NETI@home: A Distributed Approach to Collecting End-to-End Network Performance Measurements

Charles Robert Simpson, Jr. and George F. Riley

Georgia Institute of Technology (Georgia Tech), Atlanta Georgia, USA,
{rsimpson, riley}@ece.gatech.edu

Abstract. NETI@home is an open-source software package that collects network performance statistics from end-systems. It has been written for and tested on the Windows, Solaris, and Linux operating systems, with testing for other operating systems to be completed soon. NETI@home is designed to run on end-user machines and collects various statistics about Internet performance. These statistics are then sent to a server at the Georgia Institute of Technology, where they are collected and made publicly available. This tool gives researchers much needed data on the end-to-end performance of the Internet, as measured by end-users. Our 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.

1 Introduction

The need for the passive collection of end-to-end Internet performance measurements has been noted by many [1][2]. Active measurements of Internet performance have their obvious drawbacks and some believe that their results are not accurate due to the amount of traffic they inject [3][4]. Other attempts at the passive collection of Internet performance statistics have either focused on a few collection points [1] or have the need to deploy costly and restrictive collection nodes [1][5]. Some Internet service providers (ISPs) collect end-to-end performance measurements as well; however they are usually reluctant to share this data. Further, the collection methods used by each ISP can differ dramatically and are usually not as detailed as other methods [1]. NETI@home passively gathers these measurements and makes them available to researchers while at the same time respecting the rights and privacy of the end-users. This article serves two purposes, the first is to alert the research community to NETI@home, and the second is to help publicize NETI@home so that even more users will install it on their systems and thus, more data may be collected.

2 Software Description

NETI@home is written in the C++ programming language and is built on top of the open-source Ethereal [6] software package. Ethereal is a well-written network protocol analyzer (sniffer) that is becoming the de facto network sniffer standard. NETI@home uses Ethereal's ability to sniff packets in real-time with little impact on the user's system. Should we find Ethereal to be unsuitable, we can easily use another network protocol analyzer, such as tcpdump [7], or write our own using the pcap library [8]. The sniffed packets are then sent to NETI@home for analysis, where they are sorted into bidirectional flows, as defined in [9]. These packets are *not* collected with the network interface in promiscuous mode, therefore only packets sent from and received by the user's system are analyzed.

NETI@home users are able to specify a privacy level that determines which statistics are collected and reported. At the highest privacy setting, no IP addresses are recorded. At the lowest privacy setting, all IP addresses are recorded. While the absence of IP addresses may reduce potential research possibilities, we believe that the rights and privacy of the NETI@home users are of the utmost importance.

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

One of the most important goals of the NETI@home project is to have a large installed user base. To encourage end users to run the NETI@home software, a program written in Java, when run, 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. To resolve IP addresses into latitude/longitude coordinates, the NetGeo [10] database from CAIDA is used.

3 Network Statistics

NETI@home collects many different statistics from end-users, in accordance with CAIDA specifications [9], with more statistics to be added in later releases. These statistics focus on four transport-layer protocols on the Internet: the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), the Internet Control Message Protocol (ICMP), and the Internet Group Management Protocol (IGMP), as well as their underlying network-layer protocols (such as the Internet Protocol, IP). The following is a brief list of some of the statistics gathered:

3.1 Transmission Control Protocol

– Source and Destination IP Addresses

Note: The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting.

– Source and Destination Ports
– Times the Connection was Established and Closed
– Number of Bad Checksums (IP and TCP)[1]
– Number of Packets Sent and Received
– Number of Bytes Sent and Received
– Number of Acknowledgment Packets
– Number of Duplicate Acknowledgment Packets
– Number of Triple Duplicate Acknowledgment Packets
– Number of URG Flags
– Number of PUSH Flags
– Number of ECNECHO Flags
– Number of CWR Flags
– Sender and Receiver SACK Permitted
– Sender and Receiver Minimum Window Size
– Sender and Receiver Average Window Size
– Sender and Receiver Maximum Window Size
– Sender and Receiver Minimum TTL Values
– Sender and Receiver Maximum TTL Values
– Number of Packet *Retransmissions*

A *retransmission* is defined as a packet with a sequence number less than the highest sequence number that has been observed thus far.

– Number of Bytes Retransmitted
– Number of *Inactivity Periods*

An *inactivity period* is defined to be a period in which no packets are sent or received for at least 200 milliseconds. This statistic is a rough estimate of a TCP timeout.

– Minimum, Maximum, and Average *Round Trip Times*

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, then the acknowledgment number should equal its sequence number plus one.

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

– Number of Packets Received *In-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.

– Number of Packets Received *Out-of-Order*

A packet is defined to be *out-of-order* if it is not in-order.

– Number of Fragmented Packets
– Number of Don't Fragment Flags
– Sender and Receiver Maximum Segment Size (if specified)
– Number of *Rejected Connection Requests*

A connection is deemed *rejected* if the TCP three-way handshake procedure fails.

– Was the Flow *Idle Closed*?

A connection is defined to be *idle closed* if there has been no activity in the flow for 64 seconds, as per [9].

– Was the Flow *Reset Closed*, and by Whom?

A connection is defined to be *reset closed* if a packet is sent or received with the RST flag set.

– Was the Flow *FIN Closed*, and by Whom?

A connection is defined to be *FIN closed* if a packet is sent or received with the FIN flag set.

3.2 User Datagram Protocol

– Source and Destination IP Addresses

Note: The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting.

– Source and Destination Ports
– Number of Bad Checksums (IP and UDP)[1]
– Number of Packets Sent and Received
– Average Packet Sizes
– Minimum Packet Sizes
– Maximum Packet Sizes
– Number of Fragmented Packets
– Number of Don't Fragment Flags
– Time of the First Packet
– Time of the Last Packet

3.3 Internet Control Message Protocol

– Source and Destination IP Addresses

Note: The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting.

– Number of Bad Checksums (IP and ICMP)[1]
– Number of Fragmented Packets
– Number of Don't Fragment Flags
– ICMP Type
– ICMP Code

3.4 Internet Group Management Protocol

– Source and Destination IP Addresses
– Multicast IP Address

Note: The IP addresses are only recorded if the user wishes, in accordance with their selected privacy setting.

– Number of Bad Checksums (IP and IGMP)[1]
– IGMP Version
– IGMP Type
– Number of Packets Sent and Received
– Number of Fragmented Packets
– Number of Don't Fragment Flags
– Maximum Response Time
– Time of the First Packet
– Time of the Last Packet

3.5 Miscellaneous

– NETI@home Version
– Operating System

The user's operating system is determined by either the GNU autoconf tool [11] or the appropriate functions in Microsoft Windows.

– *Send Time*

The *Send Time* is the time the data was sent to Georgia Tech, as recorded by the user's system.

– *Arrival Time*

The *Arrival Time* is the time the data arrived at Georgia Tech. By observing both the send and arrival times, major timing discrepancies between Georgia Tech and the user's system can be accommodated.

4 Distributing NETI@home

It is our goal to have NETI@home installed in many academic computer labs across the country as well as on many end-user systems in homes and offices. As the number of users increases, so does the value of our data. NETI@home is currently available for the Linux and Solaris operating systems, in RPM and tarball format, and for the Windows operating system in the form of a self-installing executable. All distributions are available from our website located at http://neti.sourceforge.net/. In addition to these distributions, we are in the process of testing distributions for other platforms such as the Macintosh operating system. To use NETI@home, one must also install the Ethereal network analyzer [6], which is available from the Ethereal website, http://www.ethereal.com/.

Since NETI@home is copyrighted (copylefted) under the GNU General Public License (GPL) [12], the source code is also available from our public website, http://neti.sourceforge.net/. 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.

5 Future Work

In the future, we will add many more features to NETI@home to aid researchers. For example, we would like to be able to collect the size of the TCP congestion window, which would allow us to monitor the TCP congestion control algorithm. However, to collect the size of the congestion window will require a lower-level application, such as a kernel module, or some other novel technique. Some other new features we are considering 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 will either not be recorded, will be trimmed, or it will be fully recorded. The traceroute was not initially included because it might tax users' systems as well as the network itself. However, using the mapping program provided with NETI@home, traceroute functionality will be provided. When a user clicks a specific host on the map, another window will appear and a graphical traceroute will be performed. Since the traceroute is explicitly performed by the user, it should not be considering taxing to the user's system or the network, and it truly represents a passive technique. This data will also be sent to the server at Georgia Tech to add to the amount of measurements that are collected. Finally, the NETI@home online database will receive many enhancements and analysis will be performed on the collected data.

6 Conclusions

NETI@home is a software package designed to run on end-user systems to collect various network performance statistics. NETI@home represents an excellent opportunity to further research into the performance of the Internet. We encourage

anyone and everyone to install the software on their machines to increase the amount of data with which researchers can work. NETI@home is an unobtrusive software package that runs quietly in the background and respects the privacy of the end-user. We further encourage researchers to use the data made publicly available by NETI@home to continue the advancement of the global Internet.

References

1. Murray, M., kc claffy: Measuring the immeasurable: Global internet measurement infrastructure. In: Proceedings of PAM2001 - A workshop on Passive and Active Measurements. (2001)
2. Hou, Y.T., Dong, Y., Zhang, Z.L.: Network performance measurement and analysis part 1: A server-based measurement infrastructure (1998) Fujitsu Laboratories of America, Technical Report FLA-NCRTM98-01.
3. Barford, P., Sommers, J.: A comparison of active and passive methods for measuring packet loss (2002) University of Wisconsin Technical Report.
4. Mochalski, K., Irmscher, K.: On the use of passive network measurements for modeling the internet. In: KiVS. (2003)
5. Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F.: Design and deployment of a passive monitoring infrastructure. Lecture Notes in Computer Science **2170** (2001)
6. Combs, G., et al.: Ethereal: - a network protocol analyzer. Software on-line: http://www.ethereal.com (2004)
7. Jacobson, V., Leres, C., McCanne, S.: tcpdump. Software on-line: http://www.tcpdump.org (1989) Lawrence Berkeley Laboratory.
8. Jacobson, V., Leres, C., McCane, S.: libpcap. Software on-line: http://www.tcpdump.org (1989) Lawrence Berkeley Laboratory.
9. CAIDA: Caida: Preliminary measurement specification for internet routers. http://www.caida.org/tools/measurement/measurementspec/ (2004) The Cooperative Association for Internet Data Analysis - CAIDA.
10. Moore, D., Periakaruppan, R., Donohoe, J., kc claffy: Where in the world is netgeo.caida.org? In: INET 2000. (2000)
11. GNU: Gnu autoconf. Software on-line: http://www.gnu.org/software/autoconf/ (1998)
12. GNU: Gnu general public license. http://www.gnu.org/licenses/ (1991)