutes the NETIService application. This file is a modified version of the XYNTService originally written by Xiangyang Liu [14].
- `NETIService.ini`: This is the configuration file used by NETIService.
- `NETITray.cpp`: This is the main file for the Windows system tray program, NETITray. It allows users to select their privacy settings and maximum log file size via simple menus. Also, NETI@home can be disabled via one of the menu options.
- `NETITray.h`: This is the header file for the NETITray program.
- `NETITray.rc`: This is another Microsoft generated file for the menus, etc. for NETITray.
- `resource.h`: This is the typical Windows resource header file, used by the NETITray application.

APPENDIX B

THE GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted,

and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections

of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You

are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries

not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

REFERENCES

[1] "Amazon.com: website info: SourceForge." Available on-line: http://partner.alexa.com/amzn/redirect_to_detail?url=sourceforge.net, April 2004.

[2] Barford, P. and Sommers, J., "A comparison of active and passive methods for measuring packet loss," October 2002. University of Wisconsin Technical Report.

[3] "CAIDA: Preliminary measurement specification for Internet routers." http://www.caida.org/tools/measurement/measurementspec/, 2004. The Cooperative Association for Internet Data Analysis - CAIDA.

[4] Combs, G. and et al., "Ethereal: - a network protocol analyzer." Software on-line: http://www.ethereal.com, 2004.

[5] Corral, J., Texier, G., and Toutain, L., "End-to-end active measurement architecture in IP networks (SATURNE)," in *PAM2003 - A workshop on Passive and Active Measurements*, April 2003.

[6] Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., and Tobagi, F., "Design and deployment of a passive monitoring infrastructure," *Lecture Notes in Computer Science*, vol. 2170, 2001.

[7] "GNU autoconf." Software on-line: http://www.gnu.org/software/autoconf/, 1998.

[8] InstallShield Software Corporation, "InstallShield express," 2002. http://www.installshield.com/.

[9] Jacobson, V., "traceroute." Software on-line: ftp://ftp.ee.lbl.gov, 1989. Lawrence Berkeley Laboratory.

[10] Jacobson, V., Leres, C., and McCane, S., "libpcap." Software on-line: http://www.tcpdump.org, 1989. Lawrence Berkeley Laboratory.

[11] Jacobson, V., Leres, C., and McCanne, S., "tcpdump." Software on-line: http://www.tcpdump.org, June 1989. Lawrence Berkeley Laboratory.

[12] Jiang, H. and Dovrolis, C., "Passive estimation of TCP round-trip times," *ACM Computer Communications Review*, vol. 32, July 2002.

[13] Keys, K., Moore, D., Koga, R., Lagache, E., Tesch, M., and k claffy, "The architecture of CoralReef: An Internet traffic monitoring software suite," in *PAM2001 - A workshop on Passive and Active Measurements*, CAIDA, April 2001. http://www.caida.org/tools/measurement/coralreef/.

[14] Liu, X., "Start your Windows programs from an NT service." Software on-line: http://www.codeproject.com/system/xyntservice.asp, September 2000. The Code Project.

[15] loup Gailly, J. and Adler, M., "zlib compression/decompression library." Software on-line: http://www.gzip.org/zlib/, 1995.

[16] Mochalski, K. and Irmscher, K., "On the use of passive network measurements for modeling the Internet," in *KiVS*, 2003.

[17] Moore, D., Periakaruppan, R., Donohoe, J., and kc claffy, "Where in the world is netgeo.caida.org?," in *INET 2000*, June 2000.

[18] Murray, M. and kc claffy, "Measuring the immeasurable: Global Internet measurement infrastructure," in *PAM2001 - A workshop on Passive and Active Measurements*, April 2001.

[19] Paxson, V., Mahdavi, J., Adams, A., and Mathis, M., "An architecture for large-scale Internet measurement," *IEEE Communications*, vol. 36, pp. 48–54, August 1998.

[20] "SETI@home: Search for extraterrestrial intelligence at home." Software on-line: http://setiathome.ssl.berkeley.edu.

[21] Simpson, C. R. and Riley, G. F., "NETI@home: A distributed approach to collecting end-to-end network performance measurements," in *PAM2004 - A workshop on Passive and Active Measurements*, April 2004.

[22] Thompson, K., Miller, G. J., and Wilder, R., "Wide-area Internet traffic patterns and characteristics (extended version)," *IEEE Network*, 1997.

[23] Verburg, J., "Nullsoft scriptable install system." Software on-line: http://nsis.sourceforge.net/, 2003.