Analysis of Passive End-to-End Network Performance Measurements

A Dissertation
Presented to
The Academic Faculty

by

Charles Robert Simpson, Jr.

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

Georgia Institute of Technology
May 2007

Analysis of Passive End-to-End Network Performance Measurements

Approved by:

Dr. George F. Riley, Advisor
Asst. Professor, School of ECE
Georgia Institute of Technology

Dr. Biing-Hwang (Fred) Juang
Professor, School of ECE
Georgia Institute of Technology

Dr. Henry L. Owen
Professor, School of ECE
Georgia Institute of Technology

Dr. Richard M. Fujimoto
Professor, College of Computing
Georgia Institute of Technology

Dr. John A. Copeland
Professor, School of ECE
Georgia Institute of Technology

Date Approved: 14 December 2006

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

First and foremost, I would like to thank the NETI@home users, without whose participation there would be no NETI@home project and research at all.

I would like to thank my advisor, Dr. George Riley, for his help and guidance in the completion of my studies. 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 dissertation, 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 former roommate Julian Grizzard has also been very helpful, especially with those late-night ponderings.

Elizabeth Campell deserves many thanks, as her work in the Georgia Institute of Technology Office of Institute Communications and Public Affairs helped to publicize NETI@home and increase the number of users.

I would also like to thank everyone else who has helped with the successful completion of this dissertation.

Finally, I would like to acknowledge that all of my blessings are the work of God.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

API	application programming interface.
ARP	address resolution protocol.
ASCII	American standard code for information interchange.
CAIDA	cooperative association for internet data analysis.
CDF	cumulative distribution function.
CPU	central processing unit.
CVS	concurrent versioning system.
DHCP	dynamic host configuration protocol.
DIMES	distributed internet measurements and simulations.
DNS	domain name system.
DoS	denial of service.
EAPOL	extensible authentication protocol over lan.
ECN	explicit congestion notification.
FTP	file transfer protocol.
GB	gigabyte.
GNU	GNU's not UNIX.
GPL	general public license.
GTNetS	georgia tech network simulator.
GUI	graphical user interface.
HTTP	hypertext transfer protocol.
HTTPS	hypertext transfer protocol secure.
ICMP	internet control message protocol.
IETF	internet engineering task force.
IGMP	internet group management protocol.
IP	internet protocol.
IPv6	internet protocol version 6.
IPX	internetwork packet exchange.

ISP	internet service provider.
KB	kilobyte.
LDAP	lightweight directory access protocol.
MB	megabyte.
MD5	message-digest algorithm 5.
MSS	maximum segment size.
MTU	maximum transmission unit.
NAT	network address translation.
NETI@home	network intelligence at home.
NIMI	national internet measurement infrastructure.
NSIS	nullsoft scriptable install system.
POP	post office protocol.
RFC	request for comments.
RPM	RPM package manager.
RTT	round trip time.
SACK	selective acknowledgement.
SDK	software development kit.
SETI@home	search for extraterrestrial intelligence at home.
SSH	secure shell.
TB	terabyte.
TCP	transmission control protocol.
TTL	time to live.
UDP	user datagram protocol.
VoIP	voice over internet protocol.
WWW	world wide web.

SUMMARY

As the use of networks is increasingly becoming an important part of daily life, the measurement and analysis of these networks is becoming important as well. This dissertation first introduces a network measurement infrastructure designed to collect these measurements from end-hosts on the Internet. Then, utilizing these measurements, studies are made on the behavior of the network and network users as well as the security issues affecting the Internet. Finally, the public release of the collected data is discussed.

The NETI@home network measurement infrastructure is a distributed approach to passively gathering end-to-end network performance measurements. The client is designed to run on virtually any machine connected to the Internet and measurements are reported to a server located at the Georgia Institute of Technology. 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.

We conduct a flow-based comparison of honeynet traffic, representing malicious traffic, and NETI@home traffic, representing typical end-user traffic. We present a cumulative distribution function of the number of packets for a TCP flow and learn that a large portion of these flows in both datasets are failed and potentially malicious connection attempts. Next, we look at a histogram of TCP port activity over large time scales to gain insight into port scanning and worm activity. One key observation is that new worms can linger on for more than a year after the initial release date. We go on to look at activity relative to the IP address space and observe that the sources of malicious traffic are spread across the allocated range. Finally, we discuss other security-related observations including suspicious

use of ICMP packets and attacks on our own NETI@home server.

We present some observations and conclusions based on the behavior of the network and networking protocols, from the unique perspective of the end-user. An analysis of hop counts, based on observed TTL values, is presented. The frequency and use of network address translation (NAT) and the private IP address space are studied. Finally, several other options and flags of various protocols are analyzed to determine their adoption and use by the Internet community.

The simulation of computer networks requires accurate models of user behavior. To this end, we present empirical models of end-user network traffic derived from the analysis of NETI@home data. There are two forms of models presented. The first models traffic for a specific TCP or UDP port. The second models all TCP or UDP traffic for an end-user. These models are meant to be network-independent and contain aspects such as bytes sent, bytes received, and user think time. The empirical models derived in this study can then be used to enable more realistic simulations of computer networks and are implemented in GTNetS.

Finally, we further discuss our approaches to anonymizing the dataset and how these anonymized data and their associated analysis tools will be distributed.

CHAPTER 1

INTRODUCTION

The Internet has become an increasingly essential part of modern life. 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 an important 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 improved models for traffic flows, file sizes, burst sizes, and many other complex characteristics of the Internet. Such measurements have implications for the performance of networking protocols and operating system protocol stacks, the study of malicious network traffic, and the modeling and simulation of networks. Most of the sampling techniques for this data have come either from active measurements (`ping` [53]) or from localized passive measurements (`tcpdump` [34]). It has been documented that active measurements introduce bias into these measurements, and many claim that this bias is sufficiently large to cause some collected measurements to not be representative of actual Internet traffic [5, 48]. As for the passive measurements that have been conducted, they are only able to analyze a small portion of the Internet and cannot give a good representation of the end-user experience as they are only collected from a limited number of vantage points. 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 the study of large-scale, end-to-end, passive network measurements.

The objective of this research is to analyze passive end-to-end network performance measurements to obtain a better understanding of network activity. Specifically, there are

three key areas that are to be investigated: network security, network behavior, and user behavior. Each of these three characteristics is vital to understanding the nature of the Internet.

To complete this investigation, the NETI@home dataset is heavily analyzed, as are other datasets when necessary. The NETI@home infrastructure's unique end-user perspective aids in all three of these areas and provides an insight that would be otherwise much more difficult, if not impossible.

1.1 NETI@home

We introduced the NETI@home (NETwork Intelligence at home) software package, a distributed network monitoring infrastructure whose aim is to passively capture measurements from end-hosts on the Internet, to address the need for large-scale, end-to-end network measurements [68, 72]. Capturing measurements from end-hosts gives us a unique perspective on the behavior of both the network and network users. NETI@home is designed to run on end-user computers with a variety of operating systems and to require little or no intervention by the user. It collects an assortment of network measurements from these machines and then sends the results to the Georgia Institute of Technology for analysis.

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 his or her privacy in any way. Should privacy be inadvertently violated, at the very least it would spell the end for such a large-scale collection system.

For more information on the NETI@home project, please see my Master's Thesis [69] as well as several NETI@home related publications including [27, 71, 72].

With thousands of users strewn across the globe, we believe we can leverage the volume and quality of the NETI@home data to provide a better understanding of the state of the global Internet.

1.2 Network Security

As the Internet has grown from its infancy, it has reached a stage where security concerns have become a considerable problem. Today's Internet is plagued by a plethora of worms, viruses, malware, spam, and other malicious traffic. Utilizing network measurements enables researchers to study the growth and spread of this malicious activity, as well as investigate the origins of these problems.

1.3 Network Behavior

The end-hosts that make up the Internet run a variety of operating systems, each with its own somewhat unique network protocol stack. The differences in these protocol stacks can have a significant impact on the interoperability of Internet end-hosts. Further, many protocol stacks lag far behind the research community and the IETF when implementing improvements to networking protocols. Finally, as with any complex piece of software, operating system network protocol stacks are plagued by bugs. By utilizing the observations and analyses of network measurements, these issues can be identified and potentially corrected.

1.4 User Behavior

The simulation of computer networks has become a popular method to evaluate characteristics of these networks across a wide range of topics, including protocol analysis, routing stability, and topological dependencies, to name a few. However, for these simulations to yield meaningful results, they must incorporate accurate models of their simulated components.

One such component is end-user traffic generation. This component should be network-independent so that it can be used in a wide variety of simulation configurations without dependency on the simulated environment. These traffic models should be updated frequently, using recent measurements, to accurately reflect the changing nature and uses of the Internet. Further, such measurements should represent the heterogeneous connection methods and diverse locations of Internet users.

1.5 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 first presents work related to this dissertation as well as other relevant background information.

Next, Chapter 3 describes the NETI@home network measurement infrastructure, the key to the collection of needed measurements for this dissertation. The measurements collected by NETI@home are discussed as are the privacy settings, implementation, and distribution of the software.

Chapter 4 presents observations and analyses related to network security. This is primarily accomplished by comparing and contrasting data from the NETI@home project to those collected by the Georgia Tech Honeynet project.

Then, Chapter 5 presents observations and analyses related to the behavior of the network. This chapter includes aspects related to the performance and routing stability of network connections, from the perspective of end-users running the NETI@home software. The utilization of NAT and various protocol flags and options are also discussed.

Chapter 6 presents a study of the behavior of end-users and their usage of the network. Network independent empirical models are derived based on this behavior and then used in simulation.

Chapter 7 discusses the release of the data collected by NETI@home. Such data will prove useful for researchers. Accompanying the data are the tools developed to study the data for this dissertation.

Finally, Chapter 8 presents our conclusions. Several areas of future work are also discussed.

Appendix A details the various structures used by all versions of NETI@home, in RFC format.

CHAPTER 2

BACKGROUND

2.1 Measurement Infrastructures

There are several projects whose aim is to collect and distribute network measurements. In this section we attempt to categorize these projects. However, because of the sheer number of such projects, a complete and updated list is not provided here.

One key distinction between network measurement approaches is whether they are active or passive. Currently, the NETI@home project collects only passive measurements, that is, measurements that do not inject network traffic and try to minimize the impact of their actions on the measured phenomenon. Active measurements [15, 58], on the other hand, directly interact with the system they are attempting to measure. These measurements typically inject network traffic, such as probe packets. Passive measurements [23, 38, 82] do not inject any traffic into the network; they merely monitor traffic on the network and infer measurements from the observed traffic. Several studies have compared active and passive measurements [5, 48], 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` [53] and `traceroute` [32]. 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 [5]. Active measurements can also introduce bias into the measured system, since they actually inject packets, and thus interact with, the measured traffic.

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` [34] and Ethereal [14], 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-hosts. Other forms of passive measurements observe the network at a point that is between the two end-hosts. Two such proposed systems are OC3MON [82] and IPMON [23]. 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 [23, 82]. However, most ISPs are reluctant to release such information since it could potentially expose problems in their own networks or violate users' privacy [52].

In addition to being active or passive, many measurement infrastructures specialize in the types of measurements that they collect. We have identified three basic types of measurement infrastructures: general purpose monitoring infrastructures, security-related monitoring infrastructures, and network measurement monitoring infrastructures.

The first class, general purpose monitoring infrastructures, collects measurements that can be used for both security-related research and network measurement-related research. The most basic type of general purpose measurement is the use of passive network traces, such as those collected using `libpcap` [33]. Many institutions, researchers, and network administrators collect traces. These traces are usually limited to the collector's own network. One of the early network monitoring infrastructures, which falls into this class of general purpose monitoring infrastructures, is Vern Paxson's National Internet Measurement Infrastructure, NIMI [58]. NIMI utilizes an active approach, taking measurements between specialized nodes placed strategically throughout the Internet. However, if this infrastructure can ever be fully put in place, it will still be unable to give an entirely accurate representation of the end-user experience. Further, it suffers from the unwillingness of ISPs to install such measurement devices on their networks. Other infrastructures of note are SATURNE [15] (active), OC3MON [82] (passive), IPMON [23] (passive), and CAIDA's CoralReef [38] (passive). This class also includes the NETI@home project.

The second class of monitoring infrastructures, those related to security, collects measurements used to identify and track malicious activity on the Internet. One basic approach is the use of honeynets [61, 78], machines placed on the Internet that have no other purpose than to record the traffic they receive, which largely consists of malicious activity. A different approach is CAIDA's Network Telescope project [49], which monitors an unused portion of the IP address space. This traffic, much like the traffic to honeynets, largely consists of malicious activity. One of the larger and more distributed security-related monitoring infrastructures is the Internet Storm Center [81]. The Internet Storm Center consists of a large number of independent monitoring systems such as honeynets, which aggregate their data, allowing a much larger picture of the malicious activity on the Internet.

Finally, the remaining class of monitoring infrastructures, those specializing in network measurements, collects measurements to further research into network design, operation, and performance. One of the more popular such projects is CAIDA's Skitter [73]. Skitter actively probes the Internet to analyze network topology and performance with measurements such as traceroute and ping. Two recent projects with competing aims have also been developed to actively map the Internet with traceroute-like activity, the DIMES Project [19] and Traceroute@home [84], and are based on end-user participation much like NETI@home.

2.2 Network Security

Much research has been accomplished on studying Internet worms and their behavior. Work has been accomplished on characterizing and looking at the trends of various worms [39, 86]. Further, a detailed study of the spread time, algorithms, and damage caused by recent worms has been conducted. For example, Shannon et al. give an in-depth look at the Witty worm in [65], and Moore et al. give an in-depth look at the Slammer worm in [50]. Both of these worms have been observed in the NETI@home dataset as well as in other datasets studied such as those from the Georgia Tech Honeynet [24]. Data shows that these worms' lingering effects are still active.

CAIDA uses its previously mentioned Network Telescope, which consists of a full /8 network to observe worms, DoS attacks, network scanning, and other malicious activity

[49]. Finally, the also previously mentioned Internet Storm Center provides users and organizations with warnings against possible new threats seen on the Internet [81].

2.3 Network Behavior

The study of end-hosts, network protocols, and operating system behavior has received much attention in the research literature. Among some of the more famous studies are those related to the performance and modeling of protocols such as TCP [47, 56].

Among protocol flags and options that have been studied, one of the more studied is the IP fragmentation option. Several authors have recommended the almost total abandonment of packet fragmentation and have studied the adoption of this idea, including [37, 66].

2.4 User Behavior

The need for accurate simulation models was discussed in [22]. Several other studies have discussed modeling of either application-specific [4, 8, 10, 11, 75] or general [3, 28, 29, 76] end-user network traffic. Also, several studies have used network traffic models in simulation environments, including [12, 41, 87, 88].

Portions of our work are based on work presented in [44] and [75] and we have chosen to adopt much of their nomenclature. However, we have attempted to expand upon their work in several ways. First, the work in [44] is based on packet traces collected from a campus network. In an attempt to represent more typical end-users, we use data collected by the NETI@home project. Also, the studies conducted in [44, 75] were specific to TCP connections on port 80. We, however, model any given TCP or UDP port, as well as all TCP or UDP traffic aggregated.

CHAPTER 3

THE NETI@HOME NETWORK MEASUREMENT INFRASTRUCTURE

3.1 Description of NETI@home Software

NETI@home is an open-source software package named after the popular SETI@home [2] software. The NETI@home client is available on the NETI@home website [68] and is designed to be run by almost any client machine connected to the Internet. When run on a client machine, the NETI@home software reports end-to-end flow summary statistics to a server at the Georgia Institute of Technology. The statistics collected and the functionality of the software are further discussed in [69, 72]. Since NETI@home is designed to run on end-user systems, it provides a unique perspective on the behavior of both end-users and their systems.

NETI@home is designed to run on almost any end-host machine connected to the Internet, to maximize the volume and variety of measurements gathered. As such, it has been written for and tested on the Windows, Solaris, Linux, and Mac OS X operating systems. Our basic approach is to sniff packets sent from and received by the monitored host and infer performance metrics based on these observed packets. NETI@home is built on top of the popular `libpcap` software library [33], the de facto standard cross-platform packet sniffing library, and is written in the C++ programming language. Further, we run the NETI@home software in *non*-promiscuous mode, which ensures that we only observe and report on traffic specifically addressed to that end-system 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. One important requirement of the software is that it be unobtrusive and run quietly in the background with little or no intervention by the user and use few resources.

Currently, all measurements made by NETI@home are passive. The NETI@home software collects end-to-end flow summary statistics on the TCP, UDP, ICMP, and IGMP protocols, as well as their underlying protocols. In addition to these protocol-related measurements, the software also records information about the host on which it is running, such as the operating system type and version as well as user-supplied geographical location information. For a more in-depth discussion of the statistics gathered see [72] and for more up-to-date information see [68].

One key aspect of the NETI@home project is our commitment to user privacy. To aid in fulfilling this commitment, NETI@home users select a privacy level that determines what types of data are gathered and what is not reported. There are currently three privacy settings. Medium, the default setting, records only the network portion of the local, remote, and multicast IP addresses, as determined by the local netmask. This allows for many interesting macroscopic studies of the Internet while not compromising the identities of the end-hosts. The high privacy setting does not record any IP addresses, and the low privacy setting records the full local, remote, and multicast IP addresses observed. In addition to the privacy settings, each time a report is sent to the NETI@home server, an identical local copy is retained so that users can view these contents and verify the operation of the software. Finally, the open-source nature of the software allows anyone to verify the functionality of the software.

Once NETI@home analyzes a specified number of flows or a specified amount of time has passed, the data is compressed using the `zlib` compression library [43]. NETI@home clients then report this data via TCP to a DNS name that resolves to a server that resides at the Georgia Institute of Technology. This server receives and collects all user reports and is carefully monitored. If needed, and to ensure scalability, we will implement round-robin DNS to allow multiple servers to simultaneously collect data.

For a project such as NETI@home to be useful there must be a large userbase. To this end, we have provided some incentive for participation as well as pursued various avenues of publicity. To encourage users to run the NETI@home client software, we included a program called NETIMap. When run in conjunction with the NETI@home client software,

NETIMap plots the location of the remote host on a global map using CAIDA's NetGeo database [51]. In an effort to gain attention, we have pursued other avenues of publicity in addition to research literature. Examples of such publicity include Wired [18] and the popular Slashdot website [13, 83]. Also, our software is available on the popular SourceForge website [77].

As of October 31, 2006, there have been approximately 4,500 active NETI@home users, sending some 873,000 reports and located in over 40 nations and 91 US ZIP codes. These users have reported measurements collected from approximately 91 million TCP flows (transferring approximately 1.23 TB of application data), 175 million UDP flows (transferring approximately 21.64 GB of application data), 5 million ICMP flows, and 3 million IGMP flows. The average size of a report from a single user was 6.72 KB, compressed, and 39.75 KB once decompressed. The maximum size of a report from a single user was 1.27 MB, compressed, and 3.43 MB once decompressed. Finally, the total size of the aggregated, decompressed data collected thus far is approximately 34 GB.

NETI@home users represent a heterogeneous mixture of network users from various networks and geographical locations using a variety of network connection methods.

The wealth of data collected during this time has led to many interesting observations and analyses, although there are still many more avenues of investigation to pursue.

The distributed approach to the collection of network measurements was inspired by the SETI@home project [2], 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 early 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.

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. The GNU GPL was selected 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. For more information on the GNU GPL see [25].

3.1.1 `libpcap`

NETI@home uses the open-source `libpcap` software library [33] to collect packets for analysis. `libpcap` is the de facto standard cross-platform packet sniffing library and has a complementary implementation in Windows, WinPcap [17]. NETI@home sniffs packets in real-time with little impact on the user's system. Further, `libpcap` is used to place the network interface(s) in non-promiscuous mode.

`libpcap` is available from their website, located at `http://www.tcpdump.org/`. WinPcap is available from their website, located at `http://www.winpcap.org/`.

3.1.2 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. The high privacy setting provides the maximum respect to the users' privacy. 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 local, remote, and multicast IP addresses, which are trimmed to contain only the network portion of their addresses. The network portion is defined by the local netmask. 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

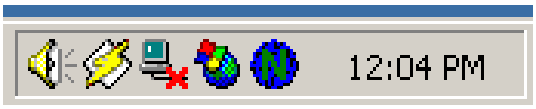


Figure 1: Screenshot of the NETITray application in the Windows taskbar. By right-clicking on this icon, a user is presented with the NETITray menu, as shown in Figure 2.

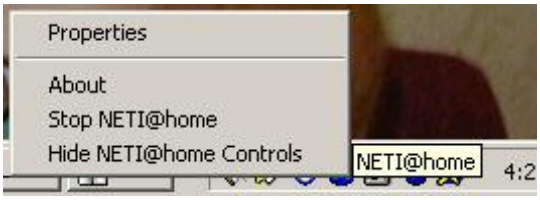


Figure 2: Screenshot of the NETITray menu. From this menu, users are able to select between the Properties window (Figure 3), the About window, an option to halt NETI@home's collection of measurements, and an option to close the NETITray application.

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.

3.1.3 NETITray

For users of the Microsoft Windows operating systems, NETI@home includes a program, titled 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 Hide NETI@home Controls, as seen in Figure 1 and Figure 2.

By selecting the Properties menu item, users are able to modify several aspects of the operation of the NETI@home software: the maximum log file size, the privacy level, the geographical location, the US ZIP, the user's email address, and the interface on which NETI@home should monitor. Users are able to select between one KB and ten MB log file

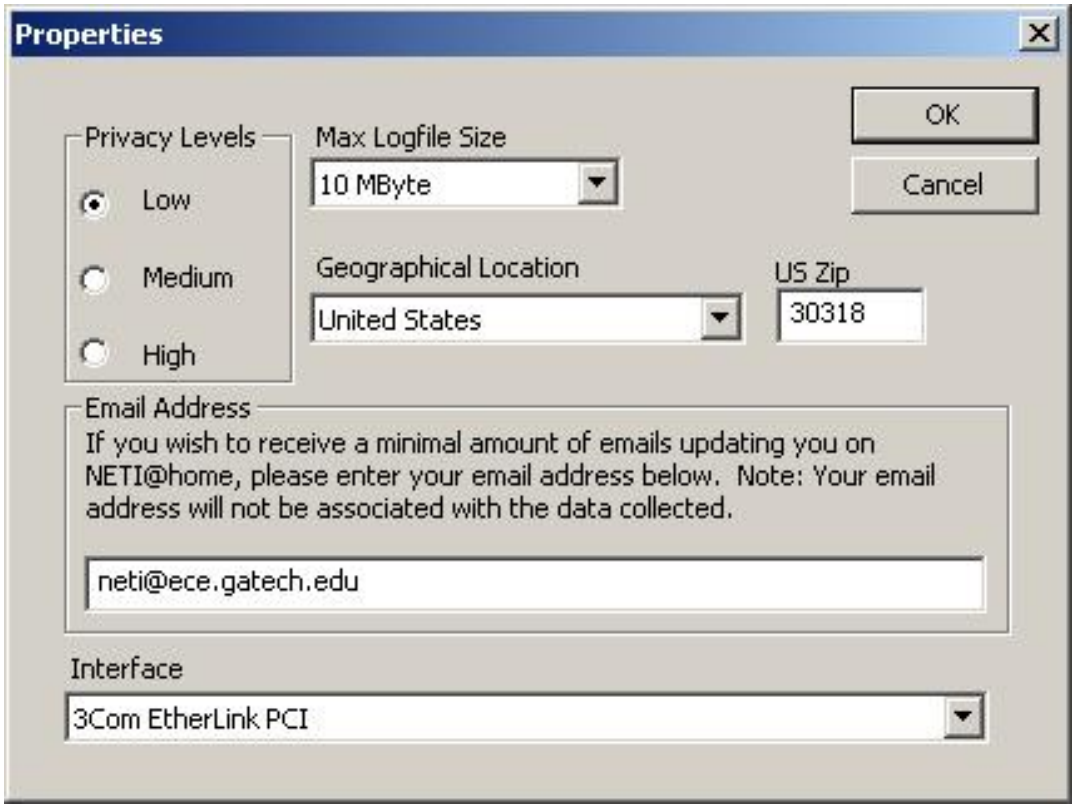


Figure 3: Screenshot of the NETITray properties window. From this window, users are able to select their desired privacy level, maximum log file size, geographical location, and desired interface to monitor. Further, users can specify a US ZIP code if they are located within the United States and can specify their email address to receive NETI@home related mailings.

sizes, which determines the maximum size of the log file stored on the user's system. The default log file size is one MB. Users are further able to select between the high, medium, and low privacy levels. The default privacy setting is the medium privacy level. Users are also able to select between approximately 260 geographical locations. Finally, the user is able to select between all available network interfaces to monitor. 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 Hide NETI@home Controls menu item closes the NETITray application without

halting the collection of measurements.

3.1.4 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 open-source and robust Nullsoft Scriptable Install System (NSIS) [85]. For RedHat Linux systems and compatibles, NETI@home is available in the form of an RPM file. For Macintosh OS X systems, NETI@home is available in the form of a Mac OS X package. 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 [26], to ease configuration and installation. Typically for these other systems, one must only execute three commands (`configure`, `make`, and `make install`).

3.1.5 Update Alerts

To ensure that users are aware of new updates, as well as to allow for prompt bug fixes, implementation of new measurements, and changes in existing measurement collection techniques, the NETI@home client occasionally queries the NETI@home server for the current version of NETI@home. If the server replies with a current version that is more recent than the version the user is running, an update alert is generated. If the user is running NETI@home in a Microsoft Windows operating system, an alert icon is shown by the NETI-Tray application. For all other operating systems, a text alert is generated in the standard system log file.

3.1.6 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 the Georgia Institute of Technology. As binary data is sent to the Georgia Institute of Technology, 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

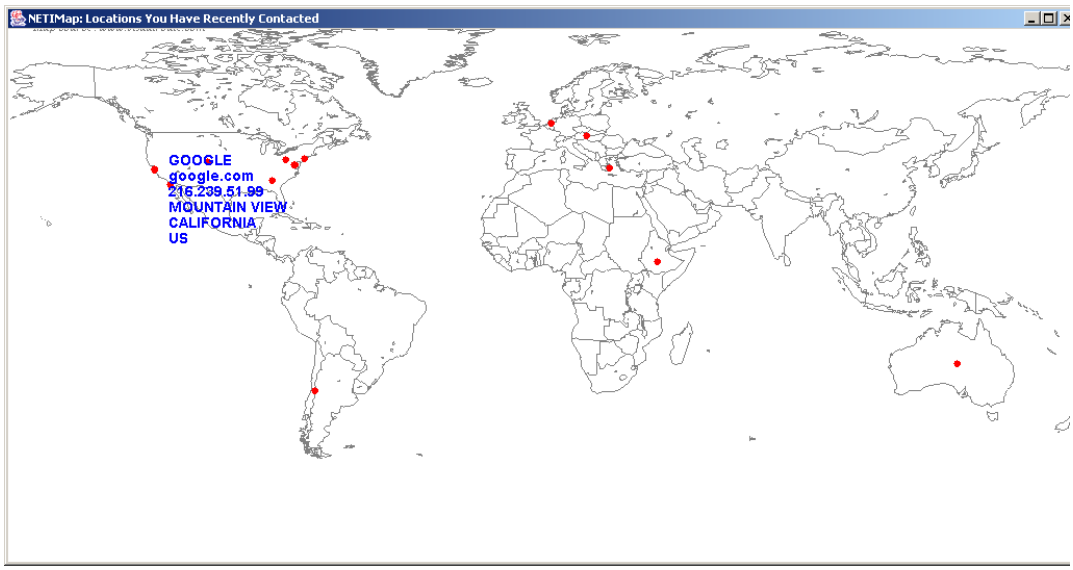


Figure 4: Screenshot of the NETIMap application. In this screenshot several locations have already been contacted and graphed. The mouse cursor is also hovering over the dot representing a connection to `http://www.google.com/`, displaying the information for that website. This information includes the name, DNS entry, IP address, city, state, and country associated with the website, all retrieved from CAIDA's NetGeo database [51].

further verify that their privacy is protected.

3.1.7 NETIMap

One important goal 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.

To resolve IP addresses into latitude/longitude coordinates, the NetGeo database [51] 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 [51]. Finally, this program is written in Java for maximum portability and for compatibility with the NetGeo database.

3.2 Network Statistics Collected

NETI@home collects many different statistics from end-users, in accordance with CAIDA specifications [7]. These statistics focus on four transport-layer protocols: 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. These sections primarily discuss the statistics gathered by the latest version of NETI@home as of the time of this writing, version 2.0. For more information about the statistics gathered by earlier versions of NETI@home, please see [69]. The statistics discussed, as well as others, including those from previous versions, are also represented in the structures shown in Appendix A.

3.2.1 Per User Report

The following statistics are those that are collected only once per user report sent to the NETI@home server. Many of these statistics are not directly related to a particular transport-layer protocol. However, the collection of these statistics will benefit Internet research.

3.2.1.1 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 Appendix A.

3.2.1.2 Arrival, Start, and Send Times

The *Send Time* is the time the data was sent to the Georgia Institute of Technology, as recorded by the user's system. The *Arrival Time* is the time the data arrived at the Georgia Institute of Technology, as recorded by the NETI@home server. By observing both the send

and arrival times, major timing discrepancies between the Georgia Institute of Technology and the user's system can be accommodated. This will be most beneficial when analyzing connection start and end times. The *Start Time* is the time the current report began recording. These times are precise to the level of detail allowed by `libpcap`, which, in turn, is as precise as the user's system allows. Typically, the level of precision is on the order of microseconds, with a granularity of milliseconds. The start and send times are reported in the NETI@home packet header, as shown in Appendix A.

3.2.1.3 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 hosts other than those running Microsoft operating systems, the GNU autoconf tool [26] is used to determine the operating system type at build time. For hosts 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 Appendix A.

3.2.1.4 Unique Identifier

When NETI@home is first installed on a user's system the initial install time is recorded. This initial install time can then be used as a unique identifier, as it is highly unlikely that two hosts will install NETI@home at the same second. Further, this time can be used to determine how long a particular user has had NETI@home installed on their system.

3.2.1.5 Privacy Level

In addition to implementing the privacy setting on each IP address collected, the privacy level is reported in the NETI@home packet header. This allows researchers to determine quickly and easily what privacy levels are selected by users. Further, this removes any ambiguity in privacy setting should any additional security approaches be used in the future.

3.2.1.6 Geographical Information

NETI@home users are able to voluntarily specify a geographical location. Should the specified location be in the United States, the user can further specify a US ZIP code in which they are located. Both of these results are then reported in the NETI@home packet header. Researchers should keep in mind however, that these are user-supplied details and may be inaccurate. In several studies presented later, we have augmented the user-supplied information with other techniques such as reverse DNS.

3.2.1.7 Datalink Type

NETI@home records the datalink type of the interface on which it is monitoring.

3.2.1.8 Packet Counts

Various packet types are also counted for reporting as well as the total number of packets observed. These packet types include: discarded packets, dropped packets, TCP packets, UDP packets, ICMP packets, IGMP packets, IPv6 packets, fragmented packets, ARP packets, IPX packets, EAPOL packets, and "other" packets. Further, a count is maintained for Wifi management, control, and data packets. So, although NETI@home may not provide detailed information about some of these packets, a count is maintained for comparison.

3.2.2 All Analyzed Flows

The following sections discuss statistics that are collected for all analyzed flow types: TCP, UDP, ICMP, and IGMP.

3.2.2.1 IP Addresses

NETI@home is able to collect the local and remote IP addresses for all analyzed packets directly from the packets.[1] IP addresses uniquely identify an Internet host, much like a telephone number uniquely identifies a telephone customer.

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

3.2.2.2 Local Netmask

NETI@home collects the netmask of the local interface that it is monitoring. This netmask is used by NETI@home to implement the medium privacy setting, if chosen by the user. Generally, a netmask is a bitmask used by a system to know which IP addresses belong to its particular subnetwork.

3.2.2.3 Times

NETI@home records the times that the flows are established and closed, or the time of the packet in the case of IGMP. These times are precise to the level of detail allowed by `libpcap`, which, in turn, is as precise as the user's system allows. Typically, the level of precision is on the order of microseconds, with a granularity of milliseconds.

3.2.2.4 Ports

NETI@home is able to collect the local and remote ports for TCP and UDP flows directly from their respective packets. The collection of ports aids researchers in determining what application is transmitting the data, such as HTTP (TCP port 80), DNS (TCP and UDP ports 53), or SSH (TCP port 22) traffic.

3.2.2.5 Checksums

NETI@home is able to determine if an IP, TCP, UDP, ICMP, or IGMP checksum is recorded and, if so, if it is correct.[2] A checksum is used to determine if the packet has been corrupted during transmission.

3.2.2.6 Number of Fragmented Packets

NETI@home maintains a counter of the number of fragmented packets sent and received 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.

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

3.2.2.7 Minimum and Maximum TTL Values

NETI@home records the minimum and maximum TTL (time to live) values for both the local and remote hosts. 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.

3.2.2.8 Number of Don't Fragment Flags

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

3.2.3 Transmission Control Protocol

TCP is one of the 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 represented in the *TCP_Stats* structures shown in Appendix A.

3.2.3.1 Number of Packets

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.

3.2.3.2 Number of Bytes

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.

3.2.3.3 Number of Acknowledgment Packets

NETI@home maintains a counter of the number of packets that are sent from the host on which it is running with the acknowledgment flag set as well as those received.

3.2.3.4 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. The number of duplicate acknowledgment packets sent by the host on which NETI@home is running, as well as those received, are recorded.

3.2.3.5 Number of Double Duplicate Acknowledgment Packets

NETI@home determines that a double duplicate acknowledgment has occurred if it sees three sequential acknowledgment packets with the same acknowledgment number. Double duplicate acknowledgments usually occur as a result of loss. The number of double duplicate acknowledgment packets sent by the host on which NETI@home is running, as well as those received, are recorded.

3.2.3.6 Number of Triple Duplicate Acknowledgment Packets

NETI@home determines that a triple duplicate acknowledgment has occurred if it sees four sequential acknowledgment packets with the same acknowledgment number. Triple duplicate acknowledgments usually occur as a result of loss and cause TCP to reduce the amount of information that is sent at a time (by reducing its congestion window). The number of triple duplicate acknowledgment packets sent by the host on which NETI@home is running, as well as those received, are recorded.

3.2.3.7 Number Beyond Triple Duplicate Acknowledgment Packets

NETI@home records the number of times it observes sequentially sent and received acknowledgment packets with the same acknowledgment number beyond the above-mentioned triple duplicate acknowledgment packets.

3.2.3.8 Number of URG Flags

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

data.

3.2.3.9 Number of PUSH Flags

NETI@home maintains a counter of the number of packets sent and received 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.

3.2.3.10 Number of ECN ECHO Flags

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

3.2.3.11 Number of CWR Flags

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

3.2.3.12 SACK Permitted

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

3.2.3.13 Minimum, Maximum, and Average Advertised Window Sizes

NETI@home records the minimum, maximum, and average advertised window sizes for both the local and remote hosts. The advertised window size can be directly observed in the TCP packet header and indicates the amount of data that a host can have outstanding, that is, unacknowledged, at a given time.

3.2.3.14 Number of Packet Retransmissions

NETI@home maintains a counter of the number of packet retransmissions sent and received 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.

3.2.3.15 Number of Bytes Retransmitted

NETI@home records the number of bytes that are retransmitted, in either direction, as a result of packet retransmissions.

3.2.3.16 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 can be used as a rough estimate of a TCP timeout.

3.2.3.17 Minimum, Maximum, Average, and SYN 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. This method for round trip time estimation only works if NETI@home is running on the host from which data is being sent.

A SYN round trip time is estimated using the TCP three-way handshake. This estimate is made by determining the amount of time that passes between sending a TCP SYN packet and receiving the corresponding SYN/ACK packet (for the host establishing the connection) or the amount of time that passes between sending a TCP SYN/ACK packet and receiving the corresponding ACK packet (for the host with whom the connection is being established). This method is based on the work in [35].

3.2.3.18 Connection Establishment Method

NETI@home records the method by which the flow was established. From NETI@home's perspective, a TCP flow can either be established by receiving a SYN packet, sending a SYN packet, or observing no SYN packet.

3.2.3.19 Connection Closure Method

NETI@home records the method by which the flow was closed. 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 [7].

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.

3.2.3.20 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.

3.2.3.21 Maximum Segment Sizes

NETI@home records the local and remote hosts' maximum segment sizes (MSS), if reported, for the TCP flow. The MSS is sent from one host to the other at the beginning of the connection to indicate the maximum size of TCP segment that can be received by the sending host.

3.2.3.22 Minimum, Maximum, and Average Packet Sizes

NETI@home records the minimum, maximum, and average packet sizes observed for packets sent and received. The packet size can be directly observed in the TCP packet header.

3.2.3.23 Window Scaling

The values of window scaling specified by both the remote and local hosts are recorded. Window scaling enables a scaling factor to be applied to the advertised window size so that the values can be larger than the size of the advertised window field alone.

3.2.3.24 Number of Failed Connections

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

3.2.4 User Datagram Protocol

UDP is another widely used protocol 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 represented in the *UDP_Stats* structures shown in Appendix A.

3.2.4.1 Number of Packets

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.

3.2.4.2 Number of Bytes

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

3.2.4.3 Minimum, Maximum, and Average Packet Sizes

NETI@home records the minimum, maximum, and average packet sizes observed for packets sent and received. The packet size can be directly observed in the UDP packet header.

3.2.5 Internet Control Message Protocol

ICMP is used for managerial purposes on the Internet. For instance, ICMP messages are sent when a packet's 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 represented in the *ICMP_Stats* structures shown in Appendix A.

3.2.5.1 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.

3.2.5.2 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.

3.2.6 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 represented in the *IGMP_Stats* structures shown in Appendix A.

3.2.6.1 Multicast IP Address

NETI@home is able to collect the multicast IP address for IGMP connections directly from the IGMP packet.[3] Multicast IP addresses are assigned to IGMP groups and are used to address the entire multicast group.

3.2.6.2 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.

3.2.6.3 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.

3.2.6.4 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.

3.2.6.5 Packet Directionality

NETI@home records the directionality of the IGMP packet, be it from local to remote host or vice versa.

3.3 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,

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

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 `libpcap` packet sniffing library 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 Institute of Technology School of Electrical and Computer Engineering. Each participant was given a complete copy of the NETI@home source code, a description of each file in the code, and a comment sheet. In addition to the code review, the NETI@home source code is available from the NETI@home website [68] in the hope that others will review the code and make suggestions.

3.3.1 NETI@home Client

The NETI@home client forms the core of the NETI@home software package. Packets sniffed by the NETI@home client 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 or 24 hours has passed, the data is compressed and transmitted to the Georgia Institute of Technology (`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 to ease implementation. Data compression is performed using the `zlib` compression library [43]. 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, privacy level, geographical location, US ZIP, and desired monitoring interface, as well as the user's email address should they wish to receive NETI@home

related mailings. The NETI@home client code reads this file and then uses the settings appropriately. To implement the medium privacy level, the local netmask is combined with the IP address using a bitwise AND, thus masking out the host portion 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.

3.3.2 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 the NETILogParse and NETIMap applications batch files are written by NSIS. 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.

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, much like a daemon on other systems. A screenshot of NETI@home running as a service is shown in Figure 5.

The NETITray application was developed to aid the manipulation of the `neti.conf` file, to remind users that the NETI@home program is running, and to allow them to halt NETI@home. 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

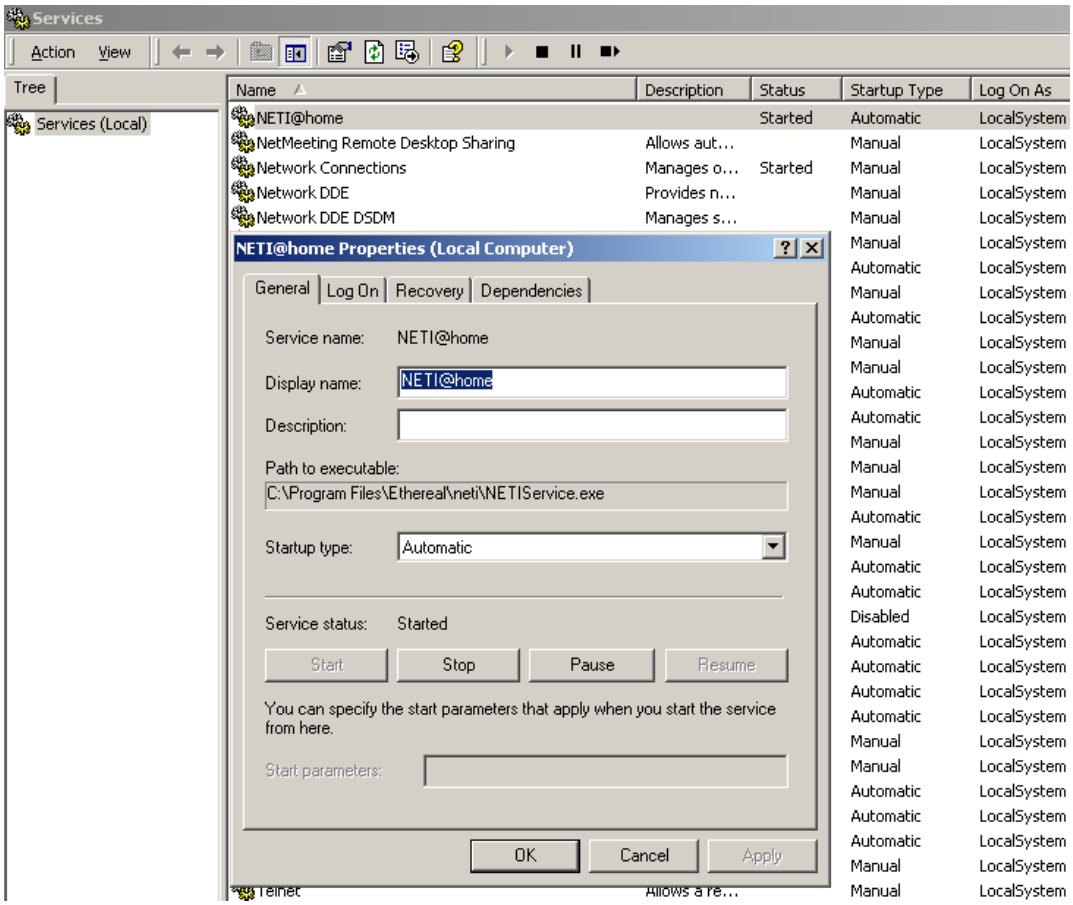


Figure 5: Screenshot of NETI@home displayed in the Windows Services Control Panel applet (background). The properties of the NETI@home application are also displayed (foreground). Running as a Windows service allows NETI@home to start at boot time and run whether or not a user is logged in on systems running Microsoft Windows.

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 Nullsoft Scriptable Install System (NSIS) [85]. 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 services. NSIS 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. Self-extracting executables created using NSIS are configured via a powerful scripting language.

3.3.3 Linux, Unix, Mac OS X, and Others

Implementation of NETI@home for Linux, Unix, Mac OS X, 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, Mac OS X, and Sun's SunOS 5.8, to name a few.

On Linux, Unix, Mac OS X, and other systems, NETI@home resides in the `/sbin` directory, as only users with root access can start NETI@home to protect user privacy and because NETI@home is essentially a network 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 [26] 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 the Georgia Institute of Technology.

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-2.0-1.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.

Finally, for users of the Macintosh OS X operating system, NETI@home is available as a Mac OS X package. Creating the OS X package is similar to creating an RPM as a configuration file and directory locations for files must be created and specified. The resulting package allows users to install and uninstall the NETI@home software.

3.3.4 NETI@home Server

To collect the incoming client data at the Georgia Institute of Technology, 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 those 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 Appendix A. 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 `xferrari.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.

In addition to collecting user measurement reports, the NETI@home also collects email addresses that are voluntarily given by NETI@home users. These reports, formatted as shown in Appendix A, are not associated with user data. Once collected, these email addresses are added to a mailing list so that users can be contacted about important information relevant to NETI@home.

A final task of the NETI@home server is the handling of update alert requests. Periodically, the NETI@home client queries the NETI@home server using TCP on port 558. Once a connection is established, the NETI@home server reports the latest version of NETI@home to the client, in ASCII format. Thus, users are alerted once a new version of NETI@home is available and can upgrade if they wish.

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.

3.3.5 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, many Windows systems do not have Sun's Java Virtual Machine installed, but rather have Microsoft's built-in `jview`. Thus, 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 [51] 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).

3.4 Distribution of NETI@home Software

One major goal of the NETI@home project is to have the NETI@home software installed in many end-user systems around the world. As the number of users increases, so does the value of the collected data. NETI@home is currently available for the Linux and Unix operating systems, in RPM and tarball format, for the Windows operating system in the form of a self-extracting executable, and for the Macintosh OS X operating system in the form of a Mac OS X package. All distributions are available from the NETI@home website [68].

Since NETI@home is copyrighted (copylefted) under the GNU General Public License (GPL) [25], the source code is also available from the NETI@home public website [68]. 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.

3.4.1 SourceForge

A project on the SourceForge website [77] 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 has a ranking of 86 in terms of traffic for websites around the world. 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 Institute of Technology NETI@home website [68]. 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.

3.4.2 `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, research results from data analyses, 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 Institute of Technology project. The Web server which hosts this website is maintained by the Georgia Institute of Technology's Office of Information Technology 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 MD5 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 MD5 hash of their file to the official hash on the website.

3.4.3 Publicity

As previously stated, a major goal 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 [72]. To attract a more mainstream audience, efforts were made

by the Georgia Institute of Technology Office of Institute Communications and Public Affairs to have NETI@home mentioned in several mass-media publications. The response was quite good, with publicity including Wired [18] and the popular Slashdot website [13, 83], among others. Continuing efforts are being made toward NETI@home related publications in other research conferences and journals, as well as mass-media publications.

CHAPTER 4

NETWORK SECURITY

This chapter presents our work pertaining to network security. First, flow-based observations are made comparing data from NETI@home to those collected by the Georgia Tech Honeynet. Next, the discovery of a potential covert channel of communication used by botnets is discussed. Finally, observations are made about the security implications of running a custom server, specifically the NETI@home server.

4.1 Flow-Based Observations from NETI@home and Honeynet Data

In [27], the authors present a flow-based comparison of the traffic seen on the Georgia Tech Honeynet, representing malicious traffic, to the data collected by NETI@home, representing typical end-user traffic. This comparison was made to aid in understanding what makes up the majority of the malicious traffic on the Internet. Comparing and contrasting these results can initiate a better understanding of the malicious traffic seen on the Internet.

4.1.1 Overview

The Internet has grown from the small ARPANET to an unfathomably large network. As with any new technology, the Internet has grown from its infancy to a stage where security concerns become a considerable problem. Today's Internet is plagued with a plethora of worms, viruses, malware, spam, and otherwise malicious traffic. In this section, we make observations about end-user Internet activity by comparing honeynet traffic and NETI@home traffic in order to better understand the security problems of the Internet.

Our strategy for understanding the malicious Internet traffic is a flow-based analysis of several years of honeynet data and NETI@home data. We study a number of metrics visually over large timescales and plot both the honeynet dataset and the NETI@home dataset and then compare the results. Some interesting points include flow activity across

the IP address space, port scan activity, new and lingering worm traffic, as well as other observations. Below we provide some background information on the datasets used.

4.1.1.1 NETI@home Data

Some of the analysis presented in this section requires using only low or medium privacy statistics and may skew the results slightly, but we feel that our user base is large enough that such skewing is minimal.

The NETI@home dataset we are analyzing was collected from June 1, 2004 to February 28, 2005 and consists of reports from at least 500 uniquely identifiable users. There are approximately 31 million TCP flows and 33 million UDP flows in this dataset, constituting 65 GB of transferred network traffic. The remaining flows consist of 600 thousand ICMP flows and 250 thousand IGMP flows.

4.1.1.2 Georgia Tech Honeynet Data

A honeynet is a network of resources whose value lies in the illicit use of those resources. All network traffic to and from a honeynet is suspicious, but a small amount of traffic may be legitimate. However, most of the traffic on a honeynet is malicious in nature.

The Georgia Tech Honeynet Project was launched in the summer of 2002 and immediately began collecting data [24]. The dataset we are using consists of nearly three years of honeynet traffic with very few service interruption points for maintenance and upgrades. All network traffic to and from the honeynet has been logged and archived, including the traffic between the honeypots.

To better understand the conclusions we draw from this data, it is important to understand the network on which this honeynet has been deployed. There are over 15,000 students enrolled at the Georgia Institute of Technology and approximately 5,000 staff and faculty employed. The supporting network consists of more than 40,000 networked systems all within the Georgia Institute of Technology's IP address space. The honeynet has been deployed within this IP address space and is accessible from internal machines within the Georgia Institute of Technology address range as well as external machines.

The honeynet dataset we are analyzing was collected from August 19, 2002 to February

28, 2005 and consists of reports from 38 unique IP addresses. There are approximately 2 million TCP flows and 350 thousand UDP flows constituting 7 GB of transferred network traffic. The remaining flows consist of 40 thousand ICMP flows and no IGMP flows. During this time period there have been on the order of ten compromises.

4.1.1.3 Observing Malicious Traffic

We visually compare the network flows of a honeynet against the network flows in the NETI@home data. In particular, we make observations to try and answer these three questions:

- What are some of the characteristics of the malicious traffic observed on the Internet?
- How much malicious traffic is seen by end-users on the Internet?
- Are there identifiable sources of malicious traffic on the Internet?

4.1.2 Network Flow Analysis

In order to compare the NETI@home dataset with the honeynet dataset, we ran a customized version of the NETI@home client on our honeynet data to convert the `libpcap` [33] format honeynet dataset to flow-based statistics in the same format as the NETI@home dataset. This conversion made it possible to directly compare the NETI@home dataset to the honeynet dataset. In this section, we describe some of the statistics that are provided by the NETI@home client.

The NETI@home client collects statistics for four common transport layer protocols: TCP, UDP, ICMP, and IGMP. Much of our analysis focuses on TCP flows since they make up the majority of the traffic seen in our datasets. However, some data from UDP, ICMP, and IGMP are also presented in our results.

The analysis technique is centered around the concept of a bidirectional flow, based on the commonly used 5-tuple, which consists of the source and destination IP addresses, source and destination ports, and the transport layer protocol. Statistics gathered for each TCP flow include various time measurements, the number of packets sent and received, the source and destination parameters, failure flags, window size measurements, and various

other information as discussed in Chapter 3. Similar statistics are gathered for the flows that are of the other types of transport layer protocols. A full discussion of the statistics gathered can be found in [72].

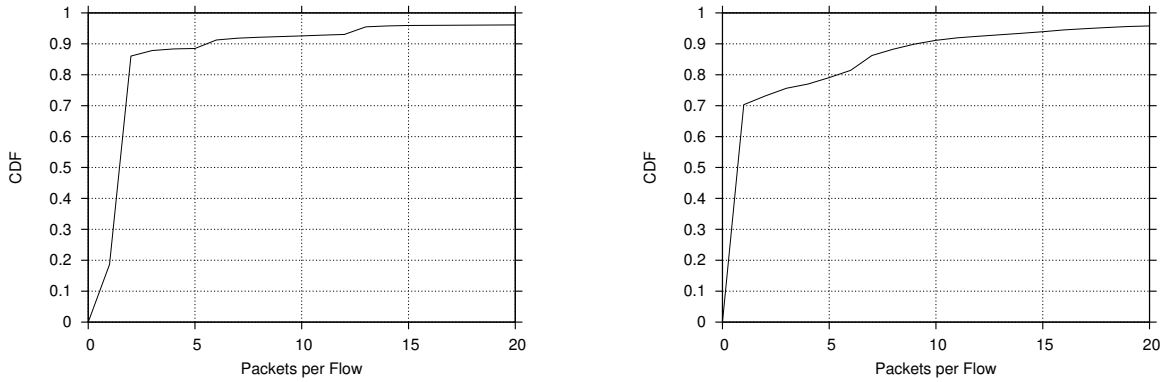
Each flow has a *local* IP and port number and a *remote* IP and port number. *Local* refers to the host on which the client is running and collecting statistics from. *Remote* refers to the other host in the flow. Therefore, if a NETI@home user with IP x makes a Web request to a given IP y, then x would be the local IP and y would be the remote IP. To further clarify, if the same NETI@home user was scanned by IP z, then x would still be the local IP and z would be the remote IP.

There are several sources of bias in our datasets that may skew our results and are worth mentioning. First, an insignificant number of NETI@home users had their clocks misconfigured so we did not include them in the results. Clock synchronization in general is a source of bias. Second, we did not include all IP results from NETI@home users when their privacy was set to high because their IP addresses are unknown. Third, the honeynet dataset is known to be complete; however, the NETI@home dataset relies on the end-users to run the NETI@home client to monitor their systems and so may have some incomplete results. Fourth, the NETI@home users must volunteer to run the client, so the data are not a truly random sample of Internet end-users. Finally, the honeypots are all on the same network, whereas NETI@home users are spread throughout the Internet.

After collecting the flow statistics for both datasets, we created a framework to analyze the data. This framework allowed us to plot various graphs for both datasets for comparison. Below, we present these graphs and discuss our observations.

4.1.3 Data Observations

In order to aid in understanding what makes up the majority of the malicious traffic on the Internet we have plotted various metrics for both the honeynet dataset and the NETI@home dataset. The NETI@home dataset represents a mixture of both legitimate/good traffic as well as malicious traffic. The honeynet dataset represents almost entirely malicious traffic. Comparing and contrasting these results can initiate a better understanding of the malicious



(a) Honeynet

(b) NETI@home

Figure 6: CDF of the number of packets per TCP flow.

traffic seen on the Internet.

4.1.3.1 Number of Packets Per Flow

We first graphed the cumulative distribution function (CDF) of the number of packets for all TCP flows for each dataset. The results are shown in Figure 6. First, observe the Honeynet curve. One can see two distinct inflection points for packet counts equal to one and two. TCP flows that consist of just one packet most likely contain one SYN packet. It is possible to have a single packet flow that is not a SYN packet. For instance, an RST or SYN/ACK packet could be received from a host that received a spoofed connection attempt. Upon further investigation, we did not observe many flows of this nature.

TCP flows that consist of two packets most likely consist of one SYN and one RST packet or one SYN and one SYN/ACK packet with no final ACK to complete the three-way handshake. Again, there are other combinations of TCP flows consisting of just two packets, but we have not observed many of these combinations. Any TCP flow consisting of two or fewer packets is a failed connection. On a honeynet, we consider these failed connections to be malicious probes. Therefore, on our honeynet dataset, about 87% of all TCP flows can be considered to be probes.

We can contrast the NETI@home CDF with the honeynet CDF and see that about 73% of all TCP flows can be considered failed connections. In the NETI@home dataset,

not all of these failed connections are necessarily malicious probe packets as they may be legitimately failed connections. However, it is interesting to note that in terms of number of packets per flow the majority of observed TCP flows for end-users are either probes or failed connections.

4.1.3.2 TCP Port Histogram

To better understand what ports and services malicious flows are targeting, we have generated a TCP port histogram over time for both the honeynet dataset, as seen in Figure 7, and the NETI@home dataset, as seen in Figure 8. Each row of points represents one day. The width of the rows span the local TCP ports from 0 to 1024, which are the well-known ports [63]. The following formula was used to create the graphs, where i is the intensity value for a given point in a given row:

$$i = \begin{cases} 0 & \text{if } c = 0 \\ 0.75 \cdot \left(\frac{c}{c_{\max}}\right)^{0.45} + 0.25 & \text{otherwise} \end{cases} \quad (1)$$

The maximum number of packets destined to a certain port on one day (i.e., one row in the figure) is denoted $c_{\max}$. A port with a packet count of c is then visualized with intensity i according to the above formula. If c is zero, the intensity is also set to zero (black). Otherwise, the intensity is chosen to be a value between 25% gray ($i = 0.25$) to white ($i = 1.0$, for the port where $c = c_{\max}$). The exponent is used to boost dark pixels to make them more visible. We choose to represent no activity with dark regions because it provides better contrast for the faint areas of activity.

There are a number of observations to be made from these graphs. Two important characteristics of the figures to observe are the horizontal lines and the vertical lines. First, the horizontal lines represent port scans. Port scans are often malicious in nature as an attacker will generally use a port scan against a target to determine that target's weaknesses. In the honeynet data, a number of port scans can be seen over time, but the NETI@home dataset shows a significantly denser number of port scans seen over time. This appears to be intuitive as there are an order of magnitude more NETI@home users, which are distributed across the Internet both topologically and geographically, than there are honeypots in our

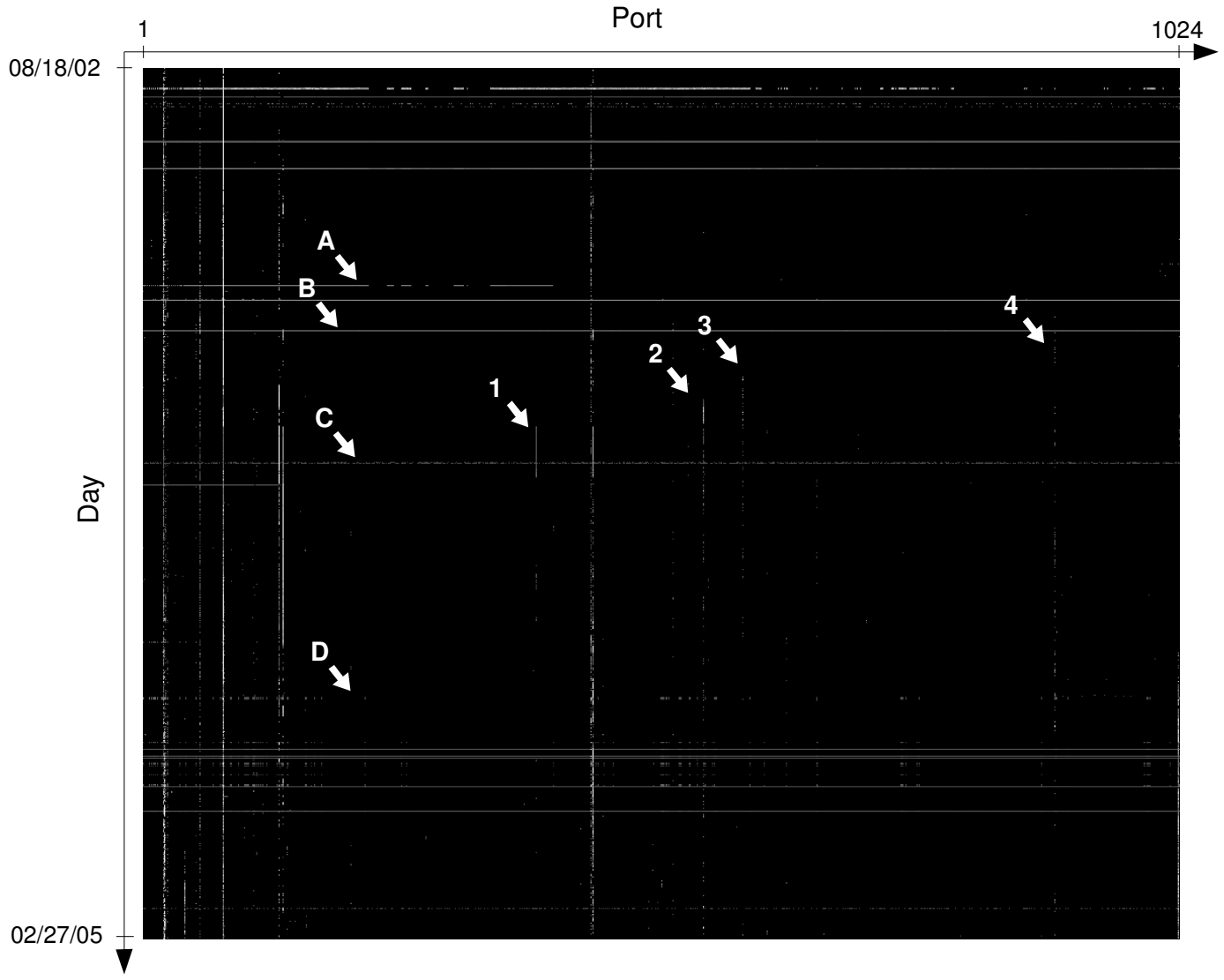


Figure 7: Honeynet TCP port histogram.

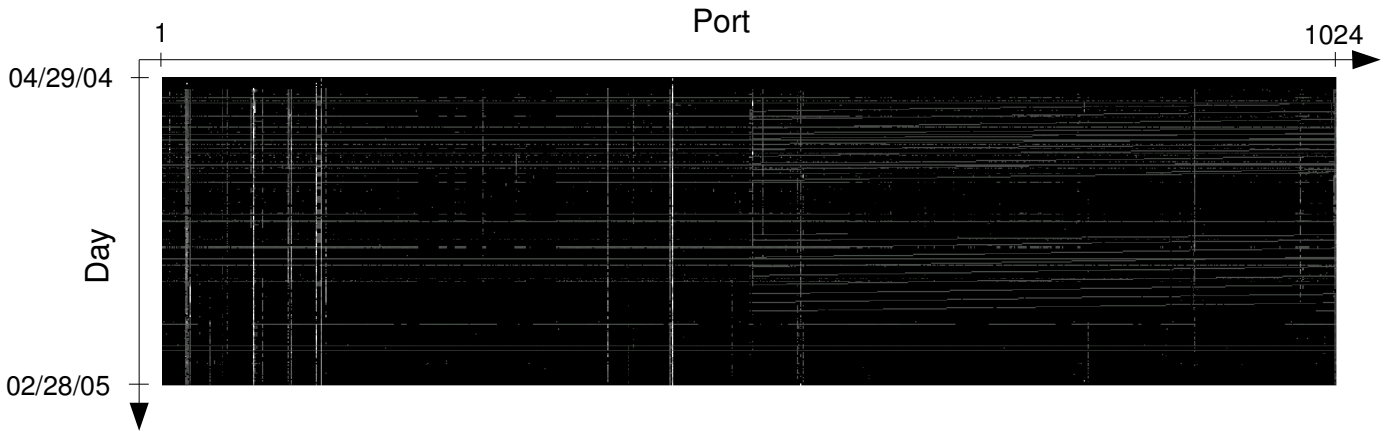


Figure 8: NETI@home TCP port histogram.

dataset. Some factors that would decrease the number of port scans seen by NETI@home end-users include firewalls, NATs, or other similar configurations. Even with these factors, some NETI@home users are seeing similar port scans as seen on our honeynet.

Another interesting observation is that there are a number of different types of scans seen. At least four different port scans are easily distinguished visually in the honeynet data, as denoted by the letters $A - D$, and similar scans are observed in the NETI@home data. The most naive port scan will scan all ports (B). The more sophisticated port scans will skip ports that are of little interest (A, C, and D). There are a number of widely available port scanning tools, which offer various options for the scanning algorithm [54, 55].

One interesting difference seen in the horizontal lines in the NETI@home dataset is the stair step lines from approximately port 512 through 1024. Since the user who reported these flows was within the Georgia Institute of Technology network and used a low privacy level, we were able to determine what caused the stair step lines. An administrative machine within the Georgia Institute of Technology network was scanning ports 512 through 1024 over the course of several days. The algorithm consists of dividing the ports into a number of ranges and scanning one range each day. The source of the scanning was a machine used to help secure the network and so was altruistic. Therefore, we do not consider these scans to be malicious in nature.

The second interesting aspect to observe in these graphs is the vertical lines. The

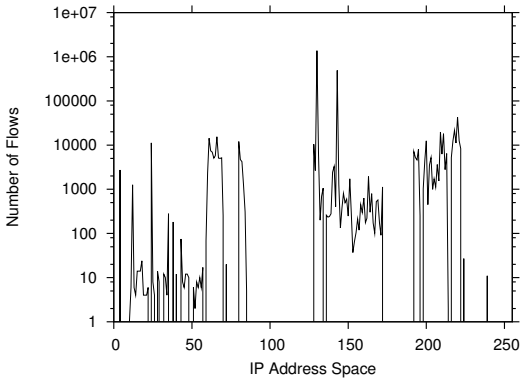
vertical lines represent ports that have continual traffic over large time scales. Looking at the honeynet graph from left to right, the most prominent TCP ports with continual traffic are 22 (SSH), 80 (WWW), 135 (Microsoft Windows Service), 139 (Microsoft Windows Service), and 445 (Microsoft Windows Service). Most of these ports have been a target of one or more worms in the past in addition to legitimate traffic.

There are a number of other vertical lines that are not as prominent in the honeynet dataset, as seen in Figure 7. The vertical line denoted by '1' is LDAP traffic and was only seen for a short period of time. The line denoted by '2' represents traffic seen from the real-time service protocol worm. The traffic at '2' is particularly interesting in the honeynet dataset. One can notice a bright burst of traffic starting on the worm release date that continues with intensity over the course of the next several days. After a number of days, the worm traffic slowly fades out as the infected machines are repaired. However, trailing effects of the worm can be seen from the point of release until the end of the dataset, which is over the course of more than a year. Therefore, we see that lingering worm traffic exists on the Internet for long periods of time after the initial release date.

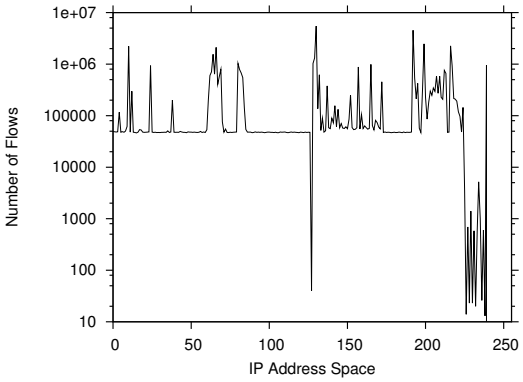
The line denoted by '3' represents traffic seen from the blaster worm, as seen in Figure 7. This line also continues for a long period of time, although its characteristics are not as distinguishable as the real-time service protocol worm. In the honeynet data, it is not clear why traffic is seen at the line denoted by '4' at port 901. This may be traffic targeting an old Trojan port, RealSecure's management port, or Samba/SWAT on RedHat Linux-based boxes. It is interesting to note that these trends seen in the honeynet data are repeated in the NETI@home data in addition to the legitimate traffic, as seen in Figure 8, although it is difficult to distinguish between legitimate traffic and worm traffic in the NETI@home dataset.

4.1.3.3 IP Address Space

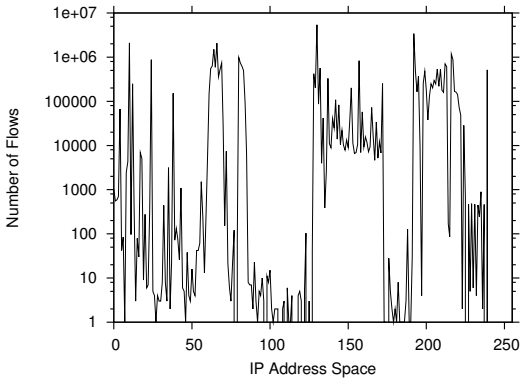
The graphs in Figure 9 show where the traffic is coming from or going to within the entire IP address space. The IP address is divided into 256 buckets based on the first byte of the IP address. Figure 9(a) shows the honeynet graph. It is clear that certain portions



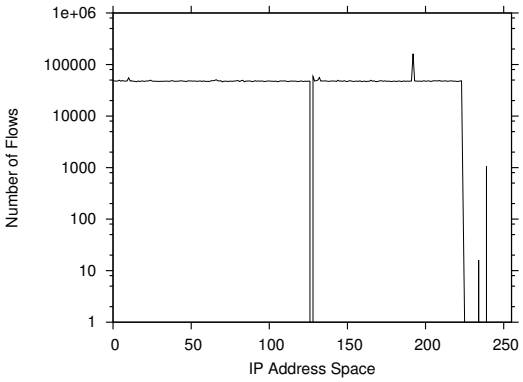
(a) Honeynet (all IP flows)



(b) NETI@home (all IP flows)



(c) NETI@home (no TCP port 445 flows)



(d) NETI@home (only TCP port 445 flows)

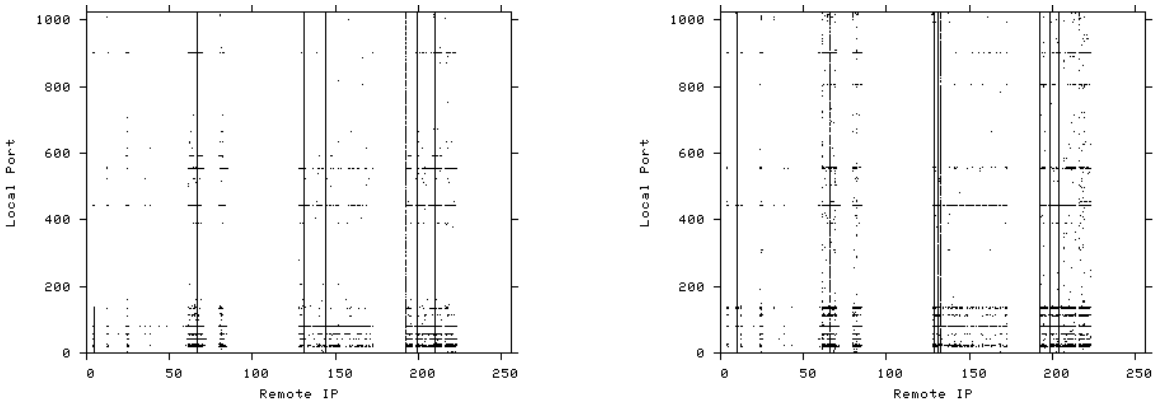
Figure 9: IP address space distribution by number of flows.

of the address space have seen zero activity on the honeynet. These portions correspond to unallocated addresses as listed in the WHOIS database. Given that there are no flows from most of these spaces to the honeynet, we conclude that there are not many spoofed IP packets coming from unallocated IPs to our honeynet. Further, either the number of packets with spoofed IP addresses coming to our honeynet is low or they are intelligently designed.

The NETI@home dataset has an additional baseline of traffic seen across most of the address range, as seen in Figure 9(b). Further investigation found that this baseline is caused by one or more NETI@home users sending out a large number of TCP flows to TCP port 445 over a short period of time. We are unsure how many users were reporting these results because of privacy settings. Figure 9(d) shows the number of flows to TCP port 445 versus the IP address space. There is clearly a horizontal line across the majority of the IP address space, which suggests that the NETI@home user or users were randomly scanning the IP address space on TCP port 445. The nature of this scanning may have been malicious in nature. For example, the user may have been infected with a worm, as there have been worms that target TCP port 445. However, we cannot conclude for certain that the traffic was malicious in nature.

In Figure 9(d), there is a small increase in traffic at bucket number 10. This is probably due to local 445 traffic on private 10.0.0.0/8 networks. Similarly, there is an increase in traffic at bucket number 192. This increase would be due to local 445 traffic on private 192.168.0.0/16 networks. The sharp drop in traffic at bucket number 127 is due to the fact that the 127.0.0.0/8 network is the dedicated localhost network. Finally, the upper ranges of the IP address space did not see any scans. These ranges contain multicast, experimental, and other types of allocations.

To better compare the NETI@home data with the honeynet data, we graphed the NETI@home dataset filtering out traffic to TCP port 445, as seen in Figure 9(c). Comparing Figures 9(a) and 9(c), one can notice a striking similarity between the NETI@home data and the honeynet data. Some differences in the NETI@home data include traffic to the multicast range and some traffic in the unallocated ranges. However, visually the two



(a) Honeynet

(b) NETI@home

Figure 10: Remote IP address and contacted local TCP port.

graphs have notably similar shapes.

Based on our observations of the IP traffic seen relative to IP address space, we note a possible algorithm for detecting suspicious machines. In previous work, it was shown that a honeynet can be used to find compromised machines on large enterprise networks by marking any machine on the enterprise that attempts to connect to the honeynet as suspicious [42]. An extension that we draw from these graphs is that any machine attempting to connect to an unallocated IP address should be considered suspicious and may be compromised.

A graph of the remote IP versus local port for both datasets can be seen in Figure 10. Again, we only plot the well-known TCP ports. In these graphs, one can see that remote IPs that appear in the flows are spread across the allocated IP spectrum, and again there is little traffic in the unallocated ranges, even in the NETI@home data. Based on these graphs, we observe that scans come from across the entire allocated IP address space.

4.2 Potential Covert Communication

We have previously noted [27] that several NETI@home users appear to be infected with malware. Further, we have noticed suspicious behavior while studying ICMP traffic. While not certain, we believe that this behavior may be related to a new type of botnet that uses ICMP as a covert channel of communication. Specifically, we have noticed that several ICMP packets contain invalid type and/or code values. Figure 11 shows the percentage of

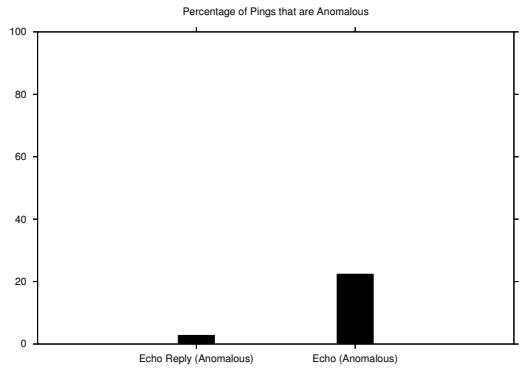


Figure 11: Anomalous echo by percentage.

observed ECHO_REQUEST and ECHO_REPLY ICMP packets (used for `ping` [53]) that contain anomalous code fields.

Further, we have observed several hosts that exhibit strange behavior after receiving these anomalous ECHO_REQUEST packets. Once received, data is transmitted on TCP ports 12345, 8081, and 5168 as well as UDP ports 2967 and 40116. After an additional 17 minutes have passed, traffic is present once again on TCP ports 12345, 5168, and 8081. This process then repeats after 30 minutes have passed since the initial ECHO_REQUEST packet. We have observed this traffic pattern persisting for an entire day at a time.

By design, NETI@home does not collect the data portion of ICMP packets, so a key element in determining the use of these anomalous packets is missing. However, we have noted a hypothetical botnet that could be at work. Such a botnet would have no need for a traditional IP or DNS-based command and control server. The commands could be sent to large portions of the Internet by sending out ECHO_REQUEST (ping) packets, which are usually deemed harmless. The anomalous code values of these packets could be used to identify compromised machines and issue further commands. If such a botnet is actually at work on the Internet, this would represent a more dangerous scenario than those posed by previous botnets, as this botnet would have no command and control server to disable.

ICMP ECHO_REQUEST and ECHO_REPLY packets have been used for malicious purposes previously, and these observations may be a result of similar activity. In [16], the authors describe a tool, Loki, created to use the data portion of ECHO_REQUEST and ECHO_REPLY packets as a covert channel of communication. ECHO packets have also

been used previously in the "Ping of Death" [36] and Smurf [9] attacks. Finally, tools such as [54, 55] use anomalous ECHO code values to determine the operating system of targeted hosts.

4.3 NETI@home Server Attacks

Finally, running a distributed network monitoring infrastructure has proven to yield some unexpected observations. For instance, after our initial publicity on Slashdot [13], the NETI@home server was subjected to several DoS attacks. Fortunately, little to no data was lost or compromised. However, such attacks demonstrate that higher profile hosts on the Internet are subject to specialized malicious attacks, even a custom-coded server such as the NETI@home server.

CHAPTER 5

NETWORK BEHAVIOR

The wealth of data collected by NETI@home has led to many interesting observations and analyses, although there are still many more avenues of investigation to pursue. This chapter highlights some of the more interesting observations and analyses related to network behavior that have been made using the NETI@home dataset.

The dataset used in this chapter consists of NETI@home data collected over a one-year period from October 1, 2004, to September 30, 2005. This dataset includes more than 36 million TCP flows, 93 million UDP flows, 1 million ICMP flows, and 660 thousand IGMP flows, as well as various other information about their corresponding hosts. Although an exact calculation is not possible because of privacy settings and dynamically assigned IP addresses, we estimate that this data was collected by approximately 1,700 users. These users represent a heterogeneous sampling of Internet users running some eight different operating systems and reporting from approximately 28 nations and 43 US ZIP codes.

As the Internet consists of a variety of machine types and operating systems, the NETI@home dataset, with its variety of users and operating systems, will be useful. Effort is made to determine what effect the differences in machine configurations has on the overall interactions on the Internet. Further, we study networking protocols to determine how well they perform on the Internet and how well they are implemented by each operating system. As each operating system tends to have its own custom network protocol stack, there should be some differences in the way protocols behave from operating system to operating system. Also, several protocols provide options that may or may not be implemented and used by the various operating systems.

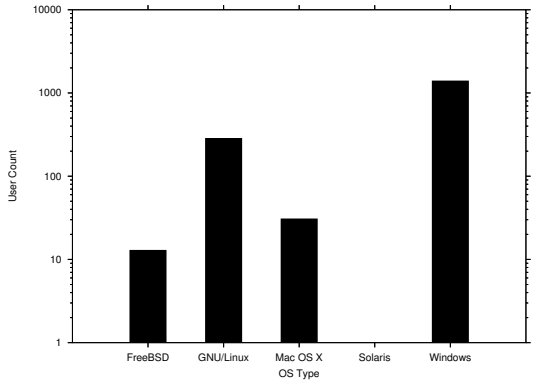


Figure 12: NETI@home user count by operating system.

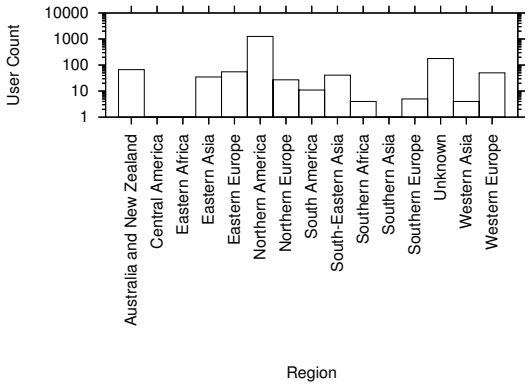


Figure 13: NETI@home user count by region.

5.1 General Observations

First, general observations have been made about the NETI@home user population and their use of the Internet. Figure 12 shows the distribution of operating systems run by NETI@home users. Figure 13 gives the distribution of geographical regions in which users are located. These are the geographical divisions specified by [79]. User location is determined by a combination of reverse DNS lookup on the user's IP address and the geographical location specified by the user. Finally, Table 1 and Table 2 show some of the more popular TCP and UDP ports by flow and the applications most commonly associated with these ports. Table 3 and Table 4 show the popular ports in terms of bytes transferred. In the future, we hope to determine long-term trends of application popularity.

Table 1: Popular TCP ports (by flows)

TCP port number	Common use	Percentage of TCP flows
80	HTTP (Web)	45.00
445	Win2k+ Server Message Block	26.64
4662	edonkey, emule (P2P)	5.26
6881	bittorrent	3.93
443	HTTPS	1.60
6346	gnutella	1.28
110	POP3	1.17
5678	rrac	1.12
557	NETI@home	0.87
139	netbios and trojans	0.86

Table 2: Popular UDP ports (by flows)

UDP port number	Common use	Percentage of UDP flows
53	DNS	36.37
162	snmptrap	29.66
137	netbios	12.96
138	netbios	5.05
6881	bittorrent	4.89
161	snmp	4.12
4672	xmule, rfa	3.39
9646	*Unknown*	2.86
9313	*Unknown*	2.07
69	tftp	1.59

Table 3: Popular TCP ports (by bytes)

TCP port number	Common use	Percentage of TCP bytes
80	HTTP (Web)	47.59
6881	bittorrent	14.16
4662	edonkey, emule (P2P)	10.36
3128	squid-http and trojans	1.56
443	HTTPS	1.48
139	netbios and trojans	1.04
22	SSH	0.98
10500	*Unknown*	0.92
8000	*Unknown*	0.86
8090	*Unknown*	0.70

Table 4: Popular UDP ports (by bytes)

UDP port number	Common use	Percentage of UDP bytes
1900	SSDP	22.09
162	snmptrap	19.44
53	DNS	16.07
138	netbios	7.75
137	netbios	5.95
6881	bittorrent	5.01
32770	Filenet NCH	3.39
1234	*Unknown*	3.39
67	Bootstrap Protocol Server	3.10
68	Bootstrap Protocol Client	3.10

5.2 Network Locality

For most flows, NETI@home records the minimum and maximum TTL values observed in both directions of a bidirectional flow. In this section, we will use the minimum and maximum TTL values that originate at the remote end-host of a connection to determine the network distance between the local and remote end-hosts.

Different implementations of network protocol stacks use different initial TTL values when sending a packet. Typically, these values are either 255, 128, 64, or 32; however, it should be noted that any value between 0 and 255 could possibly be used. As a packet travels from its source host to destination host, each router decrements this TTL value by 1. Should the TTL value reach 0, the packet will be dropped by the router and an ICMP error message will be sent to the sending host to indicate the failure. This technique prevents packets from persisting in the network in erroneous situations such as routing loops [59].

To calculate the hop count, that is, the number of routers transversed, between the source and destination hosts, we subtract the received TTL value from an assumed initial TTL value. To determine an estimate of the initial TTL value, we associate the received TTL value with the initial TTL value that is greater than the received TTL value by the least amount. For example, if a received TTL value is 110, we assume an initial TTL value of 128 and estimate the hop count to be $128 - 110 = 18$. Similarly, a received TTL value of 60 would give a hop count of $64 - 60 = 4$. We are aware that this approach could lead

to erroneous estimates; however, it does provide reasonable results in almost all cases, with exceptions noted below.

There were a few notable anomalies found while analyzing the NETI@home data using this method. First, we found an unusually high number of UDP flows (77%) that appeared to have a hop count of $255 - 150 = 105$. Upon further investigation we found that all of these flows were communications between hosts on a local area network. This led us to believe that the hop counts for such flows should be very small and we found that several commercial routers actually use an initial TTL value of 150, thus giving a hop count of $150 - 150 = 0$. We then modified our analysis to account for this finding.

Another anomaly found in the NETI@home data was the existence of several flows (3% of TCP flows) whose minimum and maximum TTL values crossed the initial TTL boundaries. The vast majority of these flows, when calculating hop counts, showed that their actual hop counts were not varying, but rather their initial TTL values. Upon further investigation, we found that 79% of the anomolous TCP flows were HTTP flows, with the remainder consisting of other high-traffic applications such as HTTPS and POP. Selecting several of these websites and conducting a manual analysis with Ethereal [14] confirmed that this behavior is indeed occurring in individual flows. No immediate explanation was found however, as some flows exhibited varying initial TTLs between TCP connection establishment packets (SYN / ACK) and all other packets while other flows exhibited differing behavior on duplicate acknowledgment packets only. Further, some flows exhibited this differing behavior between HTTP data and all other packets, and finally the remaining flows exhibited the behavior between seemingly randomly selected packets. One possible explanation for such behavior is that the user or an application is varying the initial TTL value during the flow. To investigate this possibility we searched throughout the source code of the popular open-source Apache Web server [80] and failed to find any code to modify the initial TTL. While having no confirmable explanation, we believe that such behavior may be the result of load-balancing by servers, despite the fact that it occurs within a single flow. Thus, one source of bias in our hop count study, in addition to our assumed initial TTL values, is the fact that we only record the minimum and maximum TTL values

Table 5: Initial TTL values

Protocol	255	150	128	64	32
TCP	4.54%	0.07%	31.36%	63.88%	0.15%
UDP	1.33%	76.90%	17.97%	3.19%	0.61%
ICMP	64.89%	0.03%	13.77%	21.06%	0.25%
IGMP	0.01%	0.00%	0.00%	0.03%	99.96%

Table 6: Hop count variation

Protocol	Average hop count variation	Flows with variation
TCP	0.15	3.74%
UDP	0.00	0.03%
ICMP	0.02	0.82%

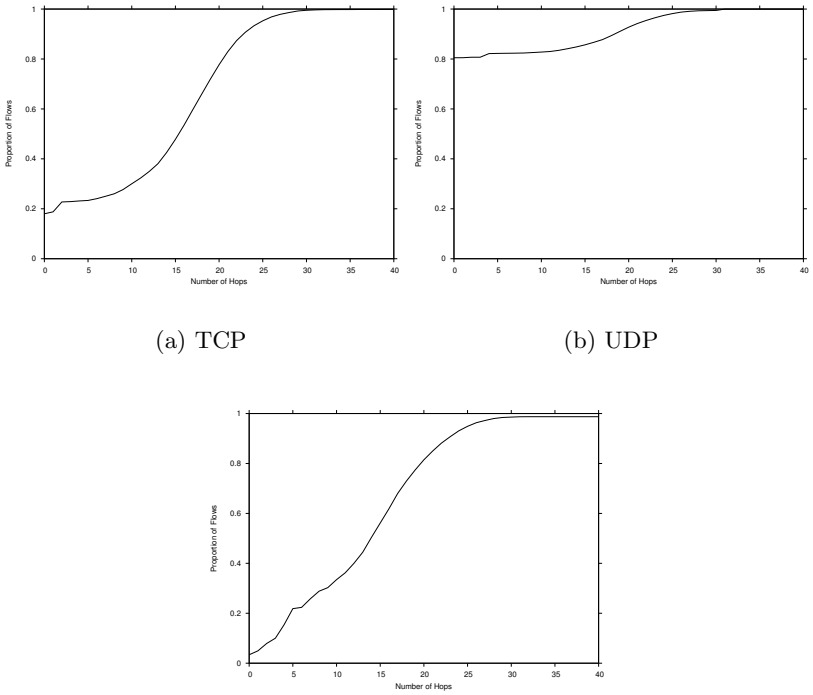
observed. In cases where the initial TTL value is varied, the true minimum and maximum hop counts may go unrecorded.

After identifying and analyzing these anomalies in our dataset, we calculated hop counts for all of the observed flows. We found that the average hop counts vary depending upon the protocol, most likely because some protocols are used more often in a local network setting. Table 5 summarizes the assumed initial TTL values and their frequency and Figure 14 plots the cumulative distribution functions of the average hop counts for TCP, UDP, and ICMP.

In addition to calculating the average hop counts observed, we analyzed the variation of the hop counts for the flows based on the minimum and maximum TTL values observed. Variations in the hop counts are most likely a result of routing changes over the flow's lifetime. Table 6 summarizes the average variations we observed for TCP, UDP, and ICMP flows as well as the percentages of these flows for which hop count changes occur.

5.3 Frequency and Use of Network Address Translation and Private IP Addresses

One of the strengths of the NETI@home project is that all connections are observed from the viewpoint of the end-user. Such a vantage point gives us the unique ability to observe local network traffic as well as the use of NAT and DHCP, which would otherwise be difficult



(a) TCP

(b) UDP

(c) ICMP

Figure 14: CDF of average hop counts.

to measure. Exercising this ability, we determine the number of NETI@home users that utilize NAT as well as DHCP. Further, we observe the number of NETI@home users who utilize the private IP address space [62] for their local network connection.

When a NETI@home user participates in a network connection we are able to observe their local IP address, provided their privacy setting allows such monitoring. Further, when a NETI@home user reports their data to the NETI@home server, we are able to determine their "external" IP address. Should these IP addresses differ, we determine that either NAT is being utilized or their address has been reassigned using a technique like that of DHCP [20]. We find that this occurs for 81.15% of our users.

Investigating the local IP addresses further, we calculated the percentage of NETI@home users with local IP addresses in the private IP address space. We have found that 77.30% of NETI@home users utilize an IP address reserved for private use. Thus, the majority of NETI@home users are members of local area networks, although these local networks may consist of one machine.

For the majority of NETI@home users whose local IP address differs from their external IP address, the IP address conversion is from an IP address in the private range to an IP address in the public range. However, we found that 17.01% of NETI@home users actually convert from one public IP address to another, in effect utilizing two public IP addresses. While some of these users may have been reassigned a new public IP address with a technique such as DHCP before reporting to the NETI@home server, there is evidence to suggest that this is not always the case. As there are a limited number of publicly available IP addresses, one role of NAT is to allow multiple machines to utilize a single public IP address. As is evidenced by these numbers, it appears that NAT is in fact helping to preserve public IP addresses for the time being. However, it also appears that NAT is being used when a public IP address has already been allocated to the offending host.

5.4 Adoption and Use of Selected Protocol Flags and Options

NETI@home records several statistics on various flags and options for TCP, UDP, ICMP, and IGMP as well as their underlying IP headers. In this section we discuss a selection of these flags and options, namely, TCP selective acknowledgment (SACK) capability, TCP window scaling capability, the specified TCP maximum segment size (MSS), the TCP urgent flag, the TCP push flag, and the IP don't fragment flag.

TCP SACK and TCP window scaling require capability on both sides of the network connection for functionality [30, 31, 46]. We found that TCP SACK capability is fairly common in the observed flows, with a significant number of flows having only one end-host capable. On the other hand, we found that TCP window scaling capability is not very common, although this option has been described for many years and has become a rate-limiting factor among TCP flows [89]. It is intuitive that the bandwidth-delay products of Internet connections will increase over time, thus increasing the need for the adoption of TCP window scaling. Table 7 summarizes these findings.

We also studied the MSS specified by the observed end-hosts of the TCP flows. We found that the average MSS specified was 467 and the median MSS was 1460, which corresponds to

Table 7: TCP option capability

Protocol option	Neither host capable	One host capable	Both hosts capable
TCP SACK	60.28%	20.92%	18.80%
TCP Window Scaling	96.62%	3.03%	0.35%

the Ethernet MTU. We also observed a minimum MSS of 24 and a maximum of 16856 bytes. Further, 63.87% of TCP end-hosts either did not explicitly specify a MSS, thus according to [60] they adopt the default MSS of 536, or their MSS declaration was unobserved.

Finally, a count of the number of TCP flows utilizing the push flag and the urgent flag, as well as flows utilizing the don't fragment flag, was made. We found that the push flag is actually quite common, appearing at least once in 62.59% of the observed TCP flows and in 20.75% of observed TCP packets. On the other hand, we found the urgent flag to be extremely rare, only appearing in less than 0.01% of TCP flows. The don't fragment flag also appeared quite often, in 33.91% of observed flows. The high occurrence of the don't fragment flag is most likely an attempt to avoid the performance penalties of fragmentation [37, 66].

5.5 Use of the DNS Infrastructure

As can be seen in Table 2 and Table 4, DNS traffic makes up a significant portion of the traffic observed by NETI@home. Furthermore, DNS is a critical infrastructure to the normal operation of the Internet. Without DNS, users would be unable to convert the well-known DNS names into their lesser-known corresponding IP addresses. One critical element to the operation of the DNS infrastructure are the 13 root DNS servers. Should these servers be unavailable, either due to intentional attack or misuse and mismanagement, the operation of the DNS infrastructure and the Internet as a whole would be at serious risk.

Utilizing the unique end-user perspective of NETI@home, we noticed that end-hosts frequently (approximately 6% of TCP DNS flows) contact the root DNS servers directly, rather than through their local DNS servers. This behavior is considered to be a bad practice [40] and is alarming as it can place undue stress on the root DNS servers that are

so critical to the current use of the Internet. Unfortunately, NETI@home is only able to report when users contact the root DNS servers when their privacy is set to low. We also find that approximately 26% of end-users with low privacy TCP DNS flows contact the root DNS servers directly. However, upon further investigation, all of these users are located within the same domain, a sample that cannot be considered representative. Furthermore, of the low privacy users with UDP DNS flows, we find that a variety of operating systems are used, including Linux and Windows. Thus, it appears the problem of directly contacting the root DNS servers cannot be attributed to users of a single operating system.

CHAPTER 6

END-USER BEHAVIOR

One of the strengths of NETI@home is its ability to observe network activity from the perspective of "typical" end-users. This perspective gives us insight into not only what the end-user sees, but also how the end-user behaves. This perspective is used in this chapter to understand end-user behavior in a network-independent fashion. It is difficult to conduct such studies without an infrastructure such as NETI@home in place. Other approaches that could be used would be to measure from a midpoint in the network, from the server side, or from a gateway such as those at the edges of campus networks. However, each of these approaches has problems that are addressed by NETI@home. From the midpoint of the network, it is difficult to be certain of one's sample size. Also, from this perspective end-to-end measurements are lost. From the server side, many end-to-end measurements can be made. However, the server side perspective depends on the popularity and audience of that particular server. Further, it is difficult to increase the sample size to many servers, as many server administrators would be reluctant to give up such sensitive information. Finally, measurements made from university campus gateways are frequently studied. This perspective also has its drawbacks. For instance, campus network users cannot be considered "typical" end-users. Also, from the perspective of the gateway, information about local area network traffic, caching, and application proxying is also lost. Thus, NETI@home is in the unique position to provide detailed analyses of end-user habits and behavior.

The work presented in [71] and extended in [70] presents our attempts to analyze network-independent user behavior using the NETI@home dataset. There, we developed network-independent traffic models for network user behavior.

6.1 Methodology

Two categories of models were created in this study. The first is specific to a TCP or UDP port, that is, we create a model of client behavior for a given TCP or UDP port. We use the model created for TCP port 80, the most common port used by World Wide Web servers, as an example, as it was studied in [44]. However, for variety we also present models for several other TCP and UDP ports. The second category of model created is an aggregate of all port-specific models. This model can be likened to a TCP or UDP client model. Such a model may prove useful for studies that are more generic and are not attempting to study a particular type of network traffic. All of these models incorporate empirical distributions directly interpreted from the NETI@home dataset.

The dataset used in this chapter consists of the same NETI@home data analyzed in Chapter 5. However, in this work, we focus primarily on the TCP and UDP flows.

Several characteristics of TCP and UDP flows were chosen to reflect network-independent behavior and to wholly represent network client behavior.

The first two aspects we model are empirical distributions of *bytes sent* and *bytes received*. These values are based only on the payload of the packets and thus do not represent the sizes of the TCP or UDP headers and their underlying headers or TCP's flow control and congestion control algorithms, merely transferred application information. This allows our models to be used in simulations where variations of TCP or UDP are employed.

The next aspect modeled is *user think time*. User think time is the term we use for the amount of time a client waits before initiating another flow. For this aspect, we developed two empirical distributions. One distribution describes the user think time when consecutively accessing a specific destination and the other describes the user think time when contacting a new destination.

Another aspect modeled is *consecutive contacts*. Consecutive contacts is the term we use for the probability that a client will choose to initiate another flow with the last destination contacted, or the client will choose to initiate a flow with a new destination. For this aspect, we developed a single empirical distribution.

Finally, the last aspect modeled is *contact selection*. Contact selection is the term we

use for the frequency distribution of contacting specific destinations. This distribution can be thought of as modeling the popularity of a destination. For this aspect, we developed a single empirical distribution.

One other aspect that we believe to be worth modeling is related to *idle time*. For applications such as World Wide Web transfers, this aspect has little meaning, as Web pages are simply requested and served. However, for interactive applications such as SSH or telnet, there are periods of time, *during* the flow, when there is no data transferred. However, using the NETI@home data, it is difficult to differentiate between network-dependent flow time and network-independent flow time. We are aware of work [28, 29] that attempts to capture this behavior and are considering implementing a similar technique into the NETI@home client software so that future models can incorporate this aspect of user behavior.

6.2 Experimental Results

From the analysis of the NETI@home dataset described previously, we were able to generate a set of empirical distributions for each component of our models. To download the complete set of distributions and for any updates to these distributions please visit `http://neti.gatech.edu/research/user.html`.

6.2.1 Bytes Sent

The number of bytes sent varies depending on the port modeled. However, upon investigation of each modeled port, our findings seem intuitive. The cumulative distribution functions of bytes sent for various ports, as well as an aggregate of all TCP ports, are shown in Figure 15.

Figure 15(a) depicts the cumulative distribution function of bytes sent for TCP port 80. Compared with previous studies [44], these results contain many more flows with zero bytes sent. However, upon investigation it does not appear that these results are due to a single NETI@home user or are anomalous. The zero-bytes-sent flows typically represent flows in which the connection failed during the TCP three-way handshake. Although these flows do not generate much network traffic (usually no more than three packets), they are significant in terms of numbers of flows and most likely influence a user's behavior.

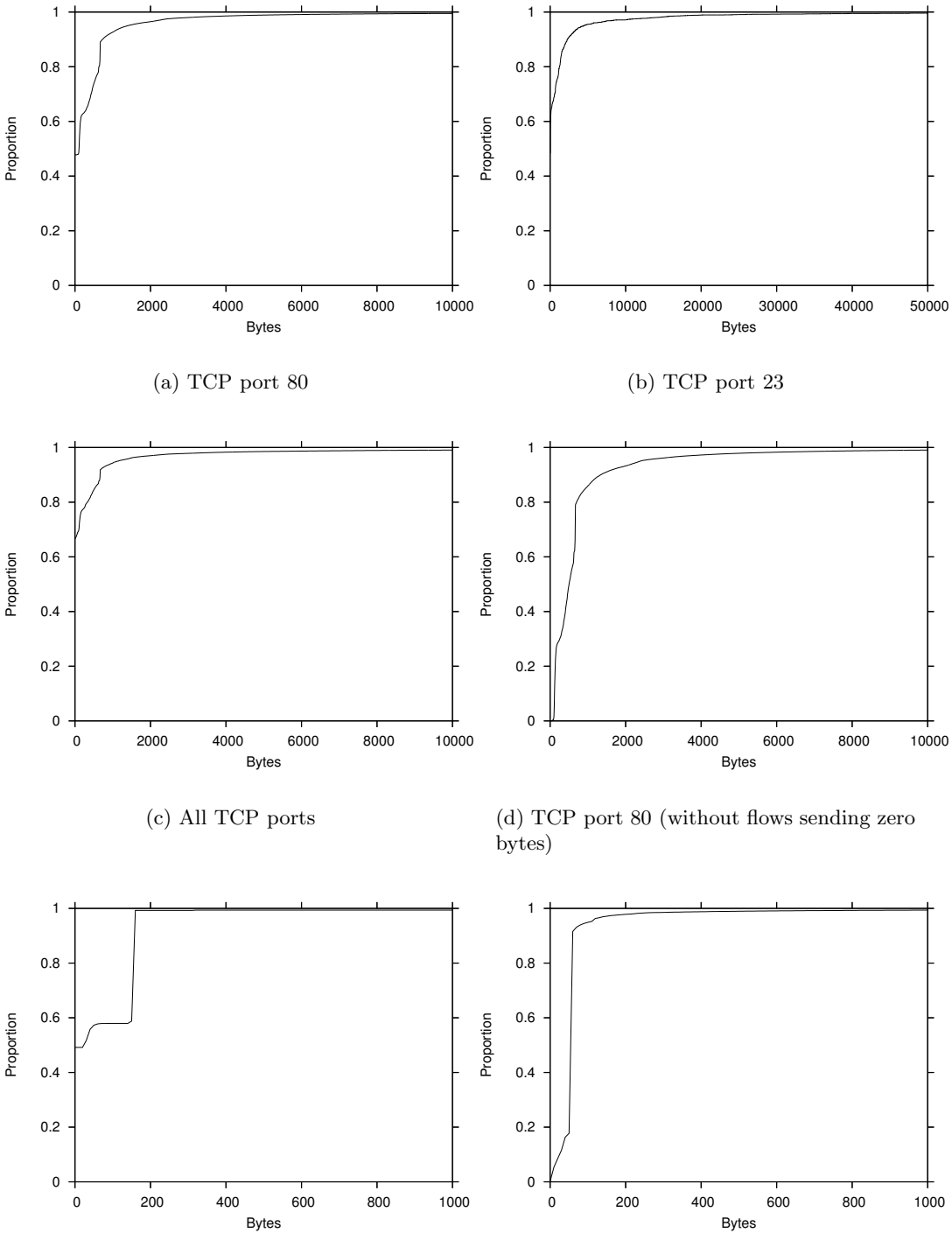
As can be seen in Figure 15(a), approximately 40% of the flows to TCP port 80 send little or no data. There are several possible causes for the large number of flows sending little or no data. First, many of these flows are failed connection attempts. Many NETI@home users utilize less reliable network connections than the campus network of [44], such as dial-up or wireless. Also, some of these flows may be to blocked sites. Many browsers and third-party software block advertisements and some organizations restrict the viewing of certain websites. Finally, a handful of NETI@home users periodically scan hosts on the Internet [27]. Considering that these users know that their network connections are monitored, it is unlikely that this scanning is intentional and may be the result of a virus or worm. While these results could be considered anomalous, we believe that this does indeed represent typical end-user behavior as seen on the Internet. Almost all remaining flows send no more than 10 KB of data to the server. Figure 15(d) shows the cumulative distribution function of bytes sent for TCP port 80, with the flows sending no data removed.

Other ports that are modeled include TCP port 23 (Figure 15(b)), TCP port 53 (Figure 15(e)), and UDP port 53 (Figure 15(f)). Each of these ports exhibit slightly different behavior than TCP port 80. TCP port 23 is the port commonly used for telnet connections. These interactive flows vary wildly, from sending very little data to a handful of flows that send large amounts of data. TCP and UDP ports 53 are the ports commonly used for DNS. Flows to these ports send very little data, as each flow typically represents a single DNS query, as can be seen in Figure 15(e) and Figure 15(f). Finally, Figure 15(c) depicts the cumulative distribution function of bytes sent for all TCP ports. This distribution shows that a large percentage (approximately 70%) of TCP flows send little or no data.

6.2.2 Bytes Received

The number of bytes received by the client is also dependent on the port modeled. The cumulative distribution functions of bytes received for various ports, as well as an aggregate of all TCP ports, are shown in Figure 16.

Figure 16(a) depicts the cumulative distribution function of bytes received for TCP port 80. Compared with [44], we also find that there are many more flows with zero bytes



(a) TCP port 80

(b) TCP port 23

(c) All TCP ports

(d) TCP port 80 (without flows sending zero bytes)

(e) TCP port 53

(f) UDP port 53

Figure 15: CDF of bytes sent.

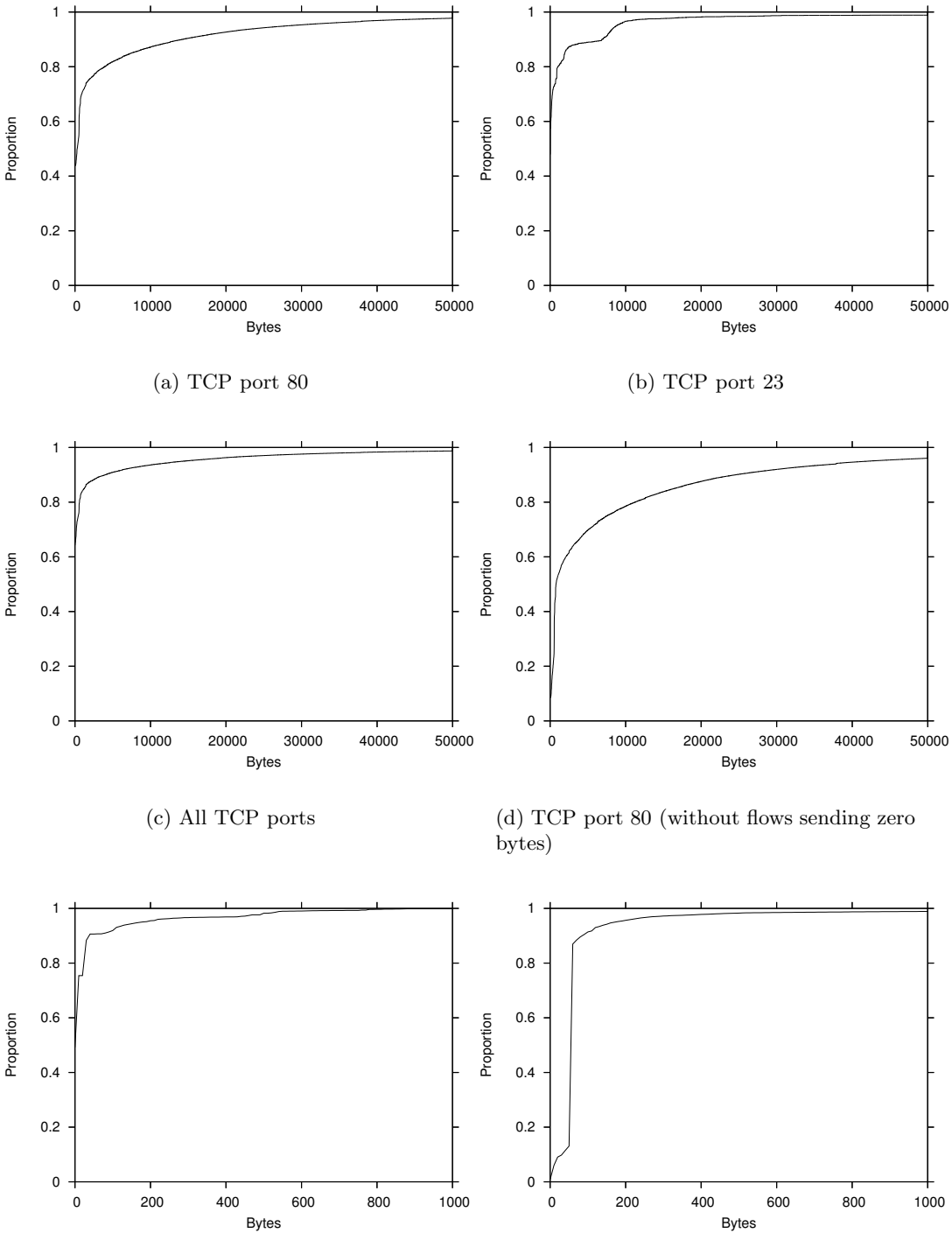
received. As with our findings for bytes sent, this is most likely due to failed connection attempts.

The distribution of bytes received for TCP port 80 has a much longer tail than that for the bytes sent. Approximately 40% of flows with a remote TCP port of 80 receive little or no data. However, more than 10% of these flows receive greater than 10 KB of data. Figure 16(d) shows the cumulative distribution function of bytes received for TCP port 80, with the flows sending no data removed. The removal of these flows appears to remove many of the flows that received no data, as one would expect.

Other ports that are modeled include TCP port 23 (Figure 16(b)), TCP port 53 (Figure 16(e)), and UDP port 53 (Figure 16(f)). These ports also exhibit slight differences from TCP port 80, as with the distributions for bytes sent. The distribution of bytes received for TCP port 23 shows that many telnet connections receive little data. However, as with TCP port 80, there are a few connections that receive a large amount of data. The distributions of bytes received for TCP and UDP ports 53 show a tendency toward very little data being received. This is due to the fact that most of the flows are responses to single DNS queries, which agrees with our findings for the complementary bytes sent distributions. Last, Figure 16(c) shows the cumulative distribution function of bytes received for all TCP ports. As with the distribution for bytes sent, the majority of TCP flows receive very little data.

6.2.3 User Think Time

The cumulative distribution function for user think time to the same destination is given in Figure 17 and to differing destinations is given in Figure 18 for various ports as well as all TCP traffic aggregated. These findings show a tendency toward shorter user think times than was found in [44] for TCP port 80. We can think of several reasons for this shortened user think time. First, the World Wide Web has become much more popular since [44] was published. Also, it is likely that NETI@home captures data from users who are active more often than it does for inactive users, as many users would simply turn off their machines while not using them, thus disabling NETI@home's monitoring. This would artificially inflate our numbers to show users who appear to be more active and is a source



(a) TCP port 80

(b) TCP port 23

(c) All TCP ports

(d) TCP port 80 (without flows sending zero bytes)

(e) TCP port 53

(f) UDP port 53

Figure 16: CDF of bytes received.

of bias.

We chose to model the user think time to the same destination separately from the user think time to a different destination. Figures 17(a) and 18(a) appear to be similar, however. We believe that it is still appropriate to model these think times separately, as these distributions can differ greatly for other TCP or UDP ports, as is shown in Figures 17(b) and 18(b). These figures show the distributions for think times for TCP port 23, the port commonly used for telnet.

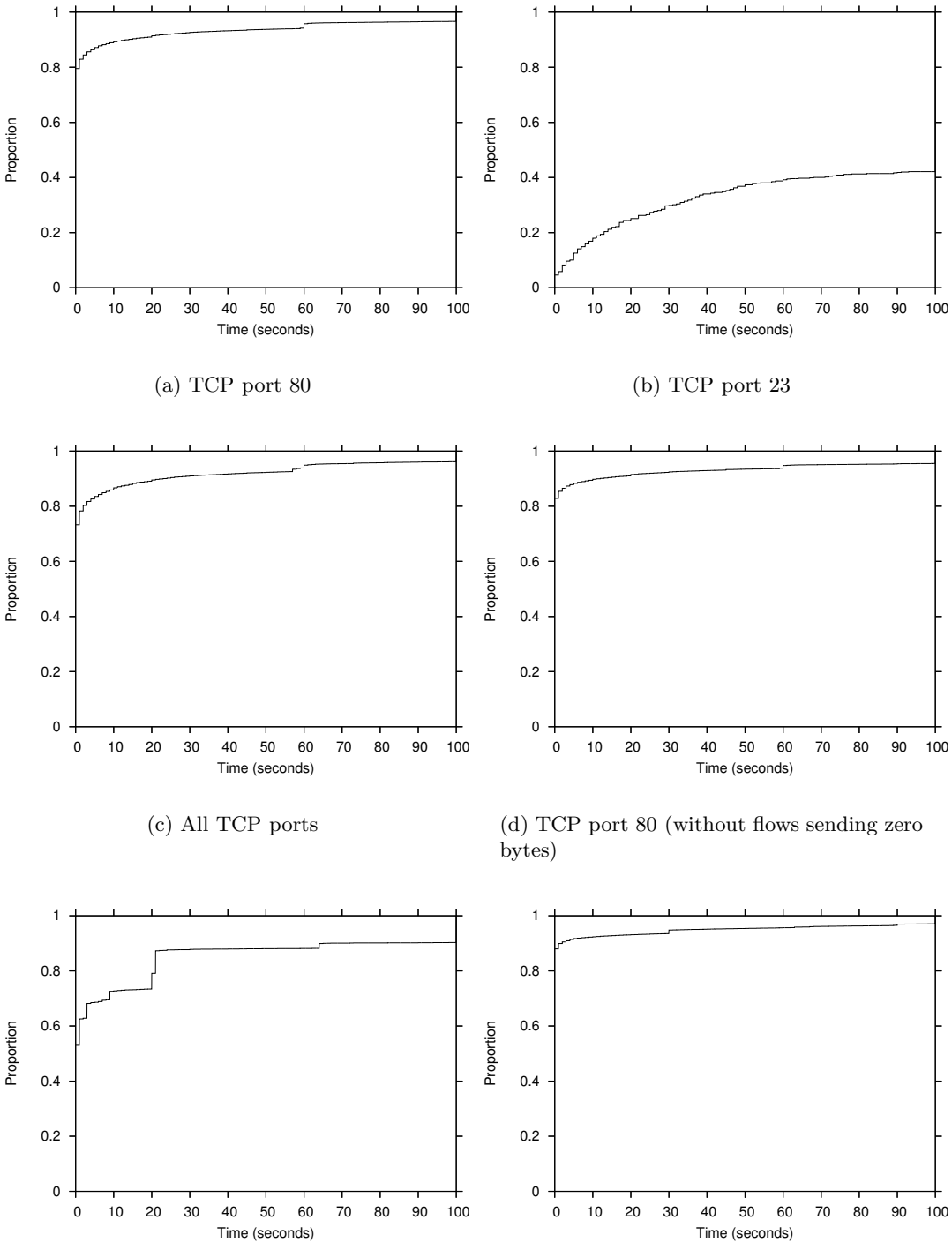
For connections to TCP port 80, the majority of user think times tends to be less than 1 second. However, for connections to TCP port 23 (telnet), the user think times have a much heavier tail, with only approximately 40% of flows having think times less than 100 seconds.

In an effort to determine whether the large number of failed connections on TCP port 80 contributes to the shorter think times observed, we show in Figure 17(d) and Figure 18(d) the think times to the same destination and to differing destinations for TCP port 80 without these failed connections. As can be seen in the figures, the absence of these failed connections does not seem to have a significant impact on the think time distributions. Thus, the shortened think times when compared to [44] must be due to changes in browsing behavior.

Finally, the stair-step patterns of Figure 17(e) and Figure 18(e), which represent TCP port 53 (DNS), are rather interesting. These stair-stepping patterns are most likely due to a timeout in queries, with local timeouts at approximately 20 seconds and secondary timeouts at approximately 60 seconds. Their UDP counterparts on port 53 (Figure 17(f) and Figure 18(f)) also show timing differences between contacting the same server and differing servers, although not as pronounced.

6.2.4 Consecutive Contacts

The cumulative distribution functions of consecutive contacts for various ports, as well as all TCP ports aggregated, are shown in Figure 19. Most of these functions are quite similar, with a few notable exceptions.



(a) TCP port 80

(b) TCP port 23

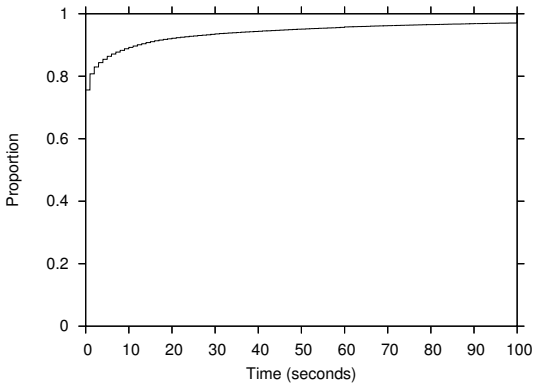
(c) All TCP ports

(d) TCP port 80 (without flows sending zero bytes)

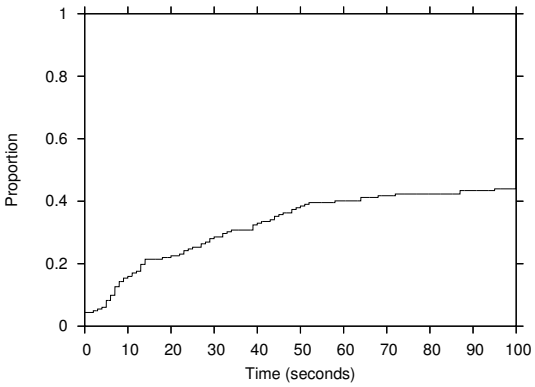
(e) TCP port 53

(f) UDP port 53

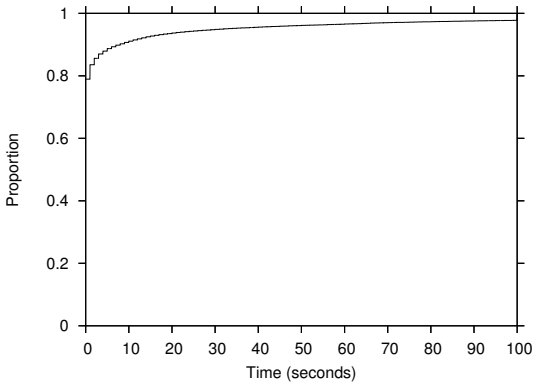
Figure 17: CDF of user think time to same IPs.



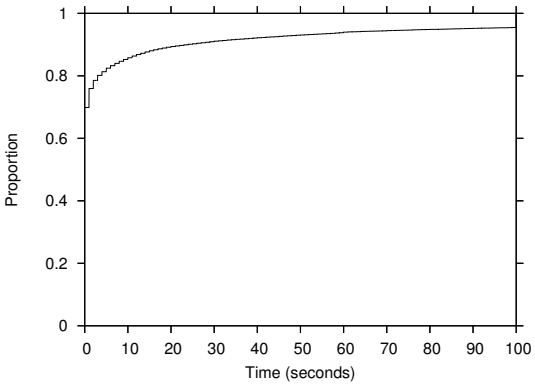
(a) TCP port 80



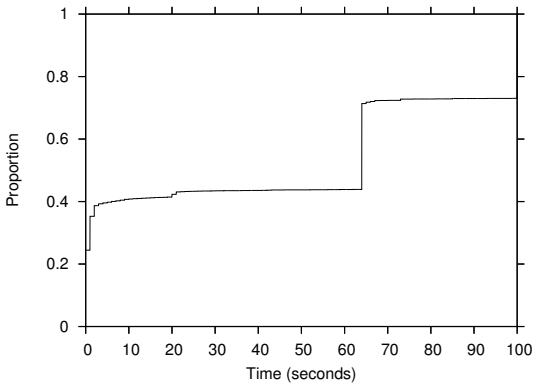
(b) TCP port 23



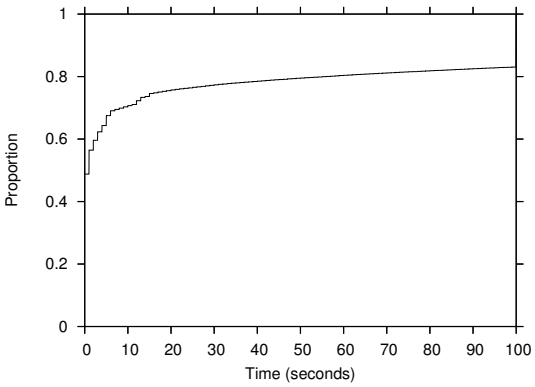
(c) All TCP ports



(d) TCP port 80 (without flows sending zero bytes)



(e) TCP port 53



(f) UDP port 53

Figure 18: CDF of user think time to differing IPs.

In Figure 19(a), we present the cumulative distribution function of consecutive contacts for TCP port 80. These results also show a tendency toward a lower number of consecutive contacts than was found in [44]. As can be seen from Figure 19(d), this does not appear to be a result of failed connection attempts. Possible reasons for this lower number of consecutive contacts than was found in [44] are that websites tend to incorporate more objects from other sites such as advertisements as well as the prevalence of caching sites for many of the popular destinations.

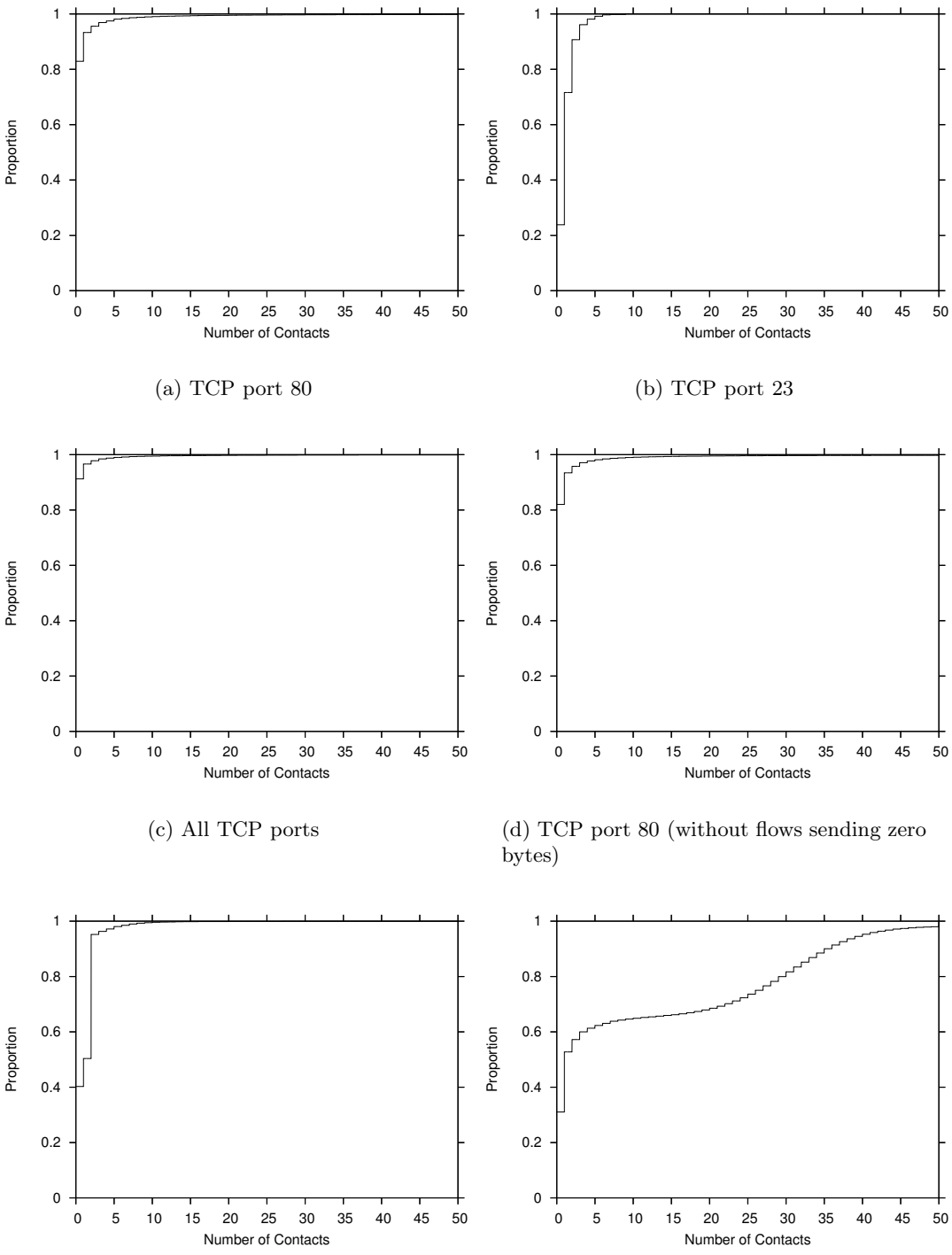
Approximately 80% of the flows to TCP port 80 are not consecutive, that is, the destination is contacted only once in a row. Further, over 99% of visits to a specific destination on TCP port 80 lasted for 10 or fewer flows in a row. Therefore, it appears that users tend to switch Web destinations fairly often, as was noted in [44].

Of note are Figure 19(e) and Figure 19(f), which show much more consecutive contacts. This appears to be intuitive as most Internet connections would query their local DNS servers more often than any other.

6.2.5 Contact Selection

Unlike [44], which used a Zipf distribution, we were able to construct a cumulative distribution function of contact selection because of the wide sampling offered by the NETI@home dataset. Figure 20 presents the cumulative distribution functions of contact selection for various ports as well as all TCP ports aggregated. One possible source of inaccuracy for this aspect is that we are unable to determine if a specific destination uses multiple IP addresses, thus reducing the frequency of selection a given contact may appear to have.

As can be seen in Figure 20(a), for TCP port 80 servers the distribution of the overall number of visits by NETI@home users is quite varied and has a heavy tail. Many servers are only visited a handful of times; however, many other servers tend to be contacted quite often, with some servers receiving millions of visits over the year studied. This trend appears to continue for all ports, with Figure 20(e) and Figure 20(f) showing very wide distributions, which is intuitive considering most DNS servers are local, but there are a handful of root DNS servers.



(a) TCP port 80

(b) TCP port 23

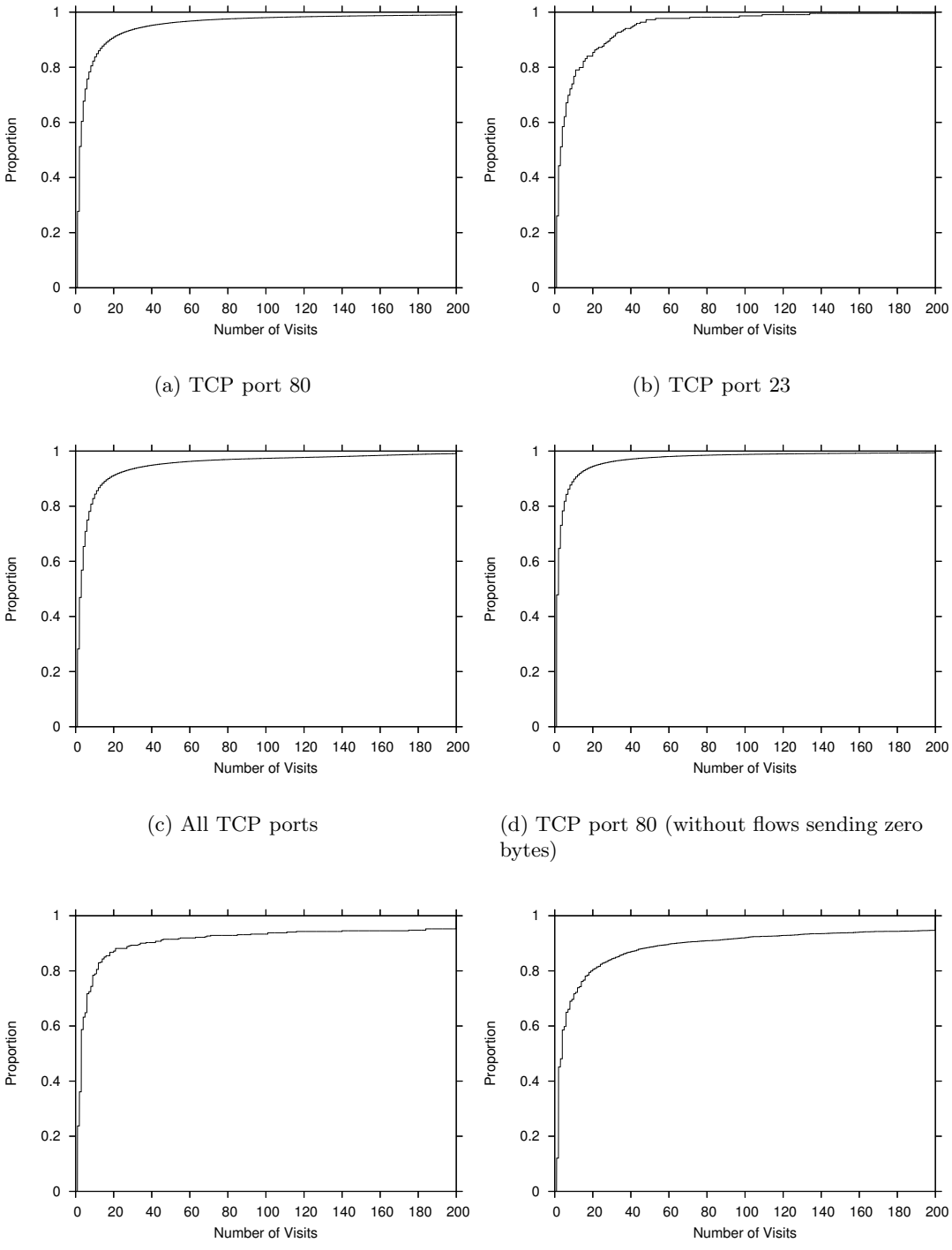
(c) All TCP ports

(d) TCP port 80 (without flows sending zero bytes)

(e) TCP port 53

(f) UDP port 53

Figure 19: CDF of number of times an IP is contacted consecutively.



(a) TCP port 80

(b) TCP port 23

(c) All TCP ports

(d) TCP port 80 (without flows sending zero bytes)

(e) TCP port 53

(f) UDP port 53

Figure 20: CDF of relative frequency of server visits over a one year period.

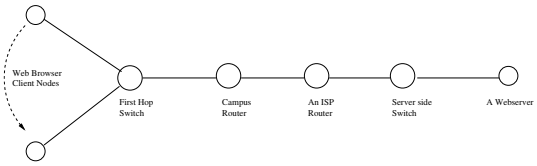


Figure 21: Network topology used for testing traffic models in simulation.

6.3 Simulation Results

To judge the usefulness of our models, we have incorporated the above derived TCP port 80 traffic models into the GTNetS environment [64]. The GTNetS environment already has some HTTP traffic models, as described in [44]. We incorporated the models derived from the analysis of the NETI@home datasets into GTNetS. We consider this approach to be a better one for traffic generation in network simulations because NETI@home datasets are more current and continue to be so [72]. An analysis program generates these models automatically from the NETI@home datasets. The traffic distribution models can then be easily used by the application layer models that drive a network simulation. In our simulation experiments, we have concentrated on the World Wide Web traffic and the HTTP models. Our implementation samples the empirical distributions to determine the particular values used at a given time. This seems a logical choice since any single distribution doesn't seem to fit the complete dataset verifiably. We model the behavior of a Web browser in GTNetS that sends an HTTP request to a designated Web server asking it to send a certain length of data that constitutes the response. When the simulation starts, the browser application chooses a server randomly from a list of target servers. It then chooses a *response size* that it wants to obtain from the Web server from the CDF that describes the *bytes received*. The size of the HTTP request packet is chosen from the *bytes sent* CDF plot. It may request one or more objects within the same TCP connection. Once the Web browser application has received the appropriate response, it proceeds to select a different server or the same server for its next request and waits for an amount of time. This amount of time, which is obtained from the CDF that describes the *user think time*, depends on whether the same server is chosen or a different server is chosen.

The network topology for simulations is obtained from [12]. It consists of a large set

Table 8: Variation in average and maximum response times when using HTTP traffic model presented in this chapter (with flows sending zero bytes)

Number of browsers	Average response time (seconds)	Maximum response time (seconds)
10	0.310057	0.539721
25	0.311265	0.64728
50	0.311925	0.688772
75	0.311463	0.707714
100	0.311327	0.725715
125	0.311567	0.736565
150	0.311746	0.750559

of Web browsers connected via a series of three routers to a Web server, as shown in Figure 21. We have chosen this to be our baseline topology because we have earlier simulation experiments conducted using the models and datasets proposed in [12].

The simulation experiment is run using three HTTP traffic models. One of the traffic models is obtained from the datasets suggested in [44] and [12]. The remaining traffic models are those obtained from the NETI@home datasets. Intuitively, empirical traffic models should be more representative of a realistic dataset than statistical traffic generators, although the former cannot be subjected to extrapolations. All the measurements are the averages of 1,000 runs of a simulation at a given data point.

Table 8 shows the average and maximum response times for a given number of Web clients when they request data from a Web server using the traffic models presented in this chapter. Table 9 shows the average and maximum response times for the same number of Web browsers when they request data from a Web server using the same traffic models with the flows sending no data removed. Finally, Table 10 shows the average and maximum response times for the same number of Web browsers when they request data from a Web server using the traffic models presented in [44].

It can be seen from the results in Table 8, Table 9, and Table 10 that the maximum response time for the HTTP traffic model presented in [44] is substantially longer than the models that are derived from the NETI@home dataset. A careful observation of the

Table 9: Variation in average and maximum response times when using HTTP traffic model presented in this chapter (without flows sending zero bytes)

Number of browsers	Average response time (seconds)	Maximum response time (seconds)
10	0.34648	0.595165
25	0.346575	0.693681
50	0.346592	0.750871
75	0.346364	0.773511
100	0.346315	0.803677
125	0.346715	0.81649
150	0.346481	0.82985

Table 10: Variation in average and maximum response times when using HTTP traffic model presented in [44]

Number of browsers	Average response time (seconds)	Maximum response time (seconds)
10	0.354478	1.25531
25	0.35087	2.12011
50	0.351921	3.52208
75	0.351714	4.39087
100	0.35378	5.53197
125	0.352605	6.48086
150	0.351689	6.84786

cumulative distribution functions of the datasets shows that the NETI@home data has a larger proportion of flow sizes that are very small, both with and without the inclusion of failed connections. This results in lower load on the Web server and consequently lower latencies. This is evident in the lower average and maximum response times as the traffic increases. On the other hand, the traffic model presented in [44] has fewer flow sizes that are very small. This results in a larger load on the server and on the network as the number of Web browsers increases. When the number of Web browsers is fairly small, the difference is not appreciable because the flow size does not influence the network.

The code used for these simulations, as well as the empirical models, are available in the latest official distribution of the GTNetS environment at `http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/`.

CHAPTER 7

DATA DISSEMINATION

This chapter describes our work to further anonymize the NETI@home dataset to prepare it for public release. We then go on to discuss the public release of the dataset and the tools used to conduct the analyses in this dissertation, so that researchers around the world can use the dataset as a basis for studies.

7.1 Anonymization Woes

Despite the previously-mentioned user-selected privacy settings, we have desired to further anonymize the NETI@home dataset, primarily to protect the privacy of our users. However, this has been a major roadblock for the past few years. Initially, we were concerned with attacks that could be performed by a malicious NETI@home user to reverse-map any anonymization done. For instance, imagine that a malicious NETI@home user wanted to reverse-map all anonymized IP addresses. In this scenario, all this user must do is scan the entire IP address range in some fashion, and then report these flows to the NETI@home server. Once we received the data, we would anonymize it in some fashion and then make it publicly available. This malicious NETI@home user could then compare their local copy of the scan to our anonymized copy and reverse-map the entire IP address space. Our initial solution to this problem was to use separate keys to anonymize the remote IP addresses and local IP addresses, thus at the least protecting the identities of the NETI@home users. However, we quickly determined that although unanonymizing the remote IP address space would be trivial, it would also be trivial to unanonymize the local IP address space, as a malicious user could simply spoof their local IP address in reports to the NETI@home server.

We then became aware of the work in [6], which could easily be extended to NETI@home. The work in [6] showed that it is possible to map the sensors used by the Internet Storm

Center [81] with a limited amount of time based on unique packets sent to each viable address in the IP address space. Such an attack would easily work on the NETI@home infrastructure as well. So, it seems to us that the only viable option to protect user privacy is to zero out all IP addresses in the publicly available NETI@home dataset. While unfortunate and a great loss to the research community, such a decision follows from our commitment to user privacy. However, we also realized that a mapping attack such as the one presented in [6] would allow an attacker to determine which IP addresses participate in the NETI@home project. At first, this seemed to be an insurmountable problem. We were presented with a doomsday view of not only releasing the NETI@home dataset, but of making any network traces of any practical utility publicly available. This may be true for projects such as honeynets [61, 78] or CAIDA's Internet Telescope [49], but it may actually work in the favor of NETI@home users. For instance, imagine that the IP addresses participating in the NETI@home project were discovered and "blacklisted" from the perspective of malicious members of the Internet community. Although we would lose the ability to track sophisticated attacks, we would gain a new incentive for hosts to participate in the NETI@home project. By virtue of participation, the NETI@home users would theoretically be protected from some of the sophisticated attacks present on the Internet as a whole. Although the publicly available dataset would unfortunately not contain any IP addresses, we would also gain the practical inability to determine the remote endpoint of individual flows.

Thus, our current solution is to publicly release the NETI@home dataset in its entirety without any IP addresses. To trusted researchers, we will offer the dataset with "anonymized" IP addresses on a case-by-case basis. Each trusted user would also have their custom dataset anonymized with a different key, to minimize the likelihood of malicious collaboration. To implement this anonymization, we use the Crypto-PAn library [21]. Further, separate keys are used for the local IP address space, the remote IP address space, and the external IP address space. However, to increase the research value of the dataset, we do not anonymize IP addresses within the private IP address space [62].

Although we have reached a conclusion on how to proceed with the dissemination of the NETI@home dataset, we have also reached a grim realization on the future of collaboration

among the Internet community. It appears from the current research that there is a trade-off between security and utility as mentioned in [74] when it comes to publicly releasing data. Our fear is that this realization will lead to a drop in the collaboration of members of the Internet community, from a point that is already low, especially following such high-profile embarrassments like the recent America Online search data fiasco [45]. We believe that this problem is in desperate need of a solution and is a definite area of future work. Further, our intuition is that this problem has no real solution. Since Internet flows, especially flows between monitored users and malicious users, are not owned by one end but are shared by at least both end-hosts, they can always be used as a covert channel of communication. Thus, to place information about such flows in the view of the public exposes information about the end-hosts somewhat equal to the utility of the posted information. From this dismal view it may be that rather than watch the collapse of the sharing of Internet data we may need to do the opposite. That is, everyone may need to share their data, with those not sharing running the risk of being the target of the most sophisticated attacks. This also further reiterates the need for secure communication channels between all end-hosts, something that researchers have realized from quite some time. We leave the response to future work and to the Internet community as a whole.

7.2 Data Dissemination

It has been noted that the area of network measurements suffers from the unwillingness of researchers to share traces among themselves, usually because of privacy concerns. Such reluctance hampers this field, as results are often not reproducible and conclusions, while sometimes questioned, often cannot be authenticated [57, 67]. To address this need we plan to set up a collaboration environment around NETI@home.

As previously mentioned, our current approach to distributing the NETI@home dataset is twofold. For public release, the dataset has all IP addresses removed. To trusted members of the research community, we will provide the dataset with IP addresses anonymized in a prefix-preserving fashion upon request. The public sharing of the NETI@home dataset will provide researchers around the world a basis for network measurement studies.

The public release of the NETI@home dataset is located at `http://neti.gatech.edu/research/data`. Each day's worth of data will be available. Further, we will periodically send out concise reports based on our own data analyses.

To request an anonymized version of the dataset, please email `neti@ece.gatech.edu` and specify your background and the range of dates for the data you would like.

To facilitate further research, we are making all of our own analysis tools available under an open-source license at `http://neti.gatech.edu/research/tools` so that other researchers can use and improve them. We encourage researchers to use this framework to conduct their own analyses and then provide their experimental code to encourage community-based analysis and discussion. We believe such a community-based approach will lead to easily validated experiments as well as the ability to expand upon previous experiments, fulfilling the need for reproducibility and authentication of results.

Finally, we plan to document our dataset and tools in the DatCat Internet Measurement Data Catalog [67]. DatCat is advocated by CAIDA to be a starting point for data sharing and allows users to provide metadata about various datasets and experiments. Our contribution to DatCat will strengthen the availability of the NETI@home dataset and tools.

CHAPTER 8

CONCLUSIONS

This dissertation has introduced a distributed network infrastructure that collects end-to-end network performance measurements from a variety of end-users around the world. We have gone on to use this infrastructure to conduct analyses related to network security, network behavior, and end-user behavior. Below, we present our conclusions and some areas of future work.

8.1 Conclusions

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 the Georgia Institute of Technology in a scalable manner, where it is made publicly available to aid researchers in the on-going quest to improve the global Internet.

We used a number of methods to analyze network flows over time for NETI@home data and honeynet data. In both datasets, the majority of the TCP flows were failed connections. In the honeynet dataset, these flows were malicious in nature. The NETI@home dataset has a smaller percentage of TCP flows that were failed connections, and these flows were

not necessarily malicious in nature.

The majority of the traffic seen in the honeynet dataset consists of port scans and worms. We observed that the outbreak of a new worm will linger on for more than a year after the release date. Similar patterns were observed in the NETI@home data, although it is difficult to distinguish between malicious and legitimate traffic.

We also found that port scanning was seen by NETI@home users and honeynet machines regularly. By using our technique of a TCP port histogram, we were able to observe a port scan of NETI@home users that slowly scanned the ports over the course of several days. This particular port scan was found to be non-malicious in nature, as it was a security scan by the Georgia Institute of Technology on machines within their IP address space. However, the majority of port scans were most likely malicious in nature and some of the malicious port scanning patterns observed in the honeynet dataset were also observed in the NETI@home dataset.

We found that both datasets showed similar flow distributions across the IP address space. In the NETI@home dataset, however, a small number of users were scanning most of the IP address space in a random fashion on a TCP port that is the target of recent worms. This shows that these contacts were likely malicious in nature. Finally, for both datasets, the source of malicious and legitimate traffic comes from across the entire allocated IP address space. We did not observe significant malicious traffic or legitimate traffic coming from the unallocated IP address space. One important conclusion to draw from these observations is that machines contacting the unallocated regions of the IP address space should be considered suspicious.

In Chapter 5, we focused on several aspects of networking performance and behavior. We found that average hop counts vary depending on the transport layer protocol being used. Further, we have found that hop counts infrequently vary over the lifetime of a flow, demonstrating that there is routing stability over the lifetime of most network flows.

Next, we observed the frequency of network address translation and its uses in IP address conversion. Further, we investigated the prevalence of hosts utilizing the private IP address range. We found that the use of NAT is quite common, as is the use of the private IP

address range.

The adoption and use of several protocol options and flags was also investigated. We found that the use of TCP SACK is common. When TCP flows are unable to utilize TCP SACK, it is often due to one end-host of the connection, with the other host having TCP SACK capability, as TCP SACK requires capability in both end-hosts to function. In contrast, we found that the use of TCP window scaling is not common. The lack of window scaling support is alarming as the transfer rates of flows may be limited by the maximum TCP advertised window size of 65,535 bytes. To conclude, the implementation of some protocol flags and options is drastically behind the specification of these flags and options.

We also found that end-hosts frequently contact the root DNS servers directly, rather than through their local DNS servers. Such behavior is not due to one particular host or operating system. This behavior could have severe effects on the performance of these root servers, which are a critical component to the current operation of the Internet.

In Chapter 6, we presented empirical models of end-user network traffic. There are two general forms of these models, one form is port-specific for a given TCP or UDP port. The second form is a generic model for TCP or UDP traffic. These models consist of network-independent distributions for the number of bytes sent, the number of bytes received, the user think time to the same destination, the user think time to a different destination, the number of times a destination will be contacted consecutively, and the popularity of specific destinations. These distributions differ from previous findings. Our data shows many more flows sending and receiving zero bytes as well as decreased think times.

The distributions derived are based on the NETI@home dataset and are meant to represent a heterogeneous sampling of network users. Such a heterogeneous sampling of users from differing network and geographical locations provides more accurate models for simulations. As the NETI@home project is ongoing for the foreseeable future, we plan to continuously update the models. For these updates and to download the complete distributions please visit `http://neti.gatech.edu/research/user.html`.

Further, we have implemented these models in the GTNetS simulation environment. In this simulation environment we tested the affect of network traffic on a Web server. These

results were then compared to the results from previous models and we found that the maximum response times have decreased. This decrease is a result of the larger proportion of flow sizes that are small. The models and code used are available in the latest distribution of the GTNetS environment at `http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/`.

Finally, in Chapter 7, our techniques for anonymizing the NETI@home data, as well as our plans for distributing this valuable data were discussed. We have found that the sharing of network measurement data is threatened, as a result of the tradeoffs between utility and security. Thus, to protect our users' privacy with a high degree of certainty, all IP address information in the NETI@home data is removed in the public release. To trusted members of the research community, these addresses will be anonymized in a prefix-preserving fashion. Further, we will also release our analysis tools to foster collaboration among Internet researchers and to aid in the validation of results. Our hope is that the NETI@home data will aid in the progress made possible by research.

8.2 Future Work

In this dissertation, we have just begun to scratch the surface of the NETI@home dataset. This dataset provides a unique perspective into end-user and networking behavior. Further studies of this dataset are planned and the NETI@home project will continue to collect data from end-user volunteers for the foreseeable future.

In the future, many more features could 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, 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 would either not be recorded, would be trimmed, or it would 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 could be provided. When a user clicks a specific host on the map, another window would appear and a graphical traceroute would 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 would also be sent to the server at the Georgia Institute of Technology to add to the amount of measurements that are collected. Although the DIMES Project [19] and Traceroute@home [84] also collect traceroute measurements from end-user volunteers on the Internet, our approach would be a passive one as opposed to their active mapping approach.

There are a number of future directions to research specific to network security. A formal statistical correlation between the honeynet data and the NETI@home data could be performed, to draw more definitive conclusions. There are numerous other network statistics that can be compared such as TTL values, window sizes, checksum errors, and so forth. The analysis of these areas of research are left to future work.

Further insight is needed to understand the anomaly observed in initial TTL values. We propose that this varying of the initial TTL value may be a result of user modification, application modification, or load-balancing by servers. We are currently attempting to contact the administrators of several Web servers that exhibit this behavior for more information.

Long term studies of the adoption of the previously mentioned protocol flags and options are an area of future work as there is great concern that many of these options need to be implemented by the authors of network protocol stacks and then utilized. Further, the prevalence of NAT and its impact on the provisioning of IP addresses, and on the behavior of networking in general, deserves greater attention.

Several enhancements to our end-user modeling technique can be made and are areas of future work. First, it would be useful to model idle times within a flow. As previously mentioned, certain applications have periods of time where the connection is idle, as in

interactive applications. Another enhancement to our model would be to determine if there is any correlation between the different aspects of our model. For example, in certain applications the number of bytes sent and the number of bytes received may be highly correlated. If so, these aspects should most likely be treated as bivariate data. Several enhancements could also be made to our consecutive contacts and contact selection components. It is intuitive that once a destination is visited and then left, that the original destination has a higher likelihood of being visited again. Thus, a model with memory, such as a Markov model, would be useful. Such a model may also incorporate zero byte flows. That is, if a connection fails, the likelihood of that connection's destination of being visited again may change. Further, our model could be extended to other protocols beyond TCP and UDP. Currently, NETI@home collects flow summary statistics for TCP, UDP, ICMP, and IGMP, so ICMP and IGMP models could easily be derived.

The models presented in this dissertation solely focus on network-independent characteristics. It would be useful however to model network-dependent aspects of the global Internet. Such a model could focus on parameters such as the proliferation of network address translation, the topology of the Internet, the number of servers visited overall, latency, loss, bandwidth, and the locality of network traffic.

The nature of the Internet and its usage is constantly changing. With an infrastructure such as NETI@home in place, changes to Internet usage, and thus updates to our models, should be studied. This will not only allow for studies comparing changing trends, but will ensure the availability of accurate and updated simulation models.

Finally, we have chosen to represent our models in empirical form. Such a form has its advantages, however analytical models could be developed from this data. These analytical models may have advantages for scaling, both temporally and spatially.

There are several areas of future work related to the NETI@home project, the most important of these being participation by the research community. As the data collected by NETI@home is now publicly available, we would like to foster collaboration on the analysis of this data. Further, we would like to promote the reuse of tools used to analyze this data in the hopes of easing development and verification of results.

Complementary to analysis, the NETI@home application can and should be upgraded to include more measurements, especially based on feedback from analyses.

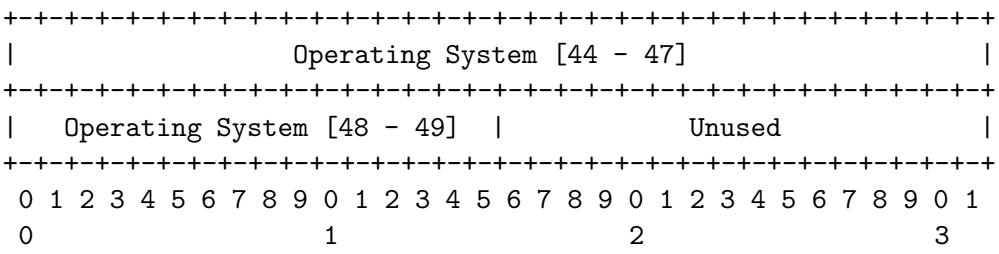
In conclusion, the research possibilities presented by the NETI@home dataset are extensive and would not be possible were it not for the participation of NETI@home users all over the world. To them, we owe our thanks.

APPENDIX A

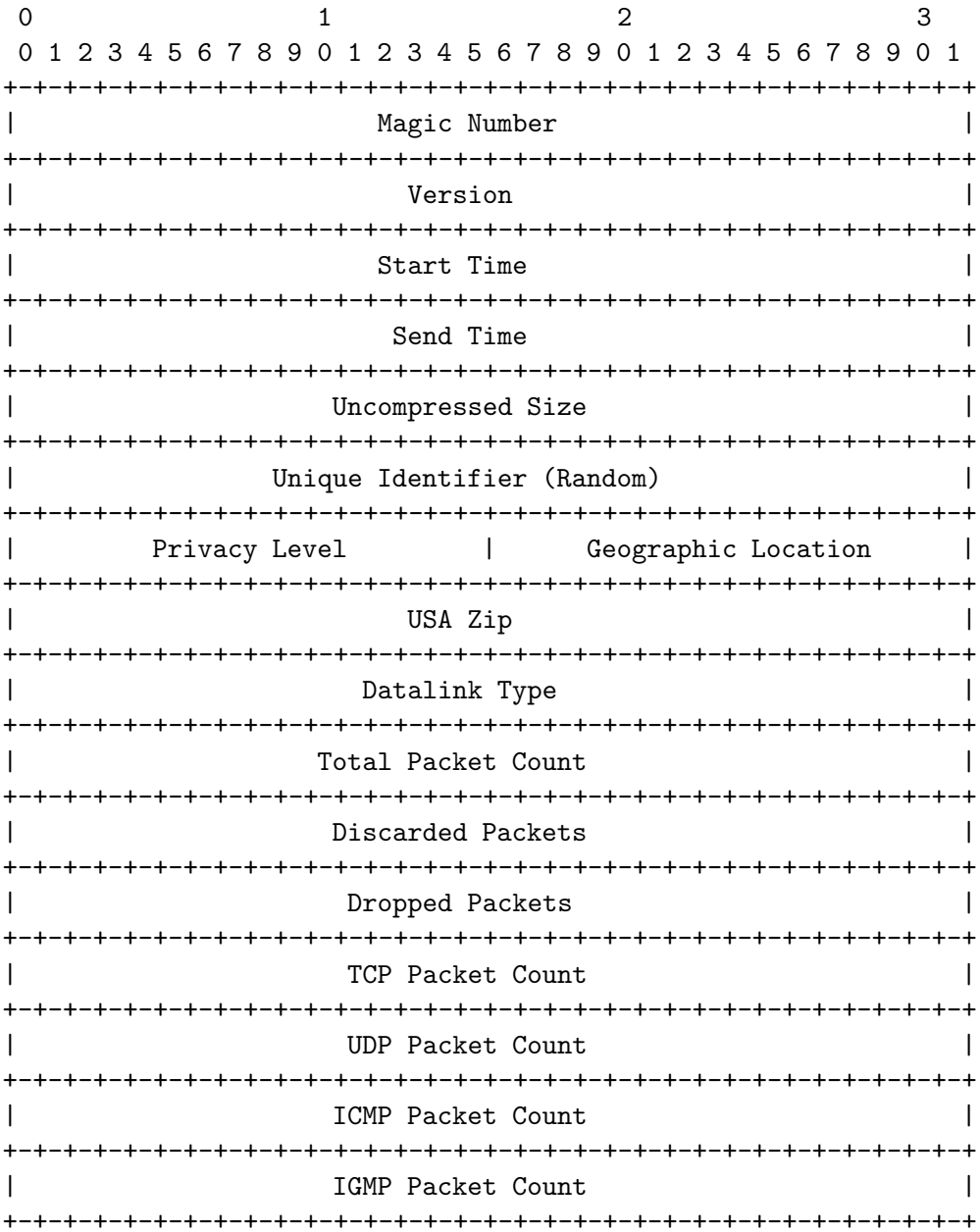
NETI@HOME STRUCTURES

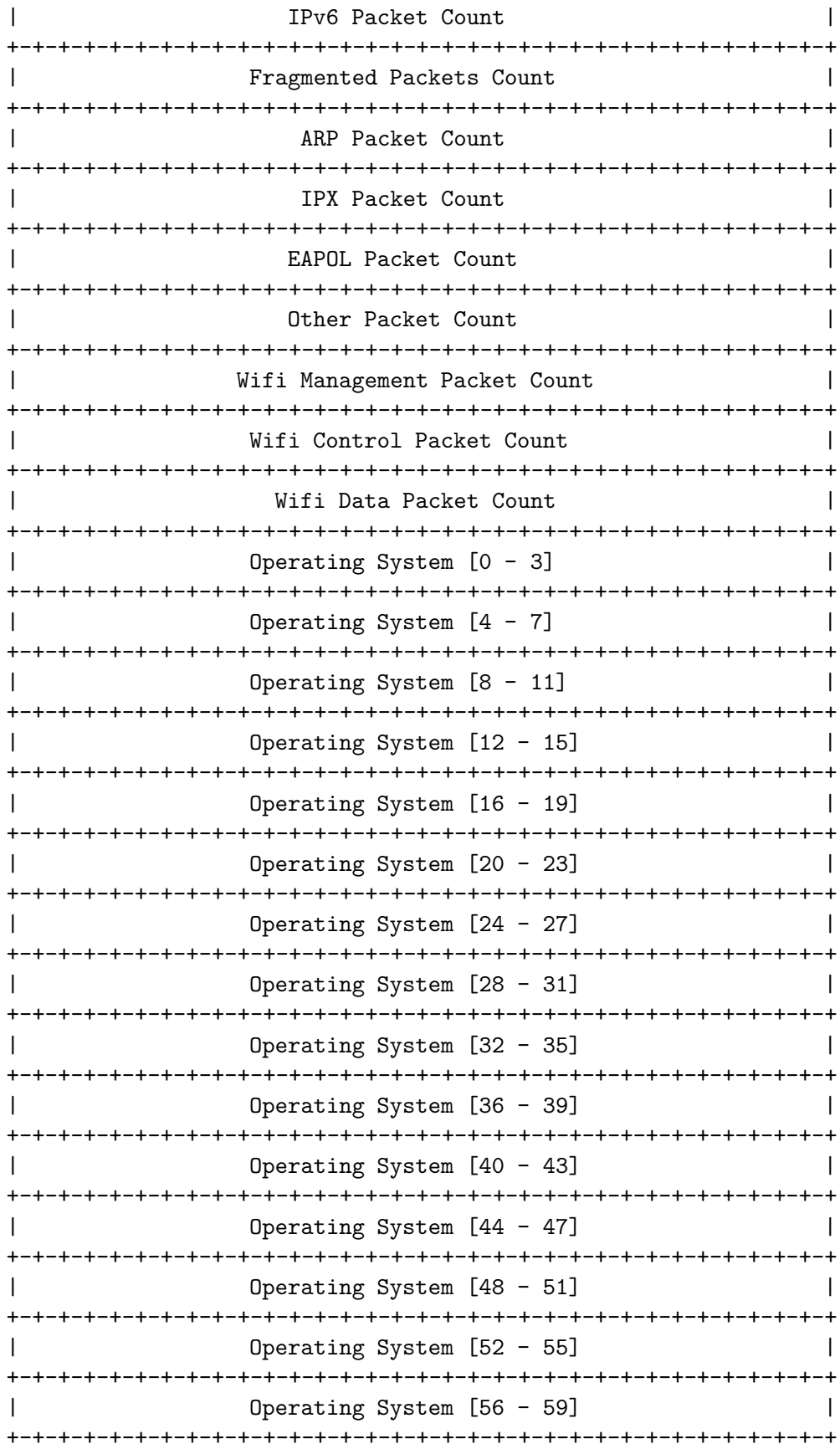
This appendix contains the various structures used by NETI@home, both versions 1 and 2, in RFC format.

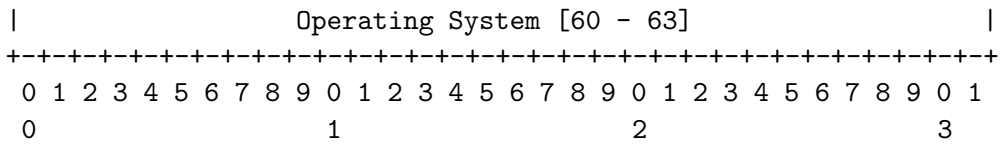
```
====================
NETI@home Header v 1
====================

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Version                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Send Time                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Uncompressed Size                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [0 - 3]                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [4 - 7]                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [8 - 11]                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [12 - 15]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [16 - 19]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [20 - 23]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [24 - 27]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [28 - 31]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [32 - 35]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [36 - 39]                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Operating System [40 - 43]                  |
```



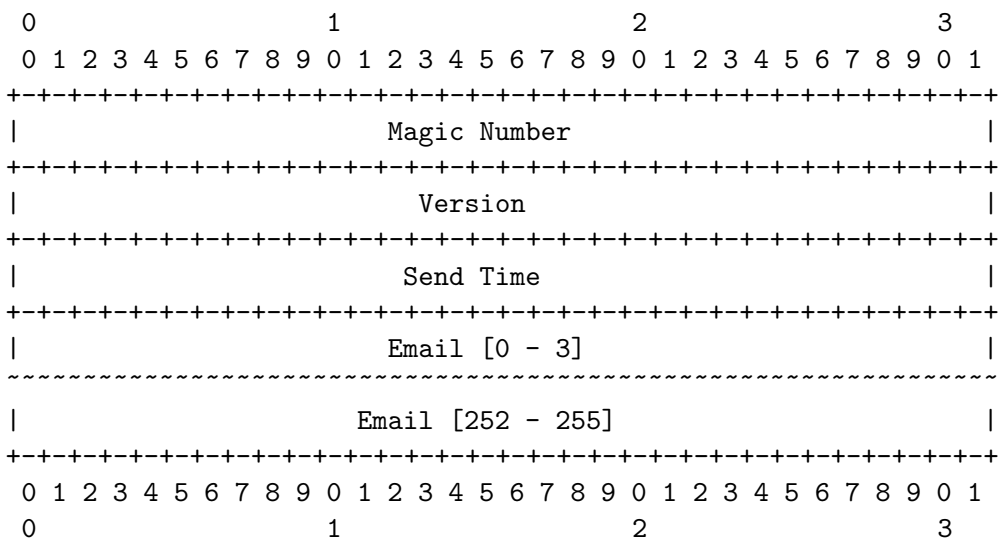
==========
NETI@home Header v 2
==========



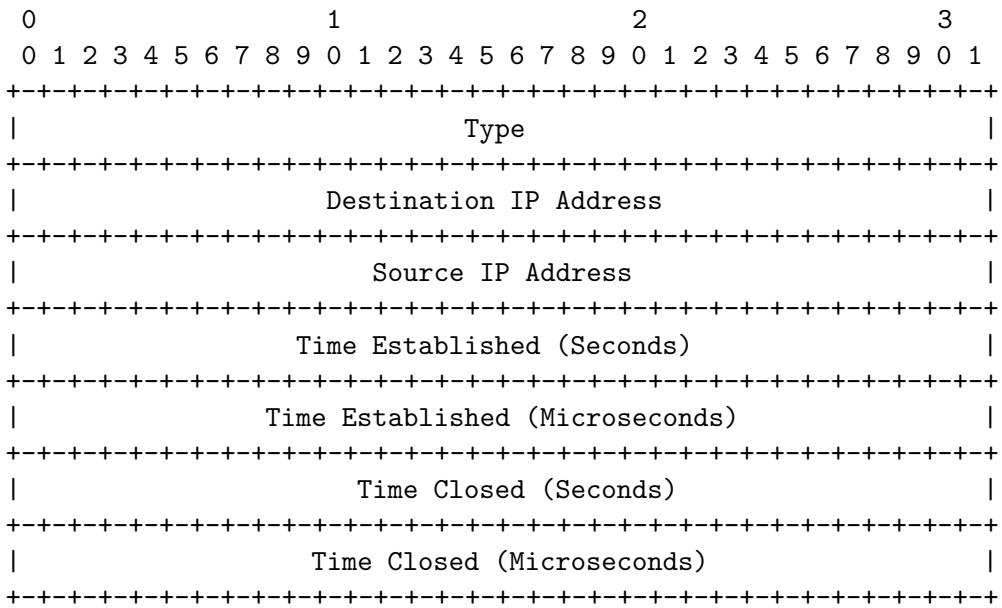

IPv6 Packet Count
Fragmented Packets Count
ARP Packet Count
IPX Packet Count
EAPOL Packet Count
Other Packet Count
Wifi Management Packet Count
Wifi Control Packet Count
Wifi Data Packet Count
Operating System [0 - 3]
Operating System [4 - 7]
Operating System [8 - 11]
Operating System [12 - 15]
Operating System [16 - 19]
Operating System [20 - 23]
Operating System [24 - 27]
Operating System [28 - 31]
Operating System [32 - 35]
Operating System [36 - 39]
Operating System [40 - 43]
Operating System [44 - 47]
Operating System [48 - 51]
Operating System [52 - 55]
Operating System [56 - 59]

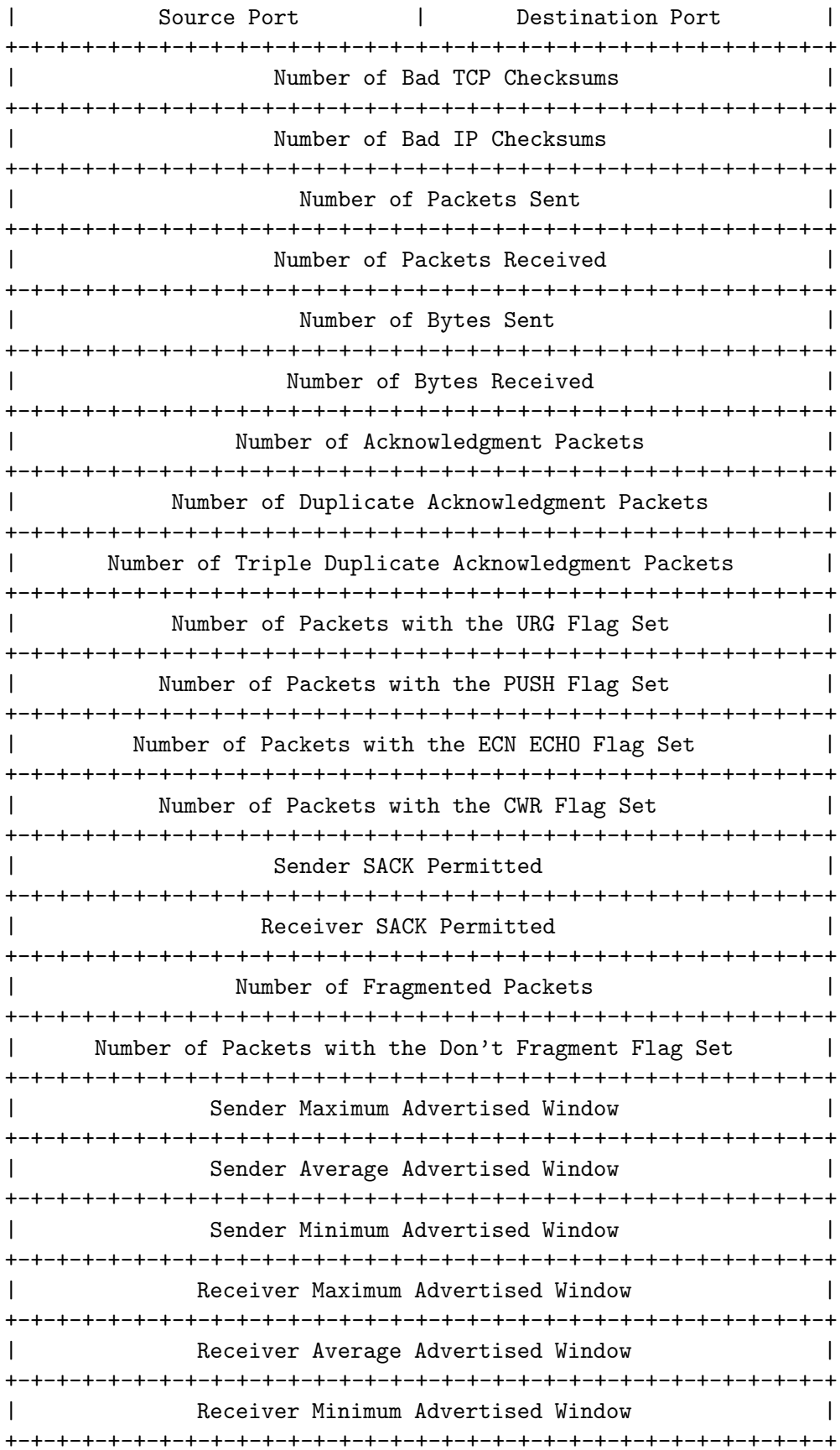


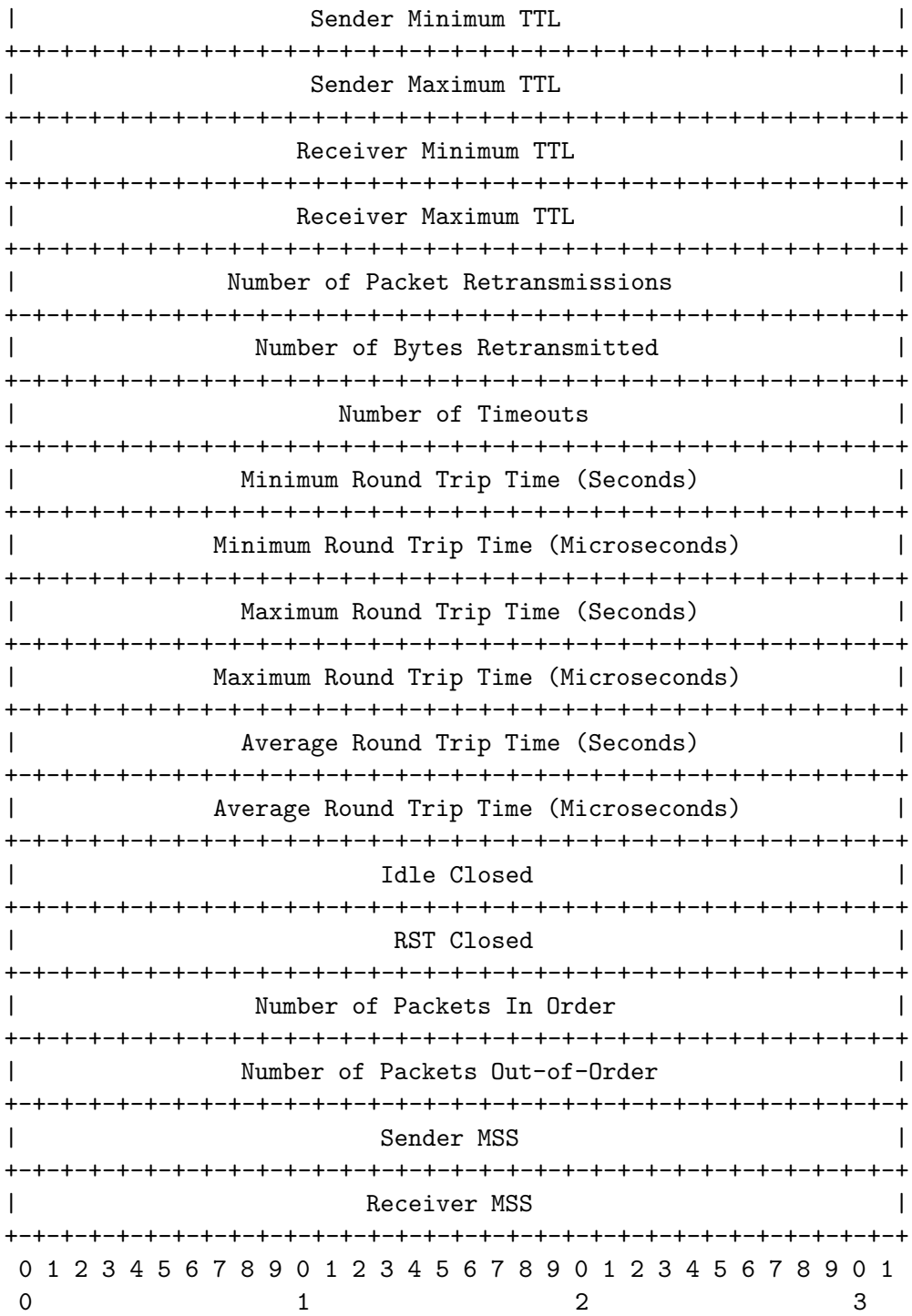
==========
Email Header v 2
==========



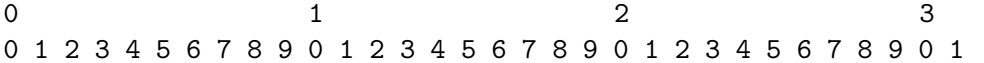
==========
NETI@home TCP_Stats v 1
==========

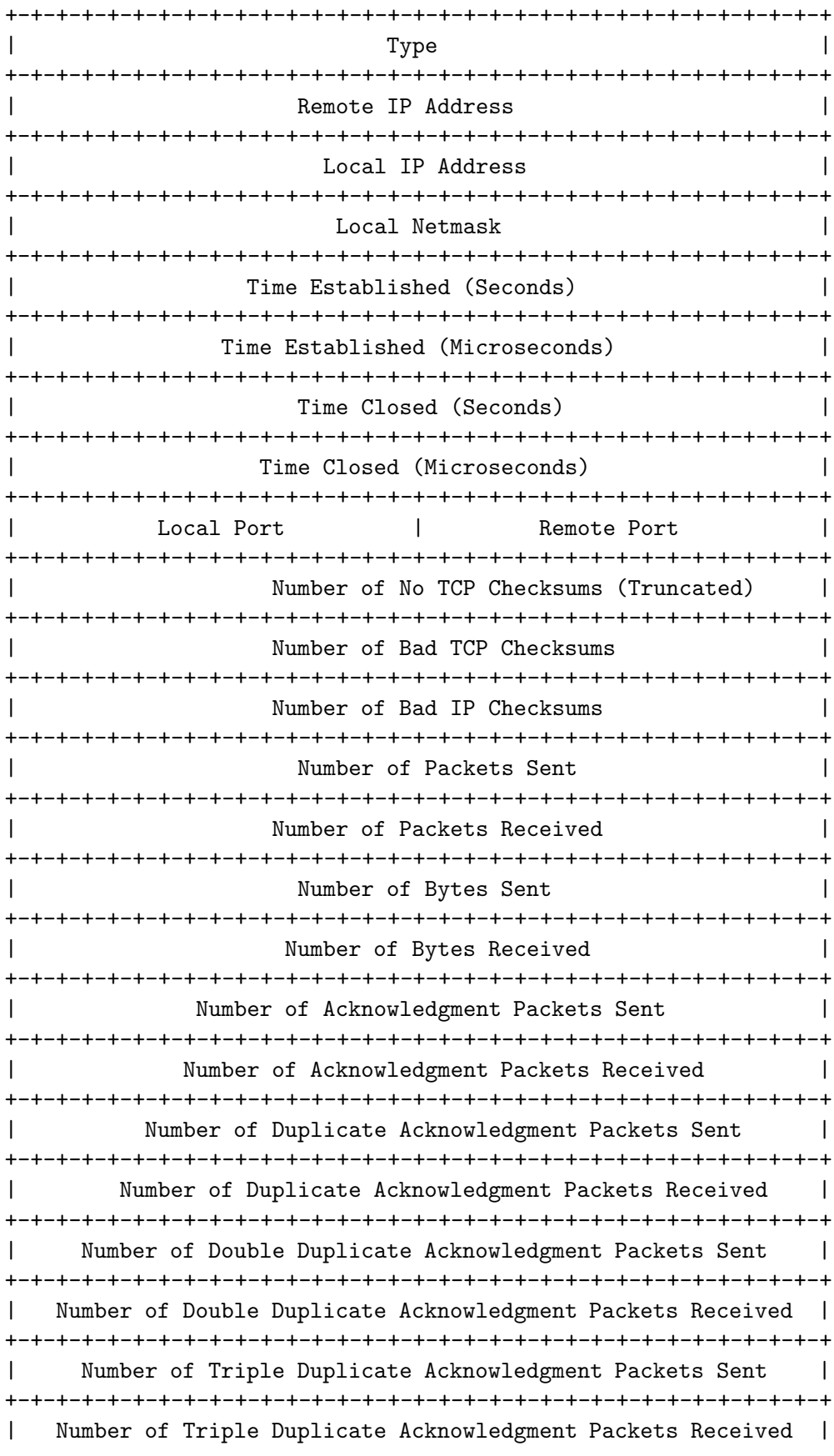


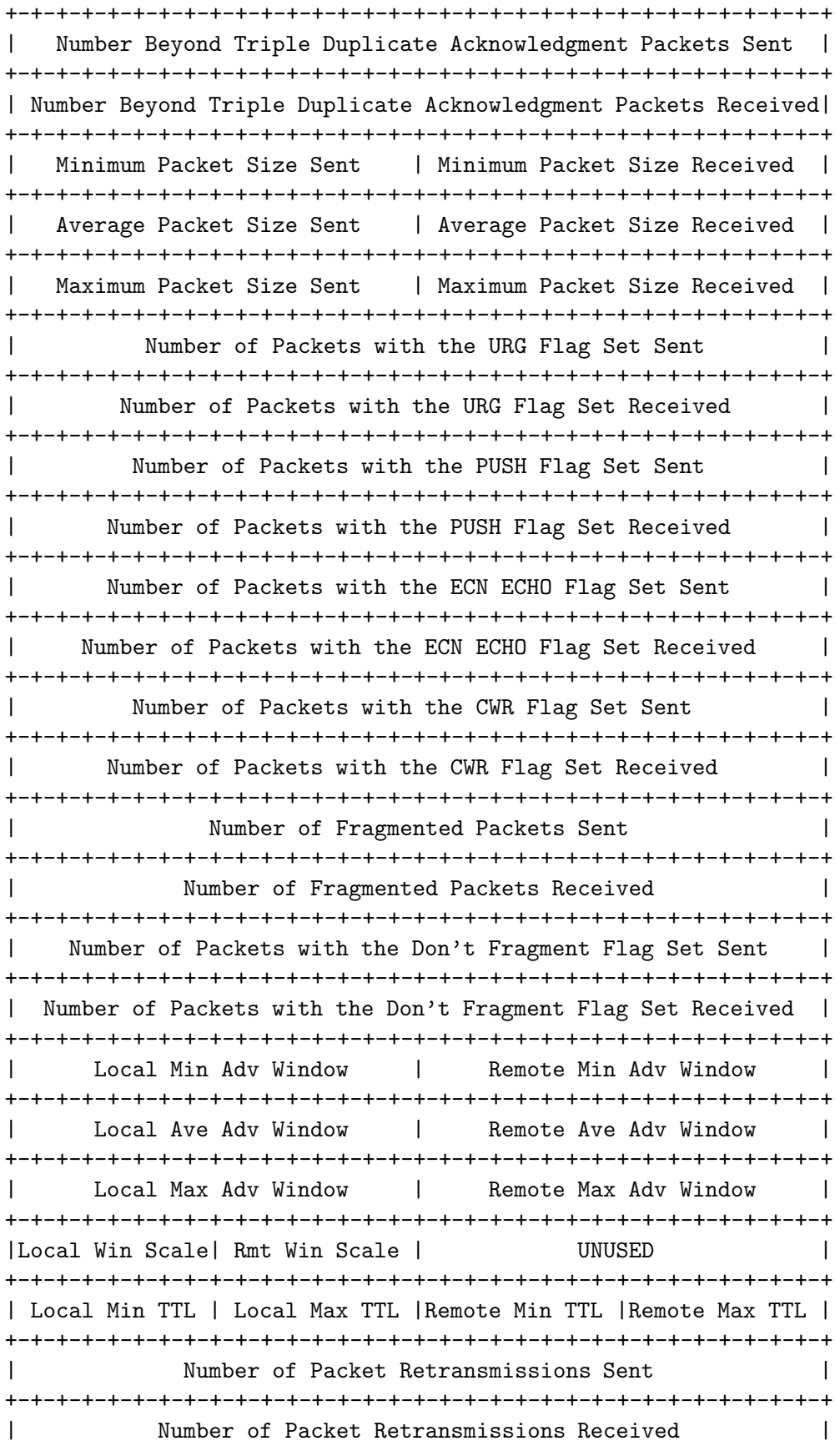

Source Port
Destination Port
Number of Bad TCP Checksums
Number of Bad IP Checksums
Number of Packets Sent
Number of Packets Received
Number of Bytes Sent
Number of Bytes Received
Number of Acknowledgment Packets
Number of Duplicate Acknowledgment Packets
Number of Triple Duplicate Acknowledgment Packets
Number of Packets with the URG Flag Set
Number of Packets with the PUSH Flag Set
Number of Packets with the ECN ECHO Flag Set
Number of Packets with the CWR Flag Set
Sender SACK Permitted
Receiver SACK Permitted
Number of Fragmented Packets
Number of Packets with the Don't Fragment Flag Set
Sender Maximum Advertised Window
Sender Average Advertised Window
Sender Minimum Advertised Window
Receiver Maximum Advertised Window
Receiver Average Advertised Window
Receiver Minimum Advertised Window

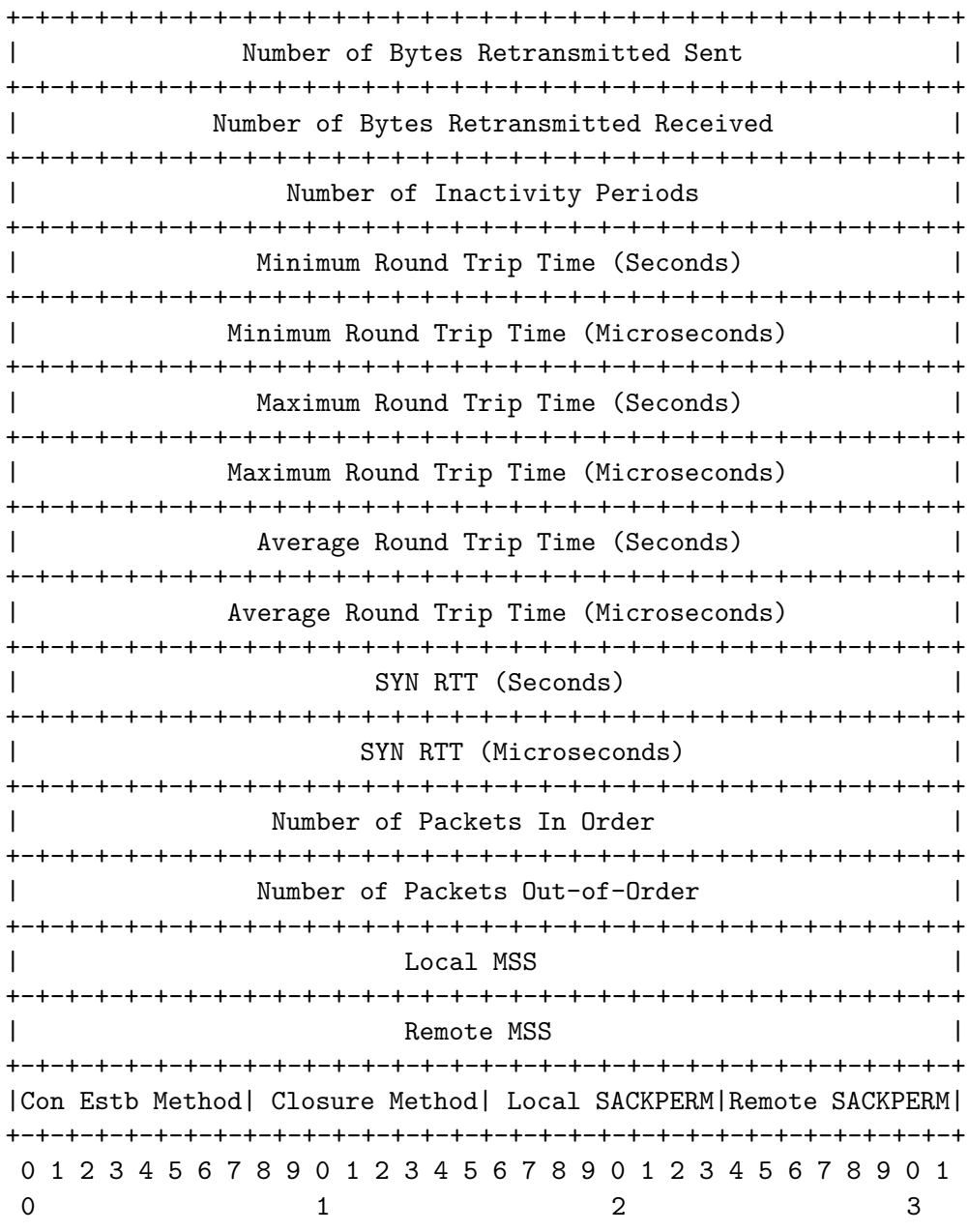



```
=======================
NETI@home TCP_Stats v 2
=======================
```

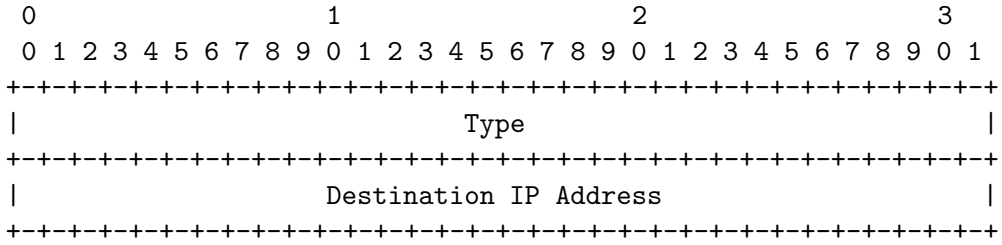


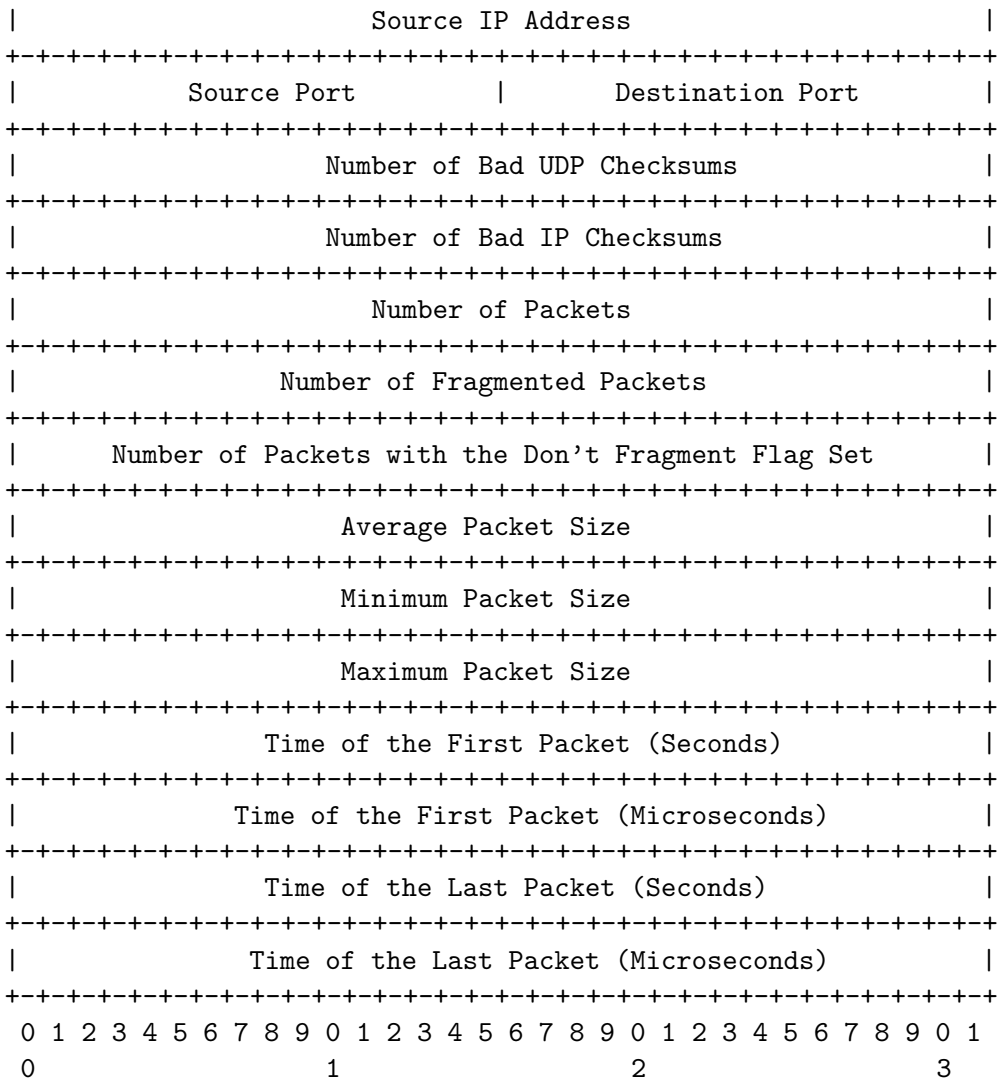

Type
Remote IP Address
Local IP Address
Local Netmask
Time Established (Seconds)
Time Established (Microseconds)
Time Closed (Seconds)
Time Closed (Microseconds)
Local Port
Remote Port
Number of No TCP Checksums (Truncated)
Number of Bad TCP Checksums
Number of Bad IP Checksums
Number of Packets Sent
Number of Packets Received
Number of Bytes Sent
Number of Bytes Received
Number of Acknowledgment Packets Sent
Number of Acknowledgment Packets Received
Number of Duplicate Acknowledgment Packets Sent
Number of Duplicate Acknowledgment Packets Received
Number of Double Duplicate Acknowledgment Packets Sent
Number of Double Duplicate Acknowledgment Packets Received
Number of Triple Duplicate Acknowledgment Packets Sent
Number of Triple Duplicate Acknowledgment Packets Received


Number Beyond Triple Duplicate Acknowledgment Packets Sent
Number Beyond Triple Duplicate Acknowledgment Packets Received
Minimum Packet Size Sent
Minimum Packet Size Received
Average Packet Size Sent
Average Packet Size Received
Maximum Packet Size Sent
Maximum Packet Size Received
Number of Packets with the URG Flag Set Sent
Number of Packets with the URG Flag Set Received
Number of Packets with the PUSH Flag Set Sent
Number of Packets with the PUSH Flag Set Received
Number of Packets with the ECN ECHO Flag Set Sent
Number of Packets with the ECN ECHO Flag Set Received
Number of Packets with the CWR Flag Set Sent
Number of Packets with the CWR Flag Set Received
Number of Fragmented Packets Sent
Number of Fragmented Packets Received
Number of Packets with the Don't Fragment Flag Set Sent
Number of Packets with the Don't Fragment Flag Set Received
Local Min Adv Window
Remote Min Adv Window
Local Ave Adv Window
Remote Ave Adv Window
Local Max Adv Window
Remote Max Adv Window
Local Win Scale
Rmt Win Scale
UNUSED
Local Min TTL
Local Max TTL
Remote Min TTL
Remote Max TTL
Number of Packet Retransmissions Sent
Number of Packet Retransmissions Received

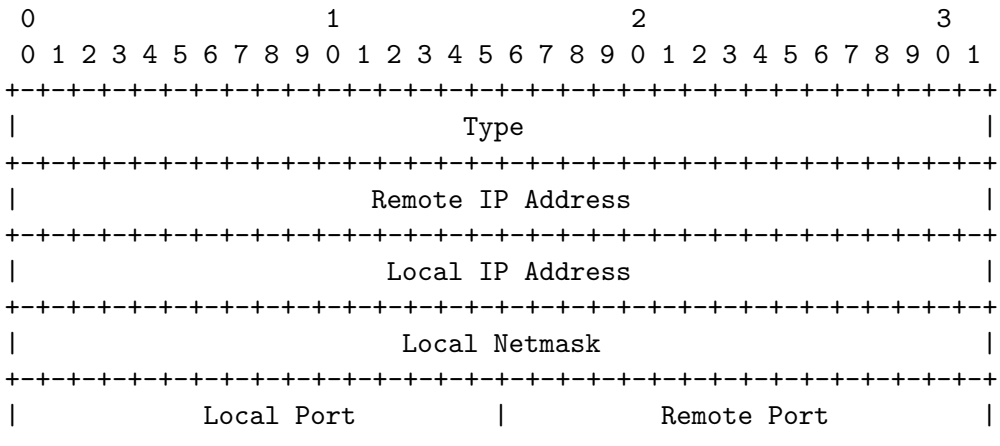


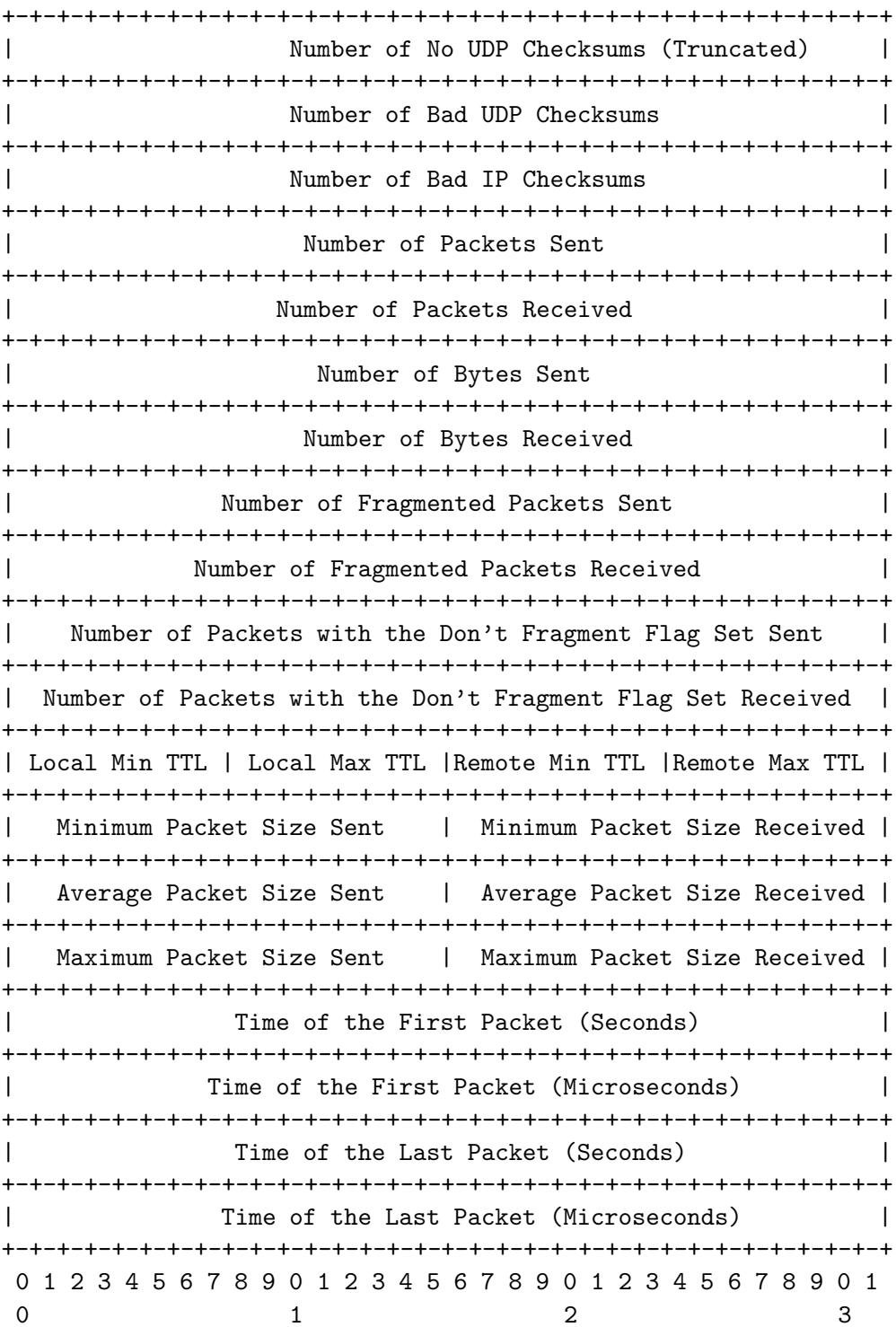

```
======================
NETI@home UDP_Stats v 1
======================
```



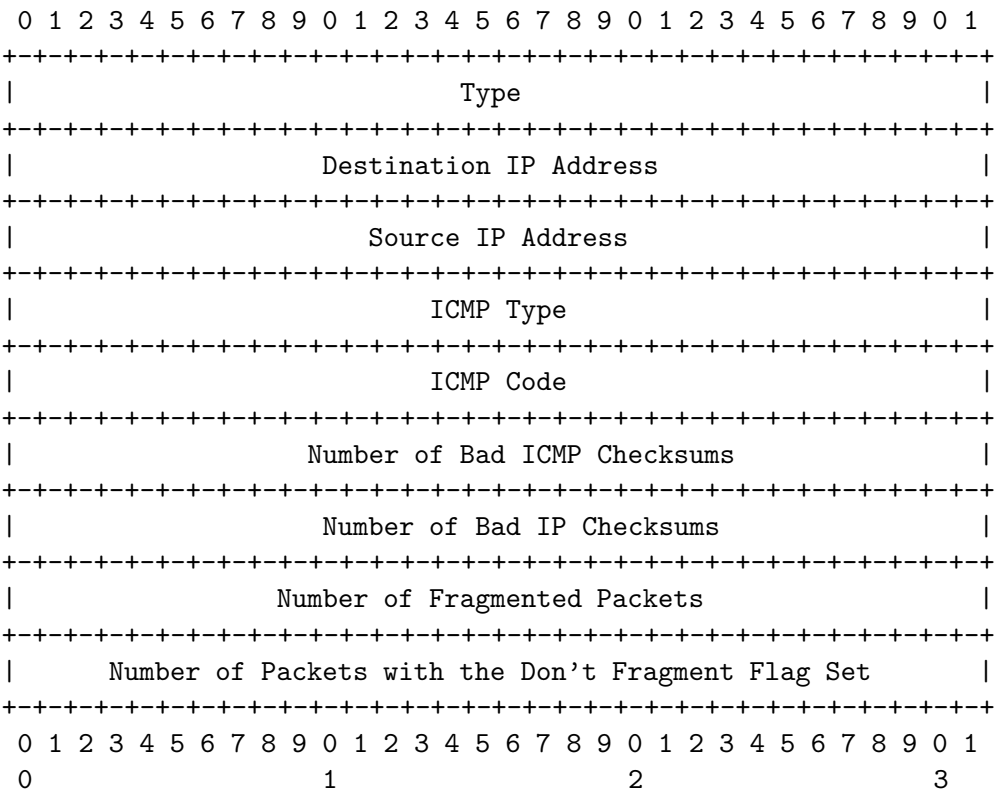



```
=======================
NETI@home UDP_Stats v 2
=======================
```

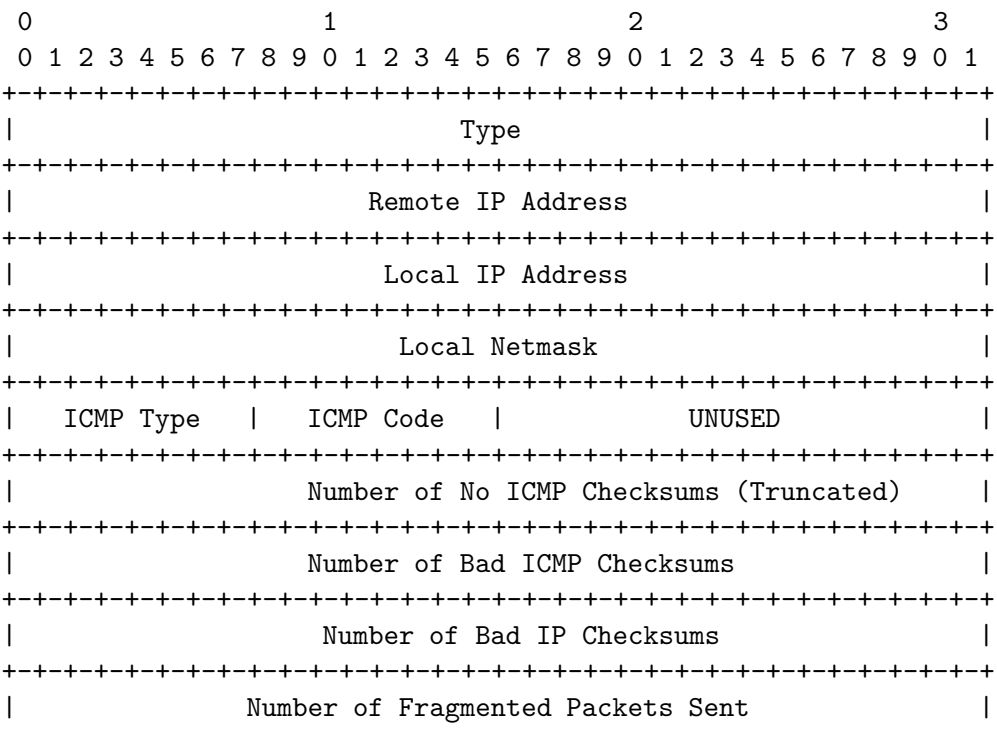


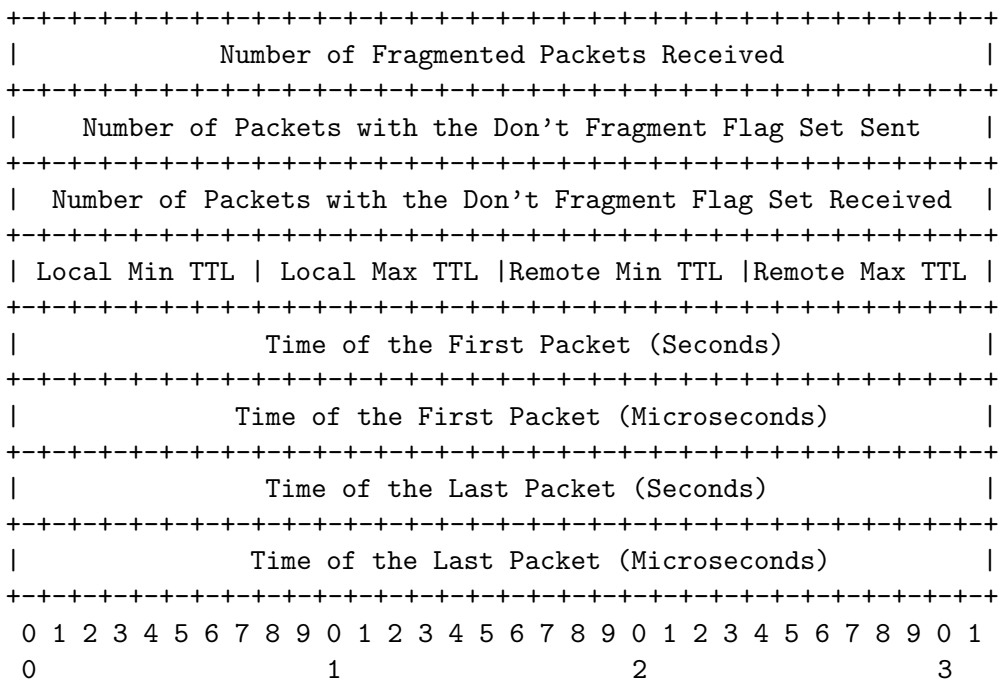



```
========================
NETI@home ICMP_Stats v 1
========================

 0                   1                   2                   3
```

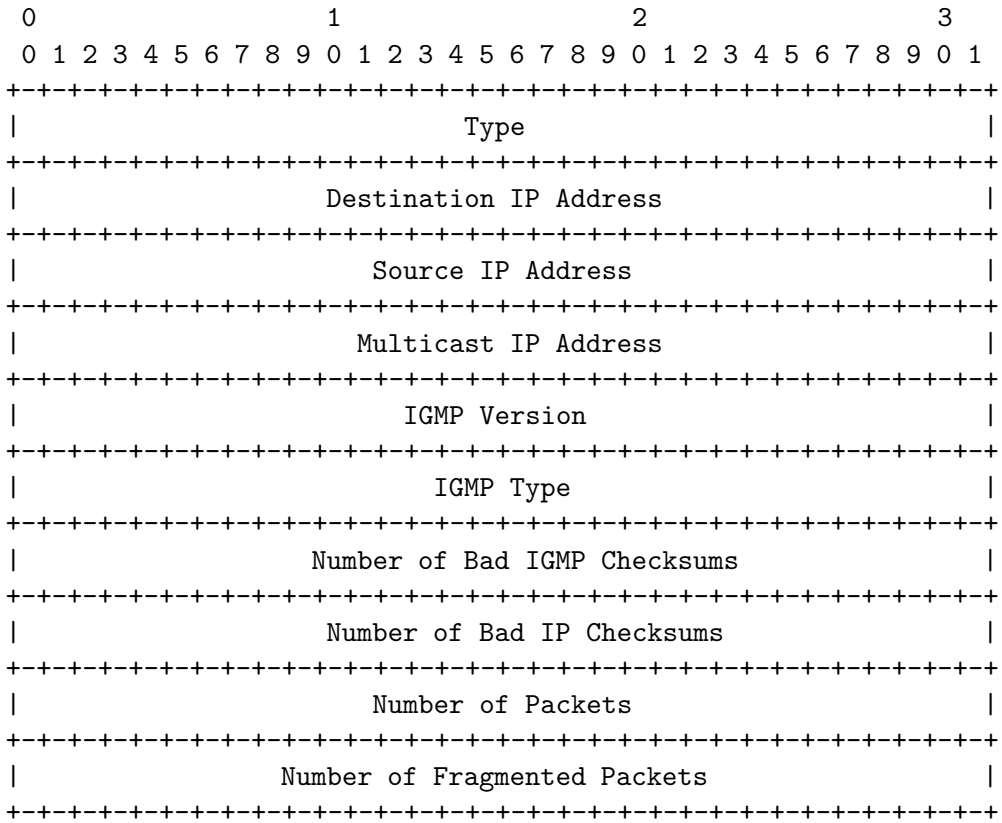


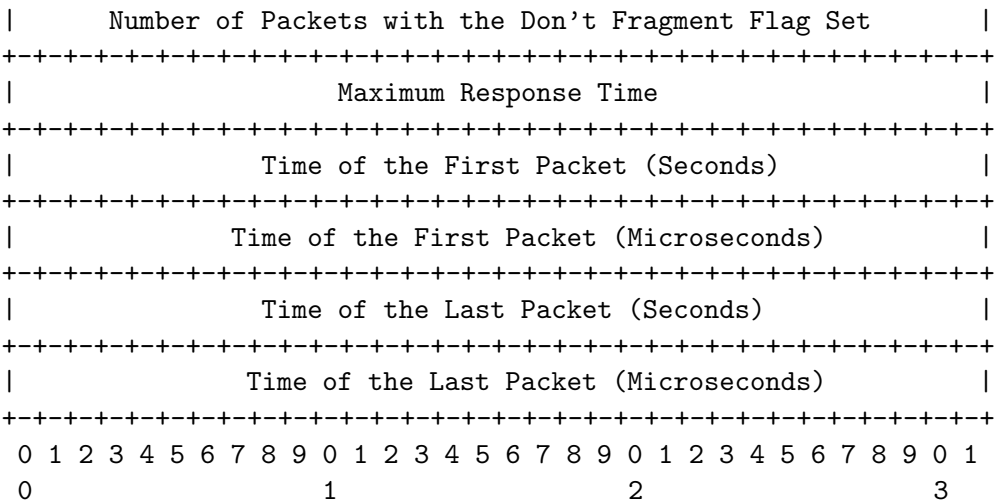

```
========================
NETI@home ICMP_Stats v 2
========================
```



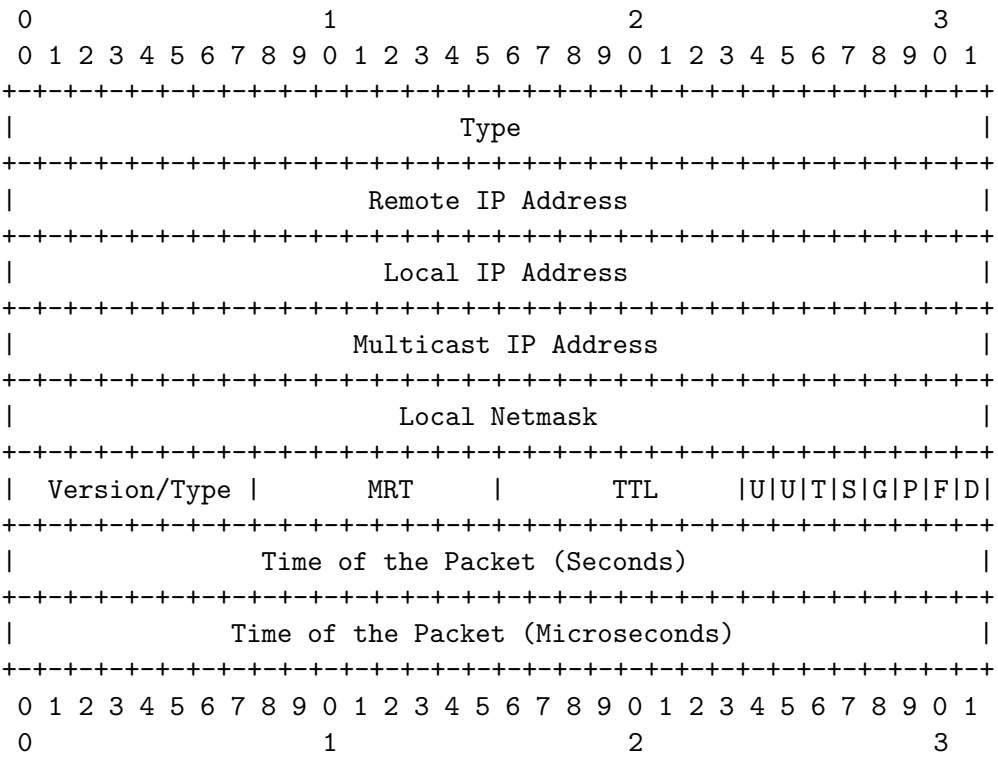



```
========================
NETI@home IGMP_Stats v 1
========================
```






```
========================
NETI@home IGMP_Stats v 2
========================
*** Now per Packet***
```




```
Version/Type => IGMP Version/Type (8 bits)
MRT => Maximum Response Time (8 bits)
U => Unused
T => Truncated (1 if truncated, 0 if not) --> No Checksum
S => Sender (1 if Local-->Remote, 0 if Remote-->Local)
```

```
G => IGMP Checksum (1 if bad, 0 if good)
P => IP Checksum (1 if bad, 0 if good)
F => Fragmented (1 if fragmented, 0 if not)
D => Don't Fragment Flag (1 if don't fragment, 0 if not)
```

REFERENCES

[1] "Amazon.com: website info: SourceForge." Available on-line: `http://www.alexa.com/data/details/traffic_details?url=sourceforge.net`, October 2006.

[2] Anderson, D. P. and et al., "SETI@home: Search for extraterrestrial intelligence at home." Software on-line: `http://setiathome.ssl.berkeley.edu`, 2003.

[3] Barakat, C., Thiran, P., Iannaccone, G., Diot, C., and Owezarski, P., "Modeling internet backbone traffic at the flow level," *IEEE Transactions on Signal Processing – Special Issue on Networking*, vol. 51, August 2003.

[4] Barford, P. and Crovella, M., "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS*, 1998.

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

[6] Bethencourt, J., Franklin, J., and Vernon, M., "Mapping internet sensors with probe response attacks," in *14th USENIX Security Symposium*, pp. 193–208, August 2005. Awarded Best Paper.

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

[8] Cao, J., Cleveland, W. S., Gao, Y., Jeffay, K., Smith, F. D., and Weigle, M. C., "Stochastic models for generating synthetic HTTP source traffic," in *IEEE INFOCOMM*, March 2004.

[9] "Cert advisory ca-1998-01 smurf ip denial-of-service attacks." `http://www.cert.org/advisories/CA-1998-01.html`, January 1998.

[10] Cheng, Y.-C., Holzle, U., Cardwell, N., Savage, S., and Voelker, G. M., "Monkey see, monkey do: A tool for TCP tracing and replaying," in *Proceedings of USENIX Technical Conference*, June 2004.

[11] Choi, H.-K. and Limb, J. O., "A behavioral model of web traffic," in *ICNP*, 1999.

[12] Christiansen, M., Jeffay, K., Ott, D., and Smith, F. D., "Tuning RED for web traffic," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 249–264, June 2001.

[13] CmdrTaco, "NETI@Home to examine net's strengths." On-line: `http://slashdot.org/article.pl?sid=04/04/27/1257211&mode=thread&tid=126&tid=95`, April 2004. Slashdot.

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

[15] 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.

[16] daemon9, "Project loki." `http://www.phrack.org/archives/49/P49-06`, August 1996.

[17] Degioanni, L., Varenni, G., Risso, F., and Bruno, J., "WinPcap." Software on-line: `http://www.winpcap.org`, 2006.

[18] Delio, M., "NETI to examine net's strengths." On-line: `http://www.wired.com/news/technology/0,1282,63180,00.html?tw=wn_techhead_2`, April 2004. Wired.

[19] "DIMES – distributed internet measurements and simulations." `http://www.netdimes.org/`, April 2006.

[20] Droms, R., "Dynamic host configuration protocol," March 1997. RFC 2131.

[21] Fan, J., Xu, J., Ammar, M. H., and Moon, S. B., "Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253–272, 2004.

[22] Floyd, S. and Paxson, V., "Difficulties in simulating the internet," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 392–403, August 2001.

[23] 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.

[24] "Georgia Tech honeynet research project." `http://www.ece.gatech.edu/research/labs/nsa/honeynet.shtml`, March 2005.

[25] "GNU general public license." `http://www.gnu.org/licenses/`, 1991.

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

[27] Grizzard, J. B., Simpson, Jr., C. R., Krasser, S., Owen, H. L., and Riley, G. F., "Flow based observations from NETI@home and honeynet data," in *Proceedings from the sixth IEEE Systems, Man and Cybernetics Information Assurance Workshop*, pp. 244–251, June 2005. Best Paper Nominee.

[28] Hernandez-Campos, F., Smith, F. D., and Jeffay, K., "Generating realistic TCP workloads," in *Computer Measurement Group International Conference*, December 2004.

[29] Hernandez-Campos, F., Nobel, A. B., Smith, F. D., and Jeffay, K., "Understanding patterns of TCP connection usage with statistical clustering," in *IEEE MASCOTS*, 2005.

[30] Jacobson, V. and Braden, R., "TCP extensions for long-delay paths," October 1988. RFC 1072.

[31] Jacobson, V., Braden, R., and Borman, D., "TCP extensions for high performance," May 1992. RFC 1323.

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

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

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

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

[36] Kenney, M., "Ping of death." `http://insecure.org/sploits/ping-o-death.html`, October 1996.

[37] Kent, C. A. and Mogul, J. C., "Fragmentation considered harmful," in *SIGCOMM '87: Proceedings of the ACM Workshop on Frontiers in Computer Communications Technology*, 1988.

[38] 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/`.

[39] Kienzle, D. M. and Elder, M. C., "Recent worms: a survey and trends," in *WORM'03: Proceedings of the 2003 ACM workshop on Rapid Malcode*, pp. 1–10, ACM Press, 2003.

[40] Larson, M. and Barber, P., "Observed DNS resolution misbehavior," October 2006. RFC 4697.

[41] Le, L., Aikat, J., Jeffay, K., and Smith, F. D., "The effects of active queue management on web performance," in *ACM SIGCOMM*, pp. 265–276, August 2003.

[42] Levine, J., LaBella, R., Owen, H., Contis, D., and Culver, B., "The use of honeynets to detect exploited systems across large enterprise networks," in *Proceedings of 4th IEEE Information Assurance Workshop*, (West Point, NY), June 2003.

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

[44] Mah, B. A., "An empirical model of HTTP network traffic," in *IEEE INFOCOMM*, April 1997.

[45] Martin, K., "AOL search data identified individuals." `http://www.securityfocus.com/brief/277`, August 2006.

[46] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A., "TCP selective acknowledgment options," October 1996. RFC 2018.

[47] Mathis, M., Semke, J., Mahdavi, J., and Ott, T., "The macroscopic behavior of the TCP congestion avoidance algorithm," *SIGCOMM Computer Communications Review*, vol. 27, no. 3, 1997.

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

[49] Moore, D., "Network telescopes: Observing small or distant security events." `http://www.caida.org/outreach/presentations/2002/usenix_sec/`, August 2002. Invited Presentation at the 11th USENIX Security Symposium (SEC 02).

[50] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N., "Inside the slammer worm," *Security & Privacy Magazine*, vol. 1, no. 4, pp. 33–39, 2003.

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

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

[53] Muuss, M., "ping." Software on-line: http://ftp.arl.mil/ mike/ping.html, 1983. Ballistic Research Lab.

[54] "nessus." `http://www.nessus.org/`, March 2005.

[55] "nmap." `http://www.insecure.org/nmap/`, March 2005.

[56] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J., "Modeling TCP throughput: A simple model and its empirical validation," in *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998.

[57] Paxson, V., "Strategies for sound internet measurement," in *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, 2004.

[58] 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.

[59] Postel, J., "Internet protocol," September 1981. RFC 791.

[60] Postel, J., "The TCP maximum segment size and related topics," November 1983. RFC 879.

[61] Provos, N., "A virtual honeypot framework," in *13th USENIX Security Symposium*, August 2004.

[62] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E., "Address allocation for private internets," February 1996. RFC 1918.

[63] Reynolds, J. and Postel, J., "Assigned numbers," October 1994. RFC 1700.

[64] Riley, G. F., "The Georgia Tech Network Simulator," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pp. 5–12, 2003.

[65] Shannon, C. and Moore, D., "The spread of the witty worm," *Security & Privacy Magazine*, vol. 2, no. 4, pp. 46–50, 2004.

[66] Shannon, C., Moore, D., and Claffy, K. C., "Beyond folklore: Observations on fragmented traffic," *IEEE/ACM Transactions on Networking*, vol. 10, December 2002.

[67] Shannon, C., Moore, D., Keys, K., Fomenkov, M., Huffaker, B., and k claffy, "The internet measurement data catalog," *SIGCOMM Computer Communications Review*, vol. 35, no. 5, 2005.

[68] Simpson, Jr., C. R., "NETI@home." Software on-line: `http://neti.gatech.edu`, 2003. Georgia Institute of Technology.

[69] Simpson, Jr., C. R., "A distributed approach to passively gathering end-to-end network performance measurements," Master's thesis, Georgia Institute of Technology, May 2004.

[70] Simpson, Jr., C. R., Reddy, D., and Riley, G. F., "Empirical models of end-user network behavior from NETI@home data analysis," *Simulation: Transactions of the Society for Modeling and Simulation International*. Invited Paper – Awaiting Publication.

[71] Simpson, Jr., C. R., Reddy, D., and Riley, G. F., "Empirical models of TCP and UDP end-user network traffic from NETI@home data analysis," in *20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006)*, pp. 166–174, May 2006. Best Paper Nominee.

[72] Simpson, Jr., 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.

[73] "skitter." `http://www.caida.org/tools/measurement/skitter/`, 2006. CAIDA.

[74] Slagell, A. and Yurcik, W., "Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization," in *SECOVAL: The Workshop on the Value of Security through Collaboration, held in conjunction with SecureComm*, September 2005.

[75] Smith, F. D., Hernandez-Campos, F., Jeffay, K., and Ott, D., "What TCP/IP protocol headers can tell us about the web," in *ACM SIGMETRICS*, pp. 245–256, 2001.

[76] Sommers, J., Kim, H., and Barford, P., "Harpoon: A flow-level traffic generator for router and network tests," in *ACM SIGMETRICS*, June 2004.

[77] "SourceForge." `http://sourceforge.net/`, April 2006.

[78] Spitzner, L., "Know your enemy: Honeynets." `http://www.honeynet.org/papers/honeynet/`. Honeynet Project.

[79] "Standard country and area codes classifications (M49)." http://unstats.un.org/unsd/methods/m49/m49regin.htm, July 2006. United Nations Statistics Division.

[80] The Apache Software Foundation, "Apache." Software on-line: http://www.apache.org, 2005.

[81] "The SANS internet storm center." http://isc.sans.org.

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

[83] timothy, "NETI@home data analyzed." On-line: http://it.slashdot.org/it/05/04/25/1710223.shtml?tid=172&tid=95&tid=218, April 2005. Slashdot.

[84] "traceroute@home." http://www.traceroutеathome.net/, April 2006.

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

[86] Weaver, N., Paxson, V., Staniford, S., and Cunningham, R., "A taxonomy of computer worms," in *WORM'03: Proceedings of the 2003 ACM workshop on Rapid Malcode*, pp. 11–18, ACM Press, 2003.

[87] Weigle, M., Jeffay, K., and Smith, F. D., "Delay-based early congestion detection and adaptation in TCP: Impact on web performance," *ACM Computer Communications Review*, vol. 28, pp. 837–850, May 2005.

[88] Xu, J. and Lee, W., "Sustaining availability of web services under distributed denial of service attacks," *IEEE Transactions on Computers*, vol. 52, pp. 195–208, February 2003.

[89] Zhang, Y., Breslau, L., Paxson, V., and Shenker, S., "On the characteristics and origins of internet flow rates," in *ACM SIGCOMM*, August 2002.

VITA

Charles Robert Simpson, Jr. ("Robby") was born in Spartanburg, South Carolina in 1980. He received his B.S. in Computer Engineering from Clemson University in 2002. He joined the School of Electrical and Computer Engineering at the Georgia Institute of Technology in 2002 from which he received an M.S.E.C.E. in 2004. He completed his studies at the Georgia Institute of Technology when he graduated with a Ph.D. in 2006. His research interests include network measurements, networking protocols, network security, network simulations, computer architecture, and software engineering.