Project Number: MLC MT03

TCP TRAFFIC ANALYSIS

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

David Loose

_____

Todd DeSantis

Date: January 14, 2003

Approved:

_____

Professor Mark Claypool, Major Advisor

_____

Professor Robert Kinicki, Co-Advisor

Table of Contents

Abstract

As broadband Internet grows in popularity, it is important to determine if improvements can be made to broadband hardware and TCP. Motorola captured three hours of traffic from a cable network, which we analyzed to identify trends that may lead to optimizations in Motorola's hardware. We found that TCP-SACK provides a performance increase and recommend its further deployment. We also found that peer-to-peer applications account for a large portion of Internet traffic and suggest measures that may be taken to address this change.

1 Introduction

Broadband Internet services affect millions of computer users all throughout the world. High-speed Internet adoption at home continues to rise sharply in the United States, increasing by 50% from March 2002 to March 2003. About 21 million broadband users connected by cable modem in March 2003, up from 13 million a year earlier [1]. Broadband is inexpensive, almost universally available, and simple to setup. The increased capacity offered by broadband allows subscribers to use the Internet in ways that were impractical with dial-up. One example of this is high-resolution streaming media. Because of this, broadband and dial-up service providers are faced with different classes of traffic that require different optimization methods. As the popularity of broadband continues to grow, so too does the need for these improvements.

Another significant change is the nature of Internet traffic. In the past, the vast majority of Internet traffic was WWW based. Now, for broadband networks aimed towards the home user, peer-to-peer (P2P) traffic accounts for the majority of HTTP bytes transferred, exceeding traffic due to WWW accesses by nearly a factor of three [1]. Furthermore, P2P documents are three orders of magnitude larger than web objects, and a small number of extremely large objects account for an enormous fraction of observed P2P traffic [2]. These significant changes in Internet traffic suggest that there may be potential to improve broadband hardware to increase performance.

This project was initiated by Motorola under Dan Grossman. In order to remain current, Motorola is interested in finding out more about network behavior from a cable broadband service perspective. They are interested in characterizing trends in network traffic, and applying this knowledge to create high performance hardware.

The purpose of this project was to assist Motorola in optimizing cable modem hardware. There have been many studies regarding TCP behavior and performance [2] [3] [5], but ours differs from studies that have been performed in the past in that we are interested in how specific options and conditions affect broadband network performance. The motivation for these conditions and options is important to consider.

Our first goal was to characterize the traffic contained in the trace files. To do this, we first examined which transport protocols were present in the trace. We also attempted to determine which application protocols were contained in the trace files by making assumptions from port numbers. Finally, we looked for trends in packet sizes to determine if there was a significant difference from previous studies. All of this information aided us in developing a clearer picture of the traffic in the trace files and to identify changes in Internet use.

Another goal was to study the prevalence of selective acknowledgements (SACK). SACK is an extension to TCP that allows a host to acknowledge that it received a packet without acknowledging the packets that came before it [14]. Though there have been

studies that analyze the performance benefits of SACK, most of them used simulated data rather than actual packet traces from the Internet [12].

Explicit congestion notification (ECN) is another extension to TCP. Using ECN, a router can set 2 bits in IP's differentiated services field or 2 bits in TCP's reserved field to indicate that there is congestion on a network link. ECN is especially helpful when used in conjunction with an active queue management mechanism that allows a router to detect congestion before the queue overflows [18]. In [3], the authors measured the prevalence of ECN in packet traces and discovered that it was enabled in just 0.14% of the packets. Our goal was to confirm or refute their findings.

We also wanted to study the effects of non-responsive traffic on TCP performance. Non-responsive traffic is defined as traffic that does not respond to congestion control. UDP is non-responsive by default. During congested periods, it is possible that a surge of non-responsive traffic could significantly decrease the performance of TCP [19]. Our goal was to determine if there is evidence of this in the trace files we were given.

This paper analyzes Internet traffic from a cable broadband service provider. Motorola captured header packets by means of a packet sniffer installed at the ISP head-end, and they generated trace files with 'tcpdump'. Motorola generated traces both downstream to the client, and upstream to the Internet at large. We analyzed three gigabytes of packet traces spanning three consecutive hours on July 10, 2003. These traces contained 42,485,439 packets from 38,572 hosts transmitting 52,838,056,419 bytes. Our results quantify: (1) aggregate application-oriented capacity usage statistics, (2) the prevalence and usage impacts of TCP-SACK, (3) the prevalence of Explicit Congestion Notification (ECN), (4) the effects of non-responsive (UDP) traffic on TCP, and (5) proper sampling sizes and techniques. Based on trends in these quantifications and results we are able to make optimization recommendations both on the hardware level (cable modems) and on the software level (TCP). Overall, we present important implications for Motorola's broadband department.

The paper is organized in 5 sections. Section 2 contains additional background and information on previous work that is related to this project. Section 3 outlines packet trace files that we analyzed, the available tools that we used in our data analysis, and the custom tools developed to extract relevant data from the trace files. Section 4 is a detailed results section and contains the bulk of our analysis. Finally, section 5 features the conclusions that we drew from our analysis, as well as the implications of our study.

2 Related Work

Many studies have used real-world packet captures to discover trends and changes in Internet traffic. Likewise, there have been studies that attempted to identify performance-enhancing changes that can be made to the TCP/IP suite and broadband networks. These studies have made important contributions to our own work. In this section, we compare our work to some of these studies and list the points that make our work unique.

2.1 Characterization of Internet Traffic

One such study, conducted by Sharad Jaiswal, et. al., formulated a passive methodology that infers the sender's congestion window by noting TCP segments that pass through a measurement point [3]. As in our work, this study gathered its data from real-world network connections rather than generated by a simulation. In the case of this study, the source was a Tier-1 network provider, which provided them with over 10 million connections. Using this data, the team was able to develop methods for estimating the sender's congestion window and round-trip time (RTT). They found that: (1) a major cause of sender throughput limitation is often the lack of data to send, instead of network congestion and (2) connections do not usually experience large RTT variations throughout their lifetimes.

The estimation methods derived in [3] are similar to the ones used by the tcptrace utility, which was used in our research. Perhaps the most important difference between this study and ours is that Jaiswal et al were primarily concerned with measuring TCP performance, not improving it. Our project focuses more on characterizing trends in network traffic that may lead to hardware or software improvements. This study provided us important background information regarding TCP characteristics that aided in our understanding of RTTs and general TCP behavior.

In [4], Felix Hernandez-Campos, Kevin Jeffay, and F.Donelson Smith studied trends in web traffic using nearly 1 terabyte of packet traces that were collected in 1999, 2001 and 2003 from the University of North Carolina at Chapel Hill's link to its Internet service provider. In addition, their results were compared to smaller but similar analyses by other researchers spanning the 1995-1998 timeframe. The authors noticed an increase in the complexity of web pages as well as the size of requests.

As in our study, the authors of [4] worked with anonymized capture data, and were interested in finding out how small a sample duration could be and still maintain a high level of accuracy. Of particular interest to us were the conclusions they reached regarding sampling sizes. The researchers found that their 90-second traces produced results that were virtually indistinguishable from the 4-hour traces. Though our study focused on more than one class of Internet application, the methods they used to derive the proper sample sizes were generic enough for our use as well.

Karthik Lakshminarayanan and Venkata Padmanabhan [5] focused on the growth of peer-to-peer networking applications. These applications exist primarily to allow their users to share large media files with other peers. The team found that the bandwidth usage of broadband hosts is asymmetric with downstream bandwidth usage approaching five times the level seen on the upstream. Like our work, the researchers focused on broadband hosts but they limited their work to a single class of applications. The methodology used in this effort is significantly different than ours. As opposed to using hardware to passively collect data, they installed measurement agents on residual hosts. They gathered various data including TCP throughput, ping, packet-pair and traceroute measurements from various vantage points.

In another study, Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy [2], took interest in the growing popularity of peer-to-peer networking. Using packets traces gathered from May 28, 2002 to June 6, 2002 at the University of Washington, the researchers found that peer-to-peer applications have eclipsed the World Wide Web as the leading use of the Internet. Further, they found that Kazaa, a popular P2P file sharing program, alone accounts for more than a third of all TCP traffic and that the majority of peer-to-peer traffic is generated by a small number of hosts. They also posed suggestions for improving the performance of peer-to-peer applications including a proxy cache to store frequently downloaded objects. Though very similar to ours, this study did little to analyze the behavior of the flows and instead focused on statistical data.

2.2 TCP-SACK

Jitendra Padhye and Sally Floyd's study [6] took interest in the deployment of various congestion control algorithms and TCP options, including ECN and SACK. The outcome of this project was TBIT, a tool that analyzes the behavior of the TCP implementations found on public web servers. TBIT can check any web server in a non-disruptive manner without any special privileges. Our project differs in that we analyzed data that was captured between two points, as opposed to querying web servers. An understanding of this project provided us additional insight as to how ECN, SACK and other TCP-related options should operate.

Sally Floyd also participated in [12], a study of the performance increase SACK provides. In this study, the authors used simulated data rather than packet traces from the Internet to reach their conclusions. They compare the time needed by various TCP implementations to recover from dropped packets and conclude that without SACK, TCP performance suffers because it can only retransmit one dropped packet per round-trip time.

2.3 ECN

Sally Floyd explored the performance implications of ECN in [20] using simulated packet traces. In this study, a random early detection (RED) gateway was used to set the ECN bits. Floyd discovered that ECN does give a modest performance gain to TCP/IP but cautions that the study should only be viewed as "a preliminary investigation" of ECN. Floyd also concludes that the main advantage of ECN is that it decreases packet drops, which benefits connections that depend on the timely arrival of data.

The authors of [21] arrived at conclusions that were similar to Floyd's. Using a small experimental test bed, they compared the performance of TCP connections using ECN in conjunction with RED against that of TCP connections without ECN. The authors found that by using ECN, their experimental hosts were able to download a 20 megabyte file 14 seconds faster than they were able to without ECN. Further, they found that a single ECN-capable host had a negative effect on the performance on non-ECN hosts on the same network.

2.4 Effects of Non-Responsive Traffic on TCP

In [22] Sarah Joyce was interested the effect unresponsive traffic would have on TCP traffic. She chose to investigate how network gaming over UDP impacted the overall network performance. The authors ran simulations at two different measurement sites where network games were played. One is located in Auckland University in New Zealand and the other in The New Zealand Internet Exchange. The particular games that were considered were Quake World and Unreal Tournament.

The results of her simulation suggested that the network games did in fact restrict the throughput of TCP on a heavily loaded link. She concluded that the effect of UDP on a link is comparable to putting a "cap" on a network. Voice over IP applications had similar results, which suggested that unresponsive traffic brings the same negative results regardless of the application.

We used a different methodology to measure unresponsive traffic's impact on TCP traffic. We isolated periods in the trace where unresponsive traffic was high and periods where unresponsive traffic was low. We then compared these two groups using average round-trip-time (RTT) per flow and packet retransmissions per flow as performance metrics.

3 Tools Used in Analysis

During the course of this study we used three open source tools and several custom scripts to gather information from the trace files. In this section, we list these tools and explain how we used them.

The data that was used in this study was obtained using the process described in [7]. In summary, Motorola designed a Capture System for Internet Subscriber Traffic (CSIST) to capture packet headers of IP traffic on a Hybrid Fiber Coax (HFC) cable network. This system is capable of passively collecting and monitoring IP sessions from one 100baseT branch of a Hybrid Fiber Coax network.

The system's software performs packet collection, packet sanitization, trace file compression, and trace direction combination. The tool tcpdump [17] was used to collect the packets for the system. Tcpdump uses the libpcap [17] library, a system-independent, open-source interface for user-level packet capturing. The tool tcpdpriv [8] was then used to sanitize the collected packets. The data sanitization includes anonymizing the IP addresses, overwriting the MAC addresses and removing other private packet information.

The captured headers themselves consist of the first 100 bytes of each packet. This is a sufficient size to capture the IP header, the transport layer header, and all of the pertinent header option fields. The packet's actual payload is not captured. The IP addresses map to unique, anonymous addresses, and the network port numbers are conserved.

The capture files used in this project were collected by one of Motorola's clients and represented three consecutive hours worth of TCP headers. We received two files from

Motorola: one containing packets from the upstream link and the other containing packets from the downstream link. We used ipsumdump [9], a tool written by Eddie Kohler of the International Computer Science Institution, to summarize the libpcap files into a more easily read combination file where both the upstream and the downstream packet traces were combined sorted by timestamp.

As explained previously, the capture files were generated using tcpdump. Tcpdump is an open source project that uses the libpcap library to capture packets. It also provides a flexible filtering syntax that allowed us to reduce the capture files to smaller, more manageable sizes. Its filtering ability is somewhat limited because it cannot detect TCP options.

Ethereal is an application that provides a graphical interface for viewing packet capture files. It also has a more advanced filtering language than tcpdump, allowing the user to filter based on a wide variety of TCP and application layer protocol options. Ethereal can create simple time-sequence, RTT, and throughput graphs that enable the user to quickly visualize the packets in the capture file. All of these features proved useful to us when identifying periods of data within the capture files that were of particular interest.

Of all the tools used during this project, tcptrace was perhaps the most useful and versatile. Developed by Shawn Ostermann at Ohio University, tcptrace recreates the events that occurred during a connection and gathers a plethora of useful statistics.

Especially helpful was tcptrace's ability to gather RTT statistics. This helped to identify periods in the capture files during which congestion was higher than usual. It was necessary to find these periods in order to study the effects of non-responsive traffic on TCP.

3.1 Custom Tools

In addition to pre-existing programs, we also found it necessary to develop custom programs. These scripts interfaced with the libpcap library in order to read and manipulate the capture files. Because they use libpcap, all of the programs can take advantage of the same filter syntax used by tcpdump. Our programs were developed for two purposes: to filter capture files or to gather statistics about the connections in a capture file.

To better collect information about SACK and ECN using connections, we wrote a program that adds the ability to filter capture files by TCP options. Libpcap's filtering syntax does not allow for this. We also wrote custom programs to perform simple statistical analyses on the capture files such as counting the number of hosts and connections.

4 Results

In this section we present and explain the results of the various analyses conducted on the packet trace files. These analyses fall into four major categories: the characterization of

traffic found in the files, the prevalence and performance of TCP-SACK, the prevalence of ECN, and the effects of non-responsive traffic on TCP performance.

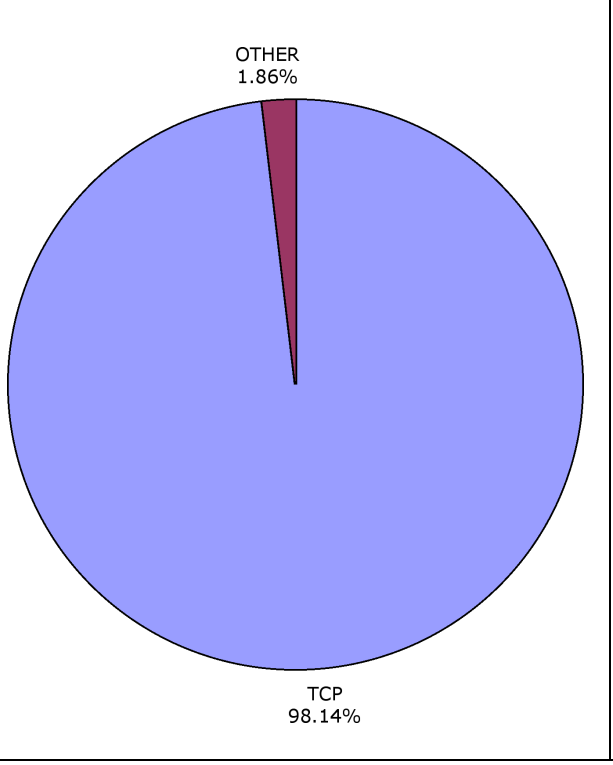
4.1.1 Characterization of Traffic

Before proceeding with more specific analysis, we first found it necessary and beneficial to gather aggregate statistics from the capture files to determine the make-up of the traffic. To obtain a broad image of the traffic, we found which transport- and application- layer protocols were most prevalent and gathered statistics on the packets themselves.

4.1.1.1 Transport Protocols

We first developed a simple tool to total the number of bytes transmitted by each transport protocol encountered. This program extracts the transport protocol number found in the IP header and adds the number of bytes in the packet to that protocol's total. Figure 1 shows a chart of this program's output when run over the entire capture file. As expected, TCP dominates among transport protocols, accounting for 98.14% of all the bytes transferred. UDP, the next most popular protocol, accounts for another 1.74%, while ICMP, GRE, ESP, and OSPFIGP combine for 0.12% of the total bytes transmitted.

Of the final four transport protocols listed ICMP is the most well-known. The Internet Control Message Protocol is used for troubleshooting and error messages. General Routing Encapsulation (GRE), also called IP tunneling, details how to encapsulate an IP packet inside another one. IP tunnels are frequently used to access private LANs from remote locations. Encapsulate Security Payload (ESP) is a security used by IPv6 and Open Shortest Path First Internal Gateway Protocol (OSPFIGP) is a routing protocol designed specifically for use on the Internets.



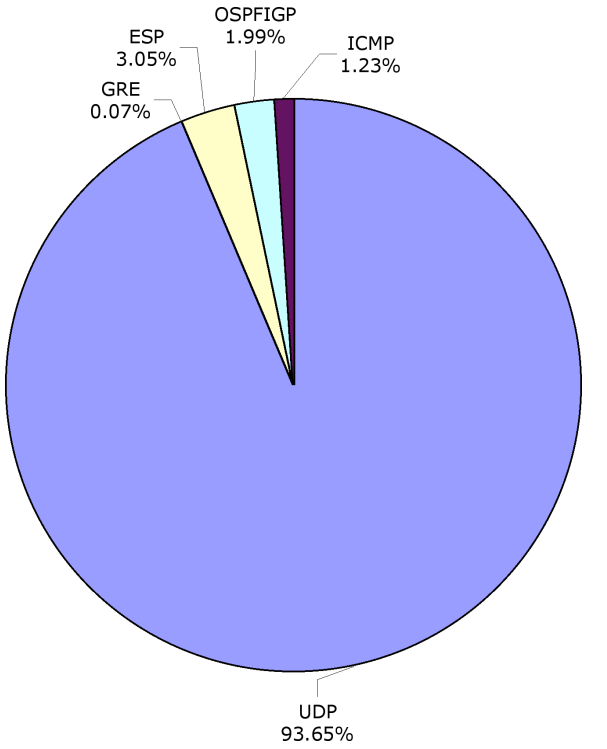


Figure 1: The graph on the left compares the number of bytes transmitted using TCP to the number of bytes transmitted by all other transport protocols. The graph on the right shows which protocols are contained in OTHER and in what portions.

4.1.1.2Application Protocols

Prior studies [2] have shown that peer-to-peer file sharing applications such as Kazaa are now the primary consumers of capacity on some parts of the Internet. To measure peer-to-peer traffic for a broadband ISP, we wrote a tool to total the number of bytes sent to and received by each port number. A table showing the top 50 capacity-using ports, sorted by total bytes transmitted, is shown in Figure 2, with peer-to-peer applications bolded. Much of this data was taken from [11]. It should be noted that it is difficult to say for certain which applications are using these ports without examining the payload data. Nonetheless, the presence of five ports that are used by well-known file sharing applications lends credible evidence to the claim that peer-to-peer programs are responsible for a significant portion of Internet traffic.

Interestingly, port 80, most frequently used by WWW servers, still accounts for more traffic than any other port despite its relatively small document size. This is due mainly to the sheer number of connections it services. It can still be said that the WWW is the most popular use of the Internet, but file-sharing applications are not far behind.

Rank	Port	Application	Bytes	Bytes To	Bytes From	Number of Flows
1	80	WWW	2,373,960,752	207,631,342	2,166,329,410	107,035
2	1214	**Kazaa**	2,298,749,546	1,537,815,918	760,933,628	34,636
3	6882	**BitTorrent**	1,737,493,411	695,768,909	1041,724,502	3,991
4	6699	**WinMX**	1,048,744,327	677,183,431	371,560,896	3,301
5	4662	**Emule**	922,462,874	551,699,262	370,763,612	40,360
6	3776	Device Provisioning Protocol	840,727,100	816,550,900	24,176,200	429
7	20	FTP data	684,783,212	20,381,119	664,402,093	895
8	3110	sim-control	664,033,874	23,126,686	640,907,188	925
9	9230	*Unknown*	652,797,894	18,348,852	634,449,042	183
10	3967	*Unknown*	651,679,984	19,613,854	632,066,130	696
11	2330	TSCCHAT	648,052,112	20,772,763	627,279,349	717
12	6881	**BitTorrent**	627,016,039	122,190,825	504,825,214	5,811
13	6883	**BitTorrent**	620,646,812	127,427,494	493,219,318	1,471
14	6884	**BitTorrent**	552,405,478	100,987,050	451,418,428	528
15	2615	Firepower	536,476,266	18,314,407	518,161,859	1,399
16	2201	Advanced Training System	461,714,688	45,192,767	416,521,921	2, 741
17	2025	Xribs	460,941,444	255,796,906	205,144,538	296
18	411	**Direct Connect /** Remote MT Protocol	411,559,535	386,305,490	25,254,045	1176
19	3623	*Unknown*	381,333,901	59,681,285	321,652,616	4040
20	1800	ANSYS License Manager	362,159,594	317,387,346	44,772,248	1539

Figure 2: This table shows the top 50 capacity-using ports. Bolded labels in the Application field indicate a file-sharing application. Unidentifiable applications are italicized. The top 50 ports are found in Appendix A [11]

4.1.1.3 Packet Size

We next graphed the cumulative distribution function (CDF) of packet sizes (shown in Figure 3) using all of the packets in the capture files. This data was gathered by another custom program that recorded the size of each packet in the capture files. The purpose of this experiment was to determine if the rise broadband deployment and the corresponding increase in large-file transfers had led to an increase in packet sizes as well. We found the average packet size to be approximately 619.89 bytes. The most common packet size was 54 bytes, which corresponds to an empty ACK with TCP, IP, and Ethernet headers. The next most common size was 1514 bytes, which corresponds to Ethernet's maximum transfer unit plus the Ethernet header. The largest packet size encountered was 2062 bytes. There were 80 such packets, accounting for less than 0.0002% of all packets.

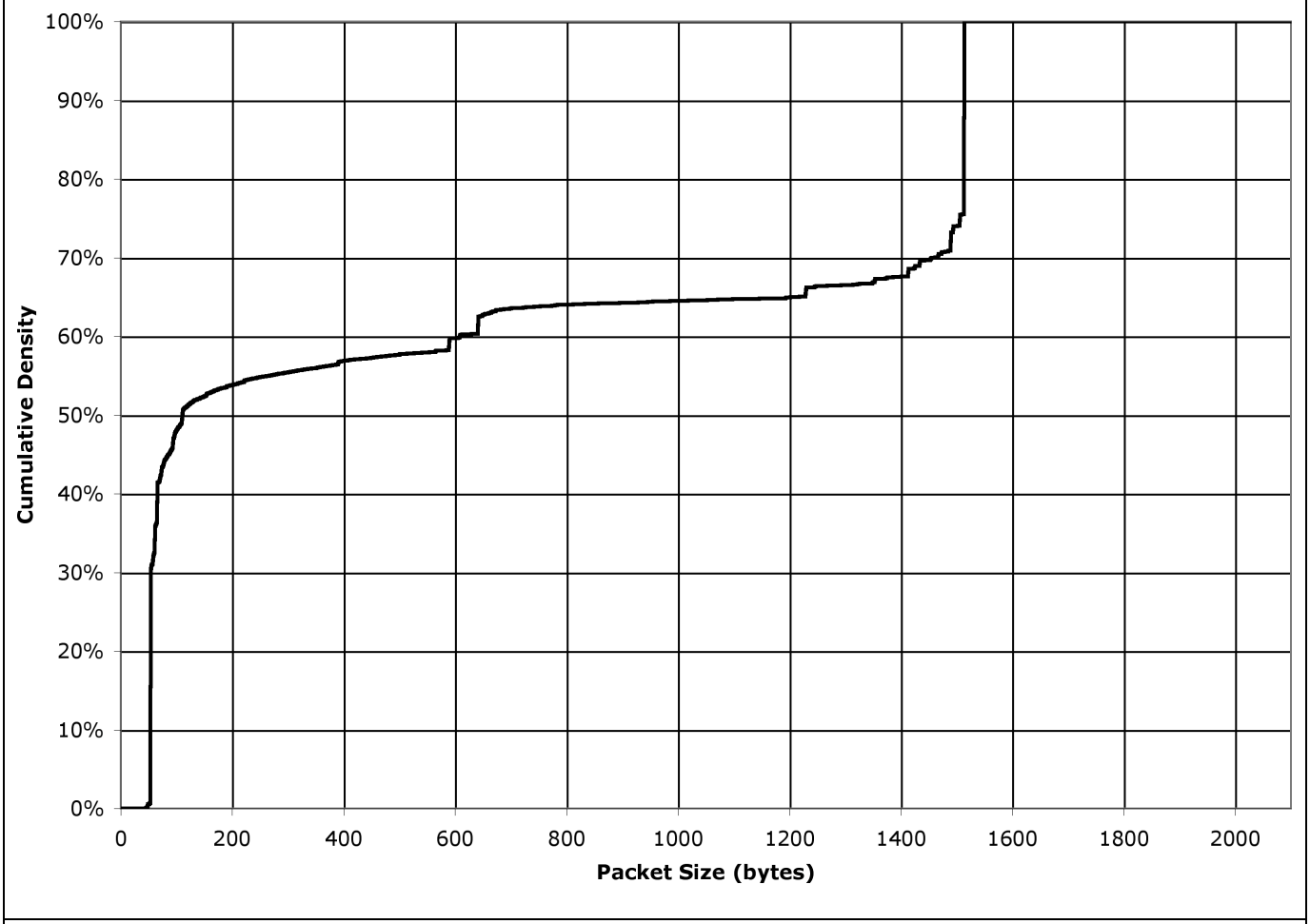


Figure 3: This graph shows the cumulative distribution function of packet sizes.

The results we obtained by measuring packet sizes was compared to those obtained at NASA's Ames Internet Exchange (AIX). The CDF of their results is shown in Figure 4. Interestingly, AIX actually saw larger packets with significant spikes at approximately 600 and 1500 bytes. There are similarities nonetheless. Two of the three most common packet sizes seen at AIX are 40 and 1500 bytes. Once the 14-byte Ethernet header is

stripped, these equal the two most common packet sizes in the capture files (54 bytes and 1514 bytes, respectively).

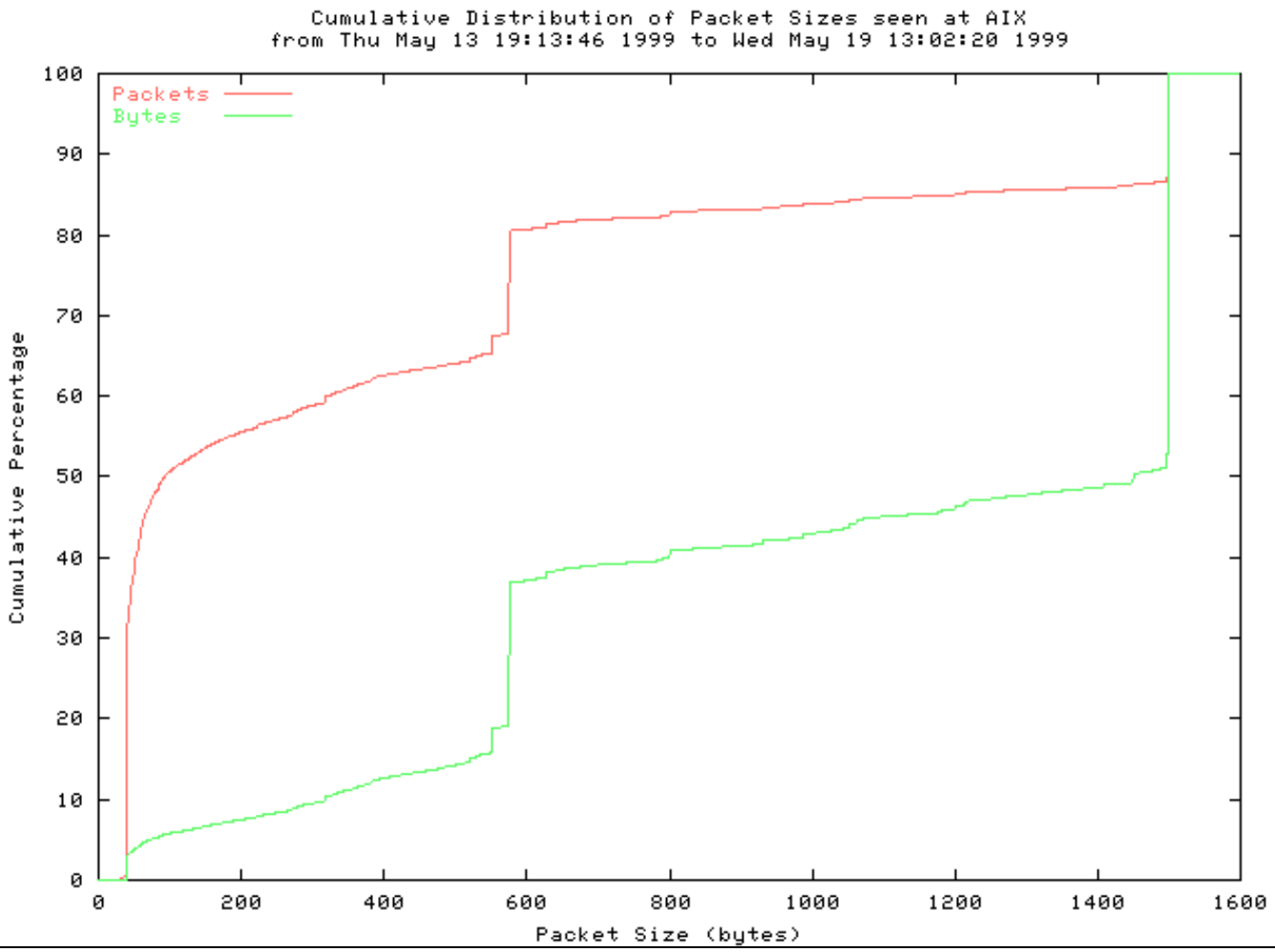


Figure 4: The CDF of packet sizes seen at NASA AIX on Wednesday May 19, 1999 [15].

There is one significant difference between our results and those obtained at AIX. The average size of the packets encountered at AIX was 413 bytes [15], which is more than 200 bytes smaller than the 619.89 we found in the capture files.

4.2 TCP-SACK

Selective acknowledgement (SACK) is an extension to the TCP protocol that allows a host to acknowledge received segments without acknowledging the segments that arrived before it. This is beneficial on congested networks where multiple segments may be lost from a single data window. The presence of SACK can be determined from the initial TCP handshake. A host that supports SACK will put the two-byte "SACK-permitted" option in the header of its SYN segment [14].

TCP implementations without SACK can only retransmit one segment for each round-trip time or they risk retransmitting data that has already been received. TCP

implementations with SACK know which segments have been received and thus are able to retransmit only the segments that were dropped [14]. SACK implementations were shown to provide significant performance gains over non-SACK implementations in simulations [12]. However, there has been little research on its real-world benefits. For this reason, we chose to examine both the prevalence of TCP-SACK implementations and the performance of the hosts that implement it.

The Windows operating system has shipped with a SACK implementation since Windows 98; the Linux kernel has since version 2.1.90. Likewise, most server operating systems including the BSDs, Solaris, Digital Unix, and the aforementioned Linux kernel also ship with or can be configured to use TCP SACK [13].

To gather data on the prevalence of SACK, we wrote a tool that iterates over the entire trace file. This tool looks for one of two things: the SACK-permitted option in a SYN packet or the SACK-block option in any other packet. If either of these is found, the host is marked as a SACK-enabled host. This information is reflected in the results, which show that a large portion of Internet hosts transmit the SACK-permitted bit in their SYN segments (Figure 5 and Figure **6**). As expected, the majority of the hosts are SACK-enabled. TCP-SACK is less common on upstream hosts, perhaps because server administrators are reluctant to deploy non-standard TCP implementations.

With its prevalence established, we next looked at the advantages of TCP-SACK. As previously mentioned, simulation-based tests have shown that the use of selective acknowledgments is quite beneficial. To measure its real-world benefit, we decided to compare the fraction of packets retransmitted by SACK-enabled flows and non-SACK-enabled flows during a thirty-minute period selected at random. This period represents minutes 50 through 80 in the trace file. All TCP flows that sent a packet during this period, whether the complete or incomplete, were taken into account when gathering this data. A graph of the CDF of these retransmission rates can be found in Figure 7. The graph shows that flows that are not SACK-enabled suffer from higher retransmission rates. This data corroborates the conclusions of previous studies [12].

Trace File	SACK-enabled Hosts	Not SACK-enabled Hosts
Upstream	29432	3138
Downstream	945	27
Total	30377	3165

Figure 5: This table shows the number of hosts that are SACK-enabled and the number that are non SACK-enabled. This data was gathered from the upstream and downstream trace files. The total row is the sum of the first two.

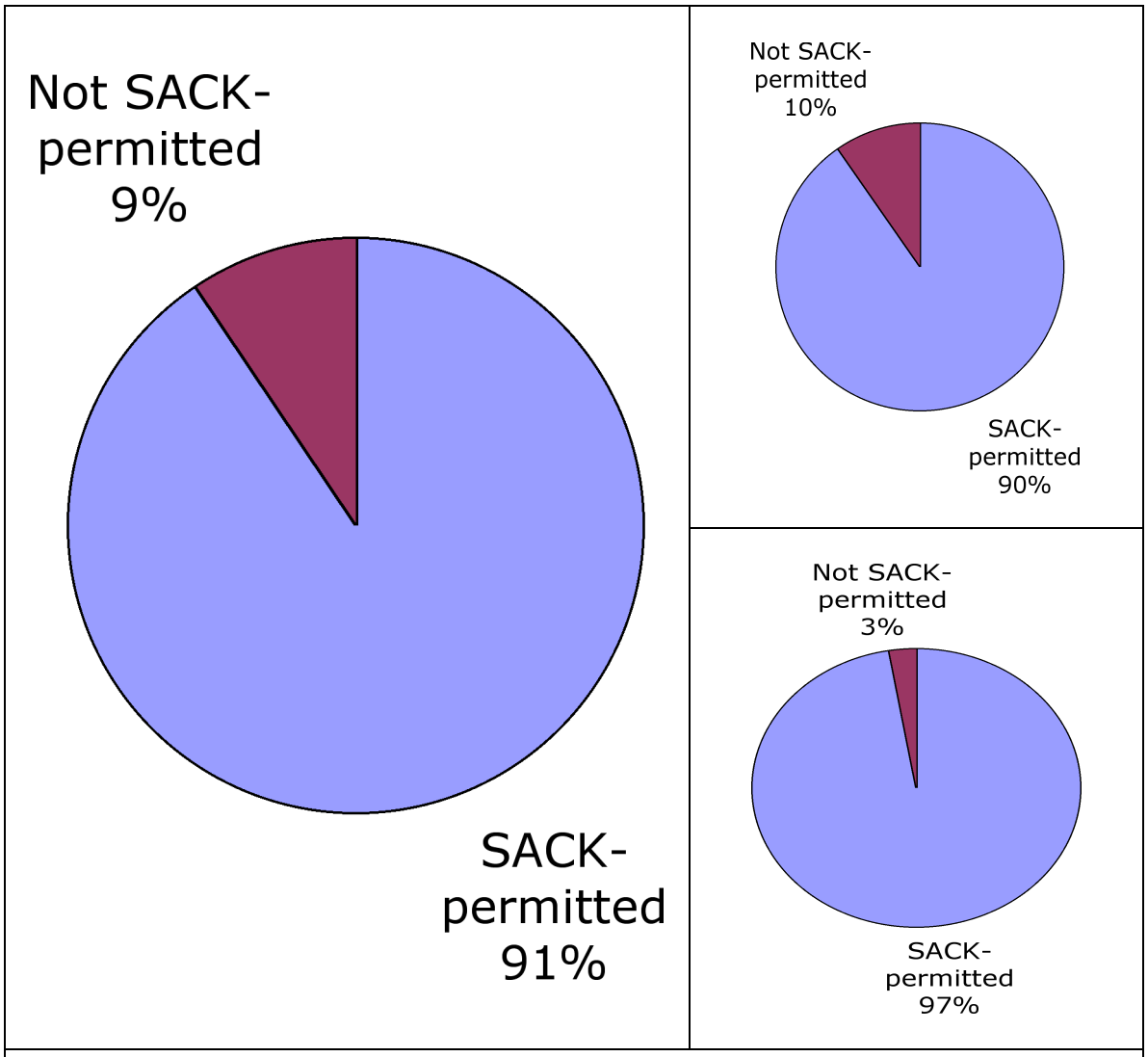


Figure 6: Prevalence of TCP-SACK by host. Only SYN-sending hosts were counted. The graph on the left shows SACK prevalence for all hosts in the trace files. Upstream hosts are on the top right and downstream hosts are on the bottom right.

4.3 ECN

In the IP header, there are two bits that correlate to ECN – the CE and the ECT bits. If either one of these bits is set, (but not the other – an XOR), then that host is capable of recognizing and using ECN. If both bits are set, then that particular host is encountering congestion, and would like to use ECN. Hosts on both sides must support ECN in order for the protocol to have any effects. The use of ECN is nearly non-existent in the data we were given. Only 7 of the 38,572 unique hosts in the trace files, or 0.018%, set the ECN capable bit. Since the number of ECN capable hosts is so close to 0%, it is safe to say that ECN simply isn't being used.

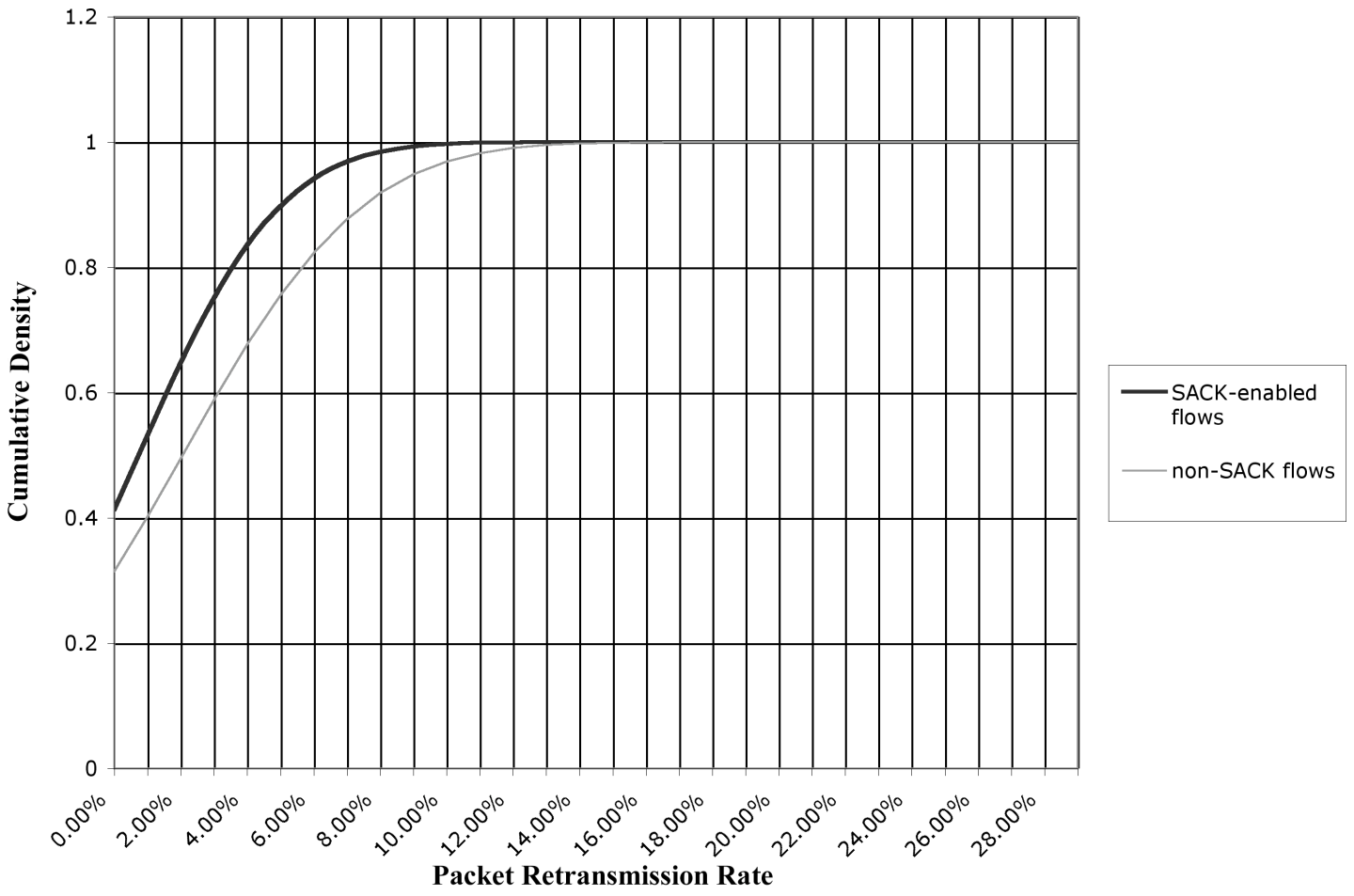


Figure 7: The CDF of packet retransmissions per TCP flow. This chart shows that TCP-SACK provides a significant advantage to flows the implement it.

4.4 Effects of non-responsive traffic on TCP

Non-responsive traffic refers to traffic that does not have any sort of congestion control mechanism built in. For our purposes we assume that all UDP traffic is non-responsive. This traffic could pose a threat to responsive traffic (i.e., traffic that does make use of some congestion control mechanism) on heavily congested network links. We identified several brief periods in which non-responsive traffic capacity spiked and attempted to determine what effect this had on responsive traffic.

On average, TCP traffic made up 98.14% of the total bandwidth usage. This left 1.87% for non-TCP traffic, which consists mostly of UDP packets. We identified three thirty-second periods during which non-TCP traffic accounted for less than 1.00% of all traffic and three thirty-second periods during which non-TCP traffic made up 5.00% or more of the total traffic. We then compared these two groups using average round-trip-time (RTT) per flow and packet retransmissions per flow as performance metrics. Tcptrace

was used to gather both RTT data and packet retransmission data. The results of our analysis can be seen in Figure 8 and Figure 9.

From the data we have analyzed, it appears as though non-responsive traffic does have a detrimental effect on TCP traffic. Both the average RTT and the retransmission rate per TCP flow suffer when non-responsive traffic peaks. However, due to the small sample size of just three minutes, it is difficult to draw any firm conclusions. Increasing the sample size was not possible due to the limited amount of non-responsive traffic in the trace files. If we were to increase the sample size, the percent of non-TCP traffic would drop well below the 5.00% mark we set as "high".

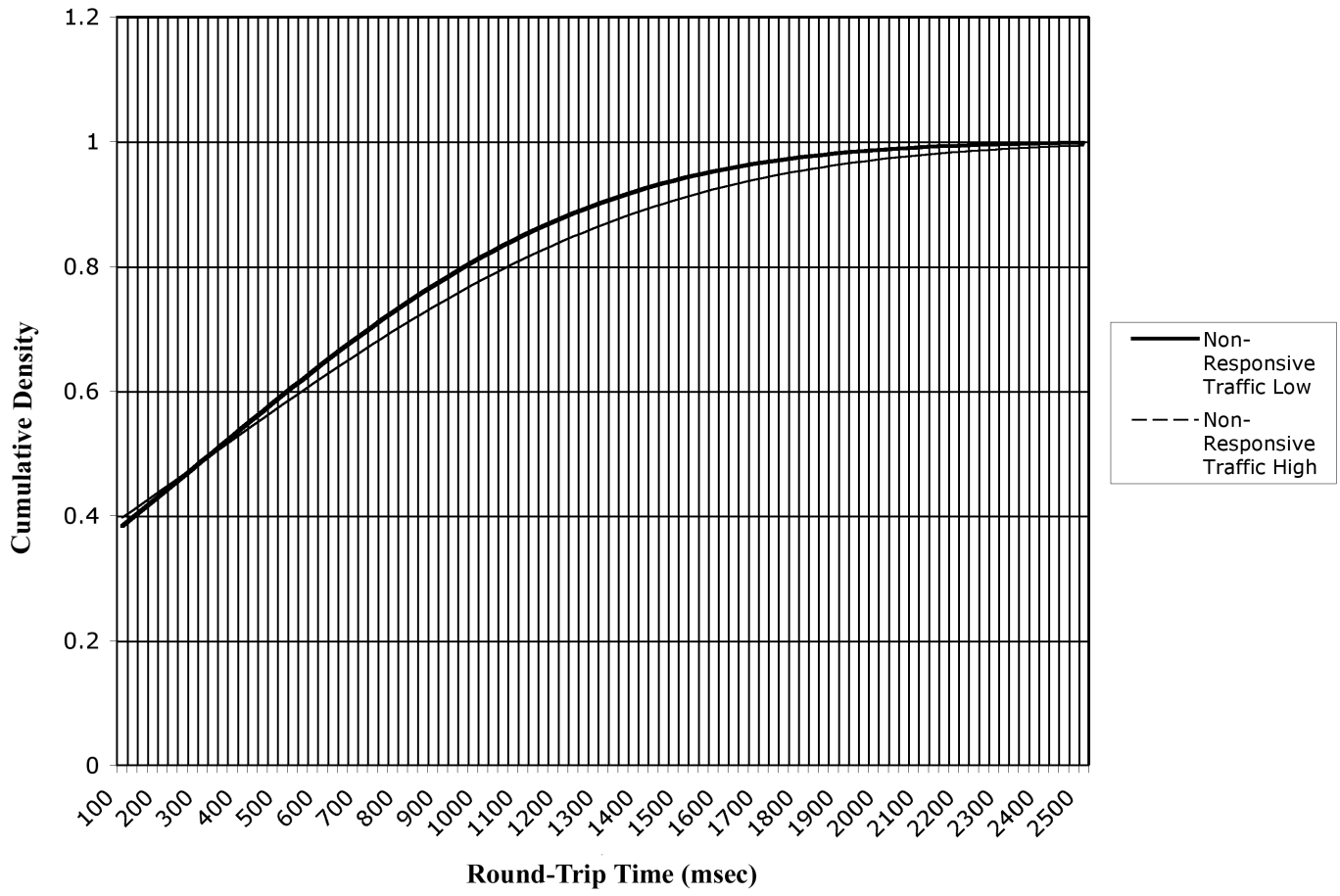


Figure 8: This graph shows the CDF of the average RTT per TCP flow. The average RTT was found for each flow using the tcptrace utility.

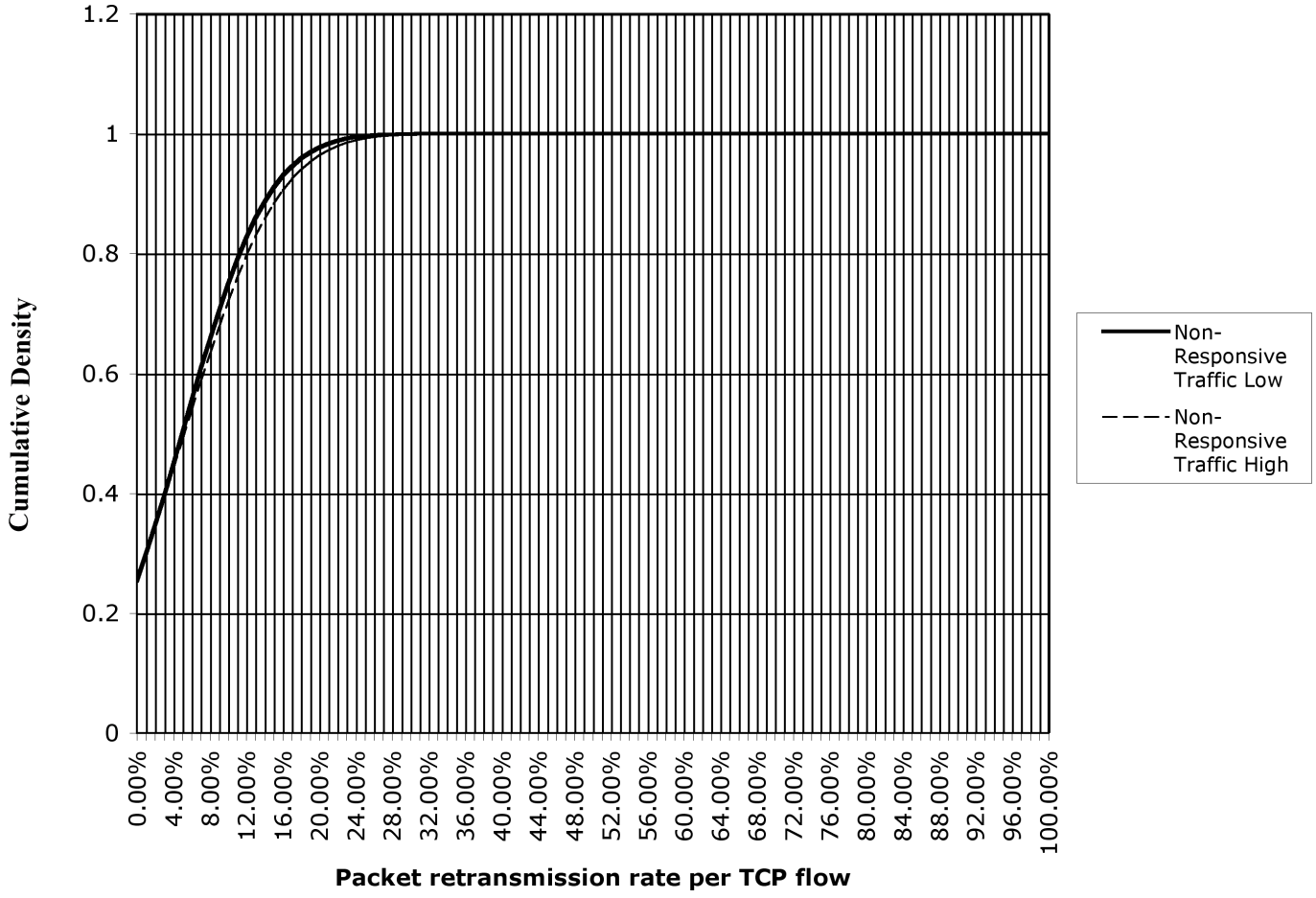


Figure 9: The CDF for packet retransmission rate per TCP flow. This graph shows little difference between periods in which non-responsive traffic is at its highest and periods in which it is at its lowest.

5 Sampling Issues

When measuring connection-based statistics, we found that it was difficult to extract data from each of the 915,026 flows in the capture files given our resources. To expedite our analyses we determined exactly what portion of the trace files would need to be examined in order to obtain accurate results. The authors of [4] performed a similar procedure and determined that 90 seconds worth of traces would yield accurate results provided there were enough sample values. Unfortunately, their traces were captured on a more congested network at various times from 1995 to 2003. Though we could not use their results, we were able to implement their methods to obtain our own.

To determine the proper sample size for our SACK performance analysis, we split the traces into 3 files containing approximately 1 hour each. The second of these files was chosen and split into 2 files each with 30 minutes of data. These were then split into 4 files with 15 minutes in each. We performed our SACK analysis on 2 of the 15-minute

files taking 1 from each of the 30 minute traces and averaged the results. The analysis was also performed on each of the 30-minute traces and, again, the results were averaged. Finally, the analysis was performed on the 1-hour long trace file. The results of these tests (Figure 10) show a significant difference between the retransmission rates of the 15-minute samples and the 30-minute samples, but a much smaller difference between the 30-minute samples and the 1-hour sample. From these tests, we were able to determine that a single, 30-minute trace file was sufficient to gather statistics regarding the performance benefits of SACK.

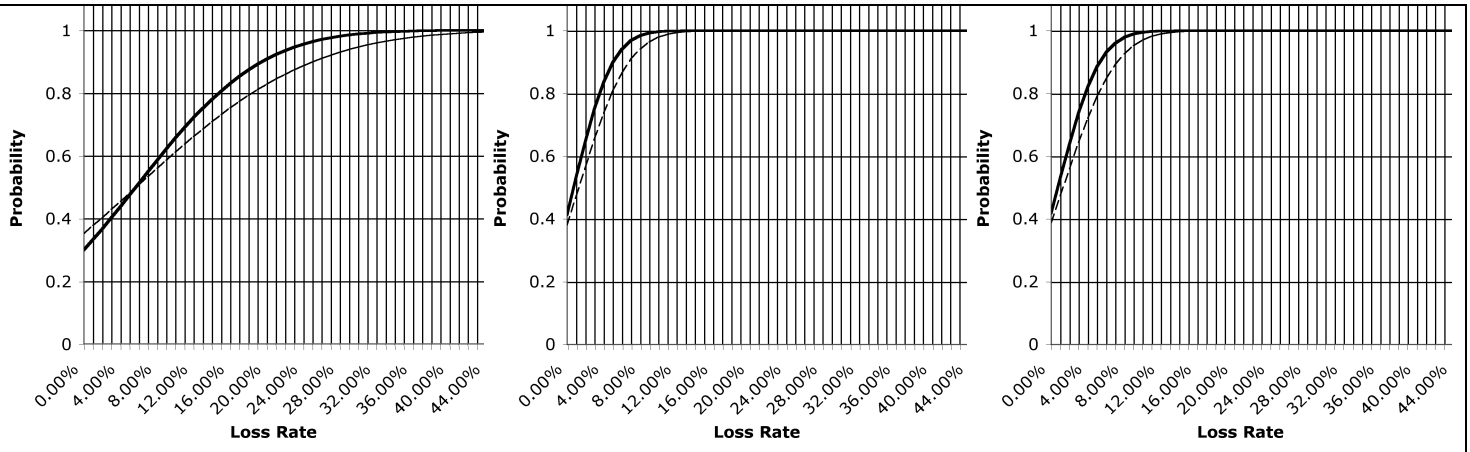


Figure 10: The results of our sampling procedure for SACK. On the left is the average of the 15-minute traces. In the middle is the average of the 30-minute traces. On the right is the 1-hour trace. Note that the difference between the curves of the 30-minute and 1-hour trace is minimal.

6 Conclusions

Based on our results, we conclude that Internet traffic is indeed changing. Though we found the WWW to be the most popular application on the Internet, in terms of both flows and bytes transmitted, peer-to-peer file-sharing utilities have made significant gains. In fact, when totaled, peer-to-peer applications use far more bandwidth than does the web and they use fewer connections to do it. In addition, peer-to-peer applications send nearly as much data as they receive.

Unfortunately, we don't have trace files taken from different times and congestion periods to compare to, so it is difficult to say whether or not file-sharing utilities overtax the upstream bandwidth of their users. Nonetheless, we consider this a likely scenario. One possible solution to this problem is to increase the amount of upstream bandwidth available to the users of cable modems. There are several ways to do this, the simplest of which is to increase the number of upstream carriers.

We were able to detect a slight detrimental effect that non-responsive traffic had on TCP in the trace files we examined. It is likely that this effect will increase during times of greater congestion. Thus, it may be beneficial to limit the rate at which non-responsive traffic is transmitted by the cable modem during periods of high congestion. This would

increase the performance of TCP at the expense of applications that use UDP, which makes up the majority of non-responsive traffic.

Improvements can be made to TCP as well. Our statistics show a wide adoption rate for SACK, however the same cannot be said of ECN, an extension to TCP with similar performance implications [16]. Unfortunately, ECN can cause problems when used with older routers. Still, we encourage the deployment of ECN as well as the continued deployment of SACK as both are capable of increasing TCP performance on congested networks.

7 Future Work

The original proposal for this project mentioned several areas of interest that we were not able to investigate during the course of our project. We planned to study the effects of ACK compression on TCP flows, however we were unable to positively identify any occurrences of ACK compression. Likewise, we could not detect any uses of TCP performance enhancing proxies (PEPs). PEPs are often used to help relieve the effects of ACK compression but they are difficult to detect because they do not alter the packets they encounter in any way. Both of these topics warrant further study, perhaps with the aid of a simulator to eliminate the need to identify them in capture files.

The analyses mentioned above would provide more insight into the behavior of TCP traffic on the Internet. Further study on these subjects as well as the subjects we investigated would be beneficial both to Motorola and to the Internet itself.

References

[1] Horrigan, J.. Broadband Adoption at Home: A Pew Internet Project Data Memo, May 2003, http://www.pewinternet.org/reports/toc.asp?Report=90

[2] Saroiu, S., Gummadi, K., Dunn, R., Gribble, S., and Levy, H.. An Analysis of Internet Content Delivery Systems, In Proceedings of the 5th USENIX Operating Systems Design and Implementation (OSDI), Boston, MA, USA, October 2002.

[3] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., and Towsley, D. Inferring TCP Connection Characteristics Through Passive Measurements. In: To appear in IEEE Infocom. Hong Kong. March 2004, http://ipmon.sprintlabs.com/pubs_trs/trs/tcp-characteristics.pdf

[4] Hernandez-Campos, F., Jeffay, K., and Smith, F.. Tracing the Evolution of the Web Traffic: 1995-2003, In Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Orlando, FL, October 2003, http://www.cs.unc.edu/Research/dirt/abstracts/MASCOTS-03a-abs.html

[5] Lakshminarayanan, K., Padmanabhan, V.. Some Findings on the Network Performance of Broadband Hosts, In Proceedings of the Internet Measurement

Conference (IMC), Miami, FL, USA, October 2003, http://www.icir.org/vern/imc-2003/papers/p301-lakshminarayanan1.pdf

[6] Padhye, J., Floyd, S. On Inferring TCP Behavior, ACM SIGCOMM, San Diego, California, USA, 2001, http://www.acm.org/sigs/sigcomm/sigcomm2001/p23-padhye.pdf

[7] Thomas, J., Cudak, M. Capture System for Internet Subscriber Traffic (CSIST). Motorola Labs, Communications Research Laboratory. April, 2003

[8] Minshall, G.. Tcpdprive Source and Documentation, http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html

[9] Kohler, E. Ipsumdump Source and Documentation, http://www.icir.org/kohler/ipsumdump/

[10] Ostermann, S. Tcptrace Manual, http://www.tcptrace.com/manual/index.html

[11] Internet Assigned Numbers Authority. Port Numbers, http://www.iana.org/assignments/port-numbers

[12] Fall, K. and Floyd, S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP, 1995, http://www.aciri.org/floyd/papers/sacks.pdf

[13] Pittsburgh Supercomputing Center. Experimental TCP Selective Acknowledgement Implementations, Feb. 2003, http://www.psc.edu/networking/all_sack.html

[14] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. RFC 2018: TCP Selective Acknowledgement Options, Oct. 1996, http://www.ietf.org/rfc/rfc2018.txt

[15] McCreary, S. Packet Length Distributions, Jun. 2002, http://www.caida.org/analysis/AIX/plen_hist/index.xml

[16] Pentikosis, K., Badr, H., and Kharmah, B. TCP with ECN: Performance Gains for Large Transfers. March, 2001, http://sunysb.edu/~kostas/art/tcpencLT.pdf

[17] Tcpdump and Libpcap Source and Documentation, http://www.tcpdump.org

[18] Ramakrishnan, K., Floyd, S., and Black, D. RFC 3168: Addition of Explicit Congestion Notification (ECN) to IP, Sep. 2001, http://www.ietf.org/rfc/rfc3168.txt

[19] Zhao, Z, Ametha, J. Darbha, S. and Reddy, A. A Method for Estimating Non-Responsive Traffic at a Router. Feb. 2002, http://tamu.edu/~reddy/pape...sigmetrics02.pdf

[20] Floyd, S. TCP and Explicit Congestion Notification. 1994. http://www-nrg.ee.lbl.gov/papers/tcp_ecn.4.pdf

[21] Pentikousis, K. Badr, H. and Kharmah, B. TCP with ECN: Performance Gains for Large Transfers. Mar. 2001, http://sunysb.edu/~kostas/art/tcpecnLT.pdf

[22] Joyce, S. Traffic of the Internet – Report, October 11, 2000, http://wand.cs.waikato.ac.nz/old/wand/publications/sarah-420.pdf

Appendix A

Rank	Port Number	Application	Bytes	Bytes To	Bytes From	Number of Flows
1	80	WWW	2,373,960,752	207,631,342	2,166,329,410	107,035
2	1214	**Kazaa**	2,298,749,546	1,537,815,918	760,933,628	34,636
3	6882	**BitTorrent**	1,737,493,411	695,768,909	1,041,724,502	3,991
4	6699	**WinMX**	1,048,744,327	677,183,431	371,560,896	3,301
5	4662	**EMule**	922,462,874	551,699,262	370,763,612	40,360
6	3776	Device Provisioning Protocol	840,727,100	816,550,900	24,176,200	429
7	20	FTP data	684,783,212	20,381,119	664,402,093	895
8	3110	sim-control	664,033,874	23,126,686	640,907,188	925
9	9230	*Unknown*	652,797,894	18,348,852	634,449,042	183
10	3967	*Unknown*	651,679,984	19,613,854	632,066,130	696
11	2330	TSCCHAT	648,052,112	20,772,763	627,279,349	717
12	6881	**BitTorrent**	627,016,039	122,190,825	504,825,214	5,811
13	6883	**BitTorrent**	620,646,812	127,427,494	493,219,318	1,471
14	6884	**BitTorrent**	552,405,478	100,987,050	451,418,428	528
15	2615	Firepower	536,476,266	18,314,407	518,161,859	1,399
16	2201	Advanced Training System	461,714,688	45,192,767	416,521,921	2,741
17	2025	Xribs	460,941,444	255,796,906	205,144,538	296
18	411	**Direct Connect / Remote MT Protocol**	411,559,535	386,305,490	25,254,045	1,176
19	3623	*Unknown*	381,333,901	59,681,285	321,652,616	4,040
20	1800	ANSYS License Manager	362,159,594	317,387,346	44,772,248	1,539
21	63332	*Unknown*	343,945,830	9,166,364	334,779,466	108
22	1657	netview-aix-7	302,290,053	13,993,439	288,296,614	461
23	2315	Precise Sft.	301,267,240	12,828,333	288,438,907	498
24	2198	*Unknown*	285,869,661	279,129,806	6,739,855	149
25	3166	*Unknown*	275,420,915	7,711,526	267,709,389	260
26	3269	Microsoft Global Catalog	261,477,098	11,166,659	250,310,439	1,441
27	27960	*Unknown*	254,116,988	126,426,719	127,690,269	1,345
28	3133	*Unknown*	229,120,192	6,304,441	222,815,751	193
29	2174	*Unknown*	227,093,857	90,084,047	137,009,810	226
30	6885	**BitTorrent**	226,482,541	68,540,748	157,941,793	604
31	2002	Globe	219,491,029	46,489,228	173,001,801	1,092
32	1414	IBM MQSeries	211,684,952	205,066,829	6,618,123	203
33	1618	Skytelnet	208,991,174	203,993,682	4,997,492	213
34	3403	*Unknown*	207,168,543	8,382,910	198,785,633	256
35	3847	*Unknown*	206,996,509	8,734,282	198,262,227	792
36	29908	*Unknown*	206,994,495	128,373,296	78,621,199	48
37	3276	Maxim ASICs	197,546,116	14,423,559	183,122,557	10,170

38	6886	**BitTorrent**	181,533,049	19,064,339	162,468,710	1,555
39	1255	*Unknown*	173,921,001	36,718,160	137,202,841	2,022
40	1191	*Unknown*	170,718,513	166,969,904	3,748,609	240
41	3187	*Unknown*	152,385,086	5,118,352	147,266,734	1,073
42	29907	*Unknown*	147,740,372	69,227,417	78,512,955	42
43	3020	CIFS	144,878,323	137,639,078	7,239,245	308
44	65325	*Unknown*	143,383,066	139,572,937	3,810,129	40
45	1150	*Unknown*	142,306,785	8,887,570	133,419,215	385
46	2310	SD Client	140,420,571	53,916,414	86,504,157	3,911
47	2493	Talarian MQS	138,745,900	3,948,870	134,797,030	148
48	4366	*Unknown*	136,583,928	132,128,553	4,455,375	158
49	2604	NSC CCS	136,427,460	4,436,001	131,991,459	147
50	29128	*Unknown*	136,344,617	132,174,841	4,169,776	20

Figure 11: This table shows the top 50 capacity-using ports. Bolded labels in the Application field indicate a file-sharing application. Unidentifiable applications are italicized [11]