Internet Traffic Characterization

Teunis J. Ott Larry H. Wong

November 17, 1999

Abstract

This is a report on, and an analysis of, IP traffic measurements taken at an ISP which is a Telcordia customer. The measurements by themselves are a snapshot for comparison in later years. The analysis concentrates on the behavior of WebSurfers, and leads to information about heavy-tailed behavior and to a "source model" for behavior of WebSurfers. It is shown that in principle such source models, together with simulation, lead to information about maximal permissible loads on links in the Internet. At this point, simulation technology is becoming a bottleneck: even sophisticated simulation packages can not include enough endstations to generate realistic loads on links of 45Mbit/sec and up. On theoretical grounds we had previously predicted that the maximal acceptable load on a link in the Internet depends on the access bandwidths of the users. This finding was again confirmed in this project.

1 Introduction

This paper is an analysis of IP traffic measurements obtained in cooperation between Telcordia Technologies and an ISP–Customer of Telcordia Technologies. More specifically, the data were obtained jointly by personel at the ISP–customer , and Teun Ott and Larry Wong at Telcordia. The analysis of the data was done at Telcordia. The objective of this project was two-fold:

- To obtain a snapshot of traffic characteristics for comparison in later years, and
- To build a "source model" (see Sections 7 and 8) that can be used in simulation studies (see SubSection 9.1).

In addition, this paper contains an example of how such a simulation model can be used to obtain information about the load to which a specific link can be loaded before there is noticeable deterioration of quality of service to the users. This load depends on the access bandwidths of those users.

A number of conclusions is presented in the conclusions section at the end of this paper.

2 The Data

Measurements were taken in the configuration described in Figure 1. Measurements were taken on two links:

- A link transporting mostly traffic to and from dial–in customers (modem banks), but also for a number of Frame Relay customers, and
- A link transporting traffic to and from DSL customers.

The measurement tool used was TCPDump. This choice was was based on the need to obtain full traces of the traffic (see below), and on availability. Some of the analysis we did could equally well (in fact more easily!) have been done with a much simpler sniffer which for (say) every 15 minute period gives little more than packetcount and bytecount per application. The bulk of our analysis indeed needed and used the full traces.

Some results on the distribution over time of day and application are given in Section 3

Full traces of time stamped packet headers were obtained for traffic in both directions. We organized the packet streams in *Flows*. A *Flow* is a stream of packets with the same origination and destination addresses, the same origination and destination portnumbers, and the same protocol identifiers. In case of TCP flows, begin and end of a flow are indicated by a SYN and a FIN or RST (Reset). In case of UDP, we did not try to identify begin and end of a flow. For UDP "flows" there is at most one "flow" for every combination of source address, destination address, source port number, and destination portnumber. The organization in pairs of "Flows" (client $\rightarrow$ server and server

→ client) allows us to see the interaction between flows of data packets and acknowledgement packets. More importantly, it allows study of the behavior of individual customers using specific applications like HTTP (TCP/IP port 80), NNTP (TCP/IP port 119), etc.

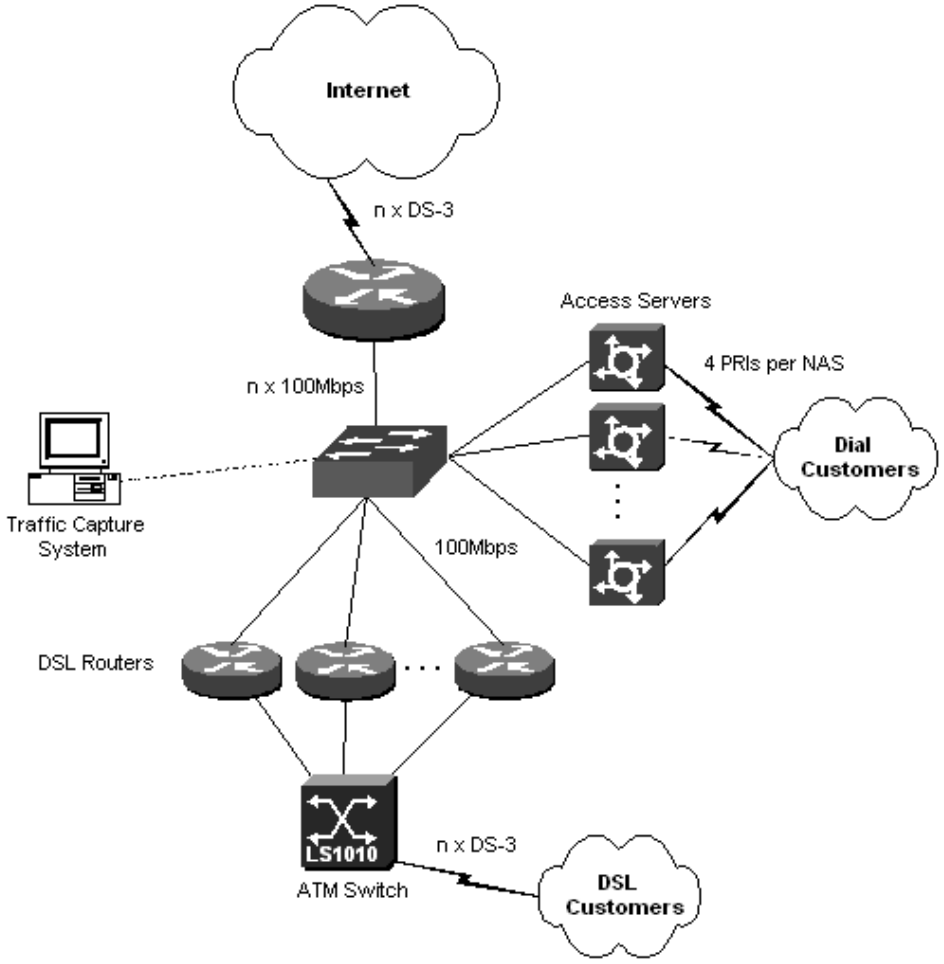

Figure 1: The measurement set–up

In previous measurements we observed that WWW traffic (HTTP, TCP/IP port 80) consistently took 20 to 80 % of the bandwidth used, with in the peak load period typically the higher number or more. For this reason we decided

early in this project to concentrate on HTTP.

Section 3 mostly discusses resource use per application, i.e. bytes, packets, and flows per application. Section 3 is mostly based on the 1999 ISP–customer data mentioned above, but for comparison we also list some further unspecified 1998 data. For the 1998 date we give resource use per application, as well as traffic matrices between applications. In later sections we build a source model WWW (HTTP) traffic based on the 1999 data. We finish by using the WWW source model to give a "proof of concept" that source models, and other knowledge of the customers (in particular bandwidths of the access lines) can be used to set "safe operating levels" for links in the Internet.

Table 1 gives a description of the datasets used in this paper, when each dataset was created and it's size in gigabytes.

Customer	Begin Date	Begin Time	End Date	End Time	# hours	Size (Gb)
Modem	8-19-99	6:08 PM	8-19-99	midnight	6	1.8
Modem	8-23-99	9:00 AM	8-24-99	12:03 PM	27	4.8
DSL	10-12-99	9:53 AM	10-12-99	12:04 PM	2	1.5

Table 1: Dataset descriptions

3 Traffic per Application

This section gives a brief description of the traffic per application (such as Web Traffic, UseNet News Traffic, etc.). First, we show results on our 1999 measurements at the ISP–customer. In a subsection we then give, for comparison, results of some further unspecified 1998 measurements.

Tables 2 and 3 give a breakdown of resource use over various applications. In this paper, we use “application” as essentially synonymous with “IP protocol (TCP, UDP) and portnumber”. A short list of the more important applications is given in Appendix B. Both Tables 2 and 3 were derived from data taken on Tuesday mornings from 10:00 AM to 12:00 noon, but on different days.

port	application	flows	fraction	pkts	fraction	bytes	fraction
80	http	53086	0.7074	560949	0.5540	190855625	0.7037
110	pop3	2102	0.0280	46760	0.0462	12823376	0.0473
9900	unknown	100	0.0013	118030	0.1166	11611391	0.0428
2189	unknown	11	0.0001	29010	0.0287	9896677	0.0365
443	https	1562	0.0208	18631	0.0184	5504080	0.0203
1758	unknown	5	0.0001	18237	0.0180	4963358	0.0183
1701	unknown	23	0.0003	21881	0.0216	3646178	0.0134
67	bootps	4	0.0001	4650	0.0046	2877552	0.0106
4753	unknown	2	0.0000	1457	0.0014	2113868	0.0078
25	smtp	366	0.0049	5979	0.0059	1857303	0.0068
119	nntp	72	0.0010	5164	0.0051	1849571	0.0068
1568	unknown	2	0.0000	19582	0.0193	1825835	0.0067
20	ftp	600	0.0080	6264	0.0062	1622595	0.0060
53	dns	9196	0.1225	10758	0.0106	1450240	0.0053
5190	unknown	209	0.0028	3461	0.0034	1419872	0.0052
other		7703	0.1026	141732	0.1400	16880864	0.0622
total		75043		1012545		271198385	

Table 2: top 15 modem applications

port	application	flows	fraction	pkts	fraction	bytes	fraction
80	http	115795	0.6083	1640319	0.3010	911521756	0.3282
20	ftp	1756	0.0092	459645	0.0843	405847778	0.1461
1080	unknown	16	0.0001	350038	0.0642	327333544	0.1179
119	nntp	184	0.0010	279524	0.0513	245294424	0.0883
5501	unknown	156	0.0008	246739	0.0453	240796730	0.0867
25	smtp	1352	0.0071	113079	0.0207	64424787	0.0232
6699	unknown	14	0.0001	51182	0.0094	58123303	0.0209
554	rtsp	952	0.0050	95878	0.0176	50850895	0.0183
1755	unknown	7	0.0000	50593	0.0093	47917917	0.0173
110	pop3	12447	0.0654	141732	0.0260	33368959	0.0120
1725	unknown	6	0.0000	95651	0.0175	32923165	0.0119
443	https	5881	0.0309	82682	0.0152	27362725	0.0099
139	netbios	156	0.0008	31987	0.0059	23635870	0.0085
2057	unknown	5	0.0000	14170	0.0026	19901692	0.0072
5190	unknown	170	0.0009	43058	0.0079	17077765	0.0061
other		51447	0.2703	1753933	0.3218	270711813	0.0975
total		190344		5450210		2777093123	

Table 3: top 15 dsl applications

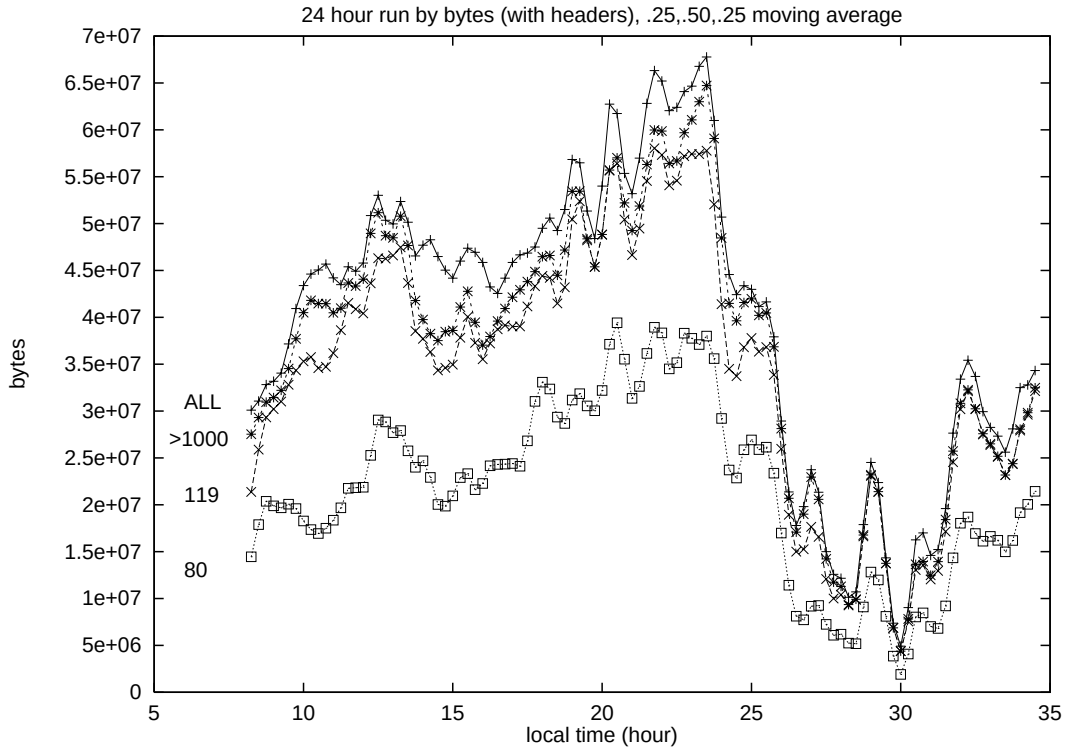

Figure 2: Applications seen in a 24+ hour modem customers

Figure 2 shows resource use by modem customers of the more important applications seen in the 27 hour period. The bottom curve gives resource use by HTTP alone. The next higher curve gives resource use by HTTP and NNTP together. The next higher gives resource use by HTTP, NNTP, and all portnumbers above 1000. The upper curve gives total resource use.

We do not have a similar figure for DSL customers: The data rate for DSL customers, and our available disk space, allowed us to store only a few hours of data. This is an instance where a sniffer that stores only certain counts is more useful than a device like TCPDump.

Tables 2 and 3 and Figure 2 indicate that HTTP and NNTP (plus HTTPS) still are the dominating applications. This evidence is not quite as strong as it was in some previous measurements. In the “modem customers” Table 2 we see that for the period at hand HTTP is only slightly more than 70 % of the bandwidth (plus 2% for HTTPS), and that NNTP has become negligible. However, in the “modem customers” Figure 2 we see that in the peak hour, HTTP and NNTP together still make up over 80 % of the bandwidth, and that at the peak period NNTP is not negligible.

The “DSL customers” Table 3 indicates that in the period at hand HTTP and NNTP together make up only about 43 % of the traffic. It is very unfortunate we did not succeed in obtaining a load distribution over the day for DSL traffic. It is possible the decrease of HTTP is due to the measurement period chosen.

6% of the traffic (in bytes) in Table 2 is “other”, and 10% of the traffic in Table 3 is “other”. This may seem worrysome amounts. We will see that our 1998 data show similar amounts, and in 1 case a large amount traffic for one “unexpected” application. This shows the “other” category probably can not be decreased.

Thus, the conclusion that by knowing the behavior of HTTP and NNTP one “knows” the behavior of the traffic still seems warranted, though possibly with less confidence than in 1998.

It must be noted that the use of TCPDump, or of any other measurement device that makes full traces of the traffic, is overkill, or even countereffective, when the goal is only to obtain results as described in this section. When only results as reported in this section on byte count and packet count are needed, a “sniffer” which for every packet only increases a small number of counters (say packet count, byte count for the application of the packet and for a few

other counters, that may depend on e.g. the source and destination addresses in the packet) but does not keep the packet header is preferable: It would not have the hard disk storage problem with high bandwidth links we had with the DSL link.

A simple “sniffer” as just described might not be able to provide counts of flows per application, and would certainly be unable to provide the type of results we will see in later sections of this paper.

3.1 Some 1998 Data

In this subsection we show data on some 1998 IP traffic measurements, for comparison with the 1999 data that are the heart of this paper. The 1998 measurements were taken at a further unspecified location, different from the 1999 data. The 1998 “modem traffic” is data on dial-in customers, but without frame relay customers mixed in. The 1998 “T1 traffic” is of a different mix of customers. The data are shown in Tables 4 and 5.

In the 1998 measurements we saw that WWW traffic, dependent on the time of the day, took 20 to 75% of the bandwidth, and WWW and NNTP (Network News Transport Protocol) together, depending on the time of the day, took 70 to 95% of the bandwidth, with in the peak load periods mostly the higher numbers. In the 1999 data this trend continues, with possibly somewhat less domination by HTTP and NNTP, and a significant decrease in NNTP. It is possible that people reading NetNews now use HTTP.

port	application	bytes	%	pkts	%	flows	%
20	FTP	493755	.08	933	.05	110	.09
21	FTP	100734	.02	1193	.07	57	.04
25	SMTP	5099592	.86	12188	.67	316	.25
53	DNS	2633703	.44	16537	.91	14559	11.42
80	WWW-HTTP	333377055	56.24	868026	47.69	86347	67.76
81	HOSTS2-NS	194463	.03	412	.02	54	.04
110	POP	12849906	2.17	62470	3.43	10556	8.28
113	AUTH-113	7210	.00	104	.01	60	.05
119	NNTP	70193635	11.84	125178	6.88	3507	2.75
123	NTP	8178	.00	87	.00	21	.02
137	NetBios	421451	.07	3861	.21	147	.12
161	SNMP	304310	.05	3306	.18	69	.05
443	HTTPS	12116002	2.04	36478	2.00	1351	1.06
514	Syslog	361469	.06	3552	.20	1	.00
1058	NIM	1721732	.29	17039	.94	69	.05
1352	Lotus-Note	4711701	.79	12047	.66	16	.01
1431	RGTP	4131170	.70	5597	.31	18	.01
1650	NKD	1936091	.33	8580	.47	16	.01
2916	Elvin	392235	.07	6063	.33	6	.00
5151	ESRI_SDE	110961990	18.72	483770	26.58	92	.07
6667	IRC	2431933	.41	19437	1.07	38	.03
other	other	25216040	4.25	88727	4.87	9870	7.75
N/A	no_port	3804816	.64	46085	2.53	320	.25
	total	529730951		1820214		127437	

Table 4: 1998 Traffic through dial–up modems

port	application	bytes	%	pkts	%	flows	%
20	FTP	896539	1.05	2524	0.91	55	0.19
21	FTP	23449	0.03	391	0.14	22	0.07
25	SMTP	4608025	5.40	14542	5.22	1169	3.98
53	DNS	201793	0.24	1761	0.63	835	2.85
80	WWW-HTTP	75782717	88.78	249460	89.57	26591	90.64
110	POP	567201	0.66	1292	0.46	20	0.07
113	AUTH-113	6132	0.01	136	0.05	134	0.46
137	Net-Bios	26438	0.03	251	0.09	30	0.10
161	SNMP	128	0.00	2	0.00	1	0.00
443	HTTPS	164795	0.19	600	0.22	51	0.17
520	Route	780	0.00	15	0.01	1	0.00
1352	Lotus-Note	1814956	2.13	3114	1.12	111	0.38
other	other	1210030	1.42	3685	1.32	210	0.72
N/A	no_port	650	0.00	109	0.04	108	0.37
	total	85364206		278493		29336	

Table 5: 1998 Traffic on T1

Table 6 gives the matrix of traffic intensities in bytes between the various applications (ports) in similar 1998 modem bank measurements. We did not do this analysis for our 1999 data. The sum of all entrees is 100%. Blank entrees indicate an absolute zero, while zeros indicate a small positive value rounded down to zero. Thus, in our modem data 48.28% of all bytes observed were going from a port 80 (WWW) on some source to a port not in our list of frequently observed portnumbers, on some destination. These of course are WWW packets from a server (port 80) to a client (any port). Only 7.736% of the bytes in the same data went from a "other" port to port 80. These of course are WWW packets from a client to a server. Table 7 gives similar information on a per packet basis. Here, the corresponding percentages for port 80 are 23.32 and 24.18. Clearly, Packets from a Web Server to a client tend to be large, while packets from a client to a Web Server tend to be small (mostly acknowledgement packets). Table 8 gives similar information on a per flow basis.

	20	21	25	53	80	110	119	137	161	443	514	1058	1352	1431	1650	2916	6667	5151	other	NA
20																			0.078	
21																			0.01	
25												0.001							0.072	
53								0				0.001	0	0	0	0			0.264	
80												0.101	0.031	0.03	0.024	0.005			48.28	
110												0.008	0.002	0	0				1.802	
119												0.103				0			11.1	
137				0				0.071												
161																			0.026	
443												0.001							0.778	
514											0.061									
1058			0.001	0	0.019	0.001	0.005			0									0.084	
1352				0	0.006	0													0.063	
1431				0	0.005	0													0.676	
1650				0	0.003	0													0.273	
2916				0	0		0												0.035	
6667																			0.294	
5151						0.001												18.72		
other	0.005	0.007	0.787	0.178	7.736	0.354	0.631		0.025	1.265		0.086	0.73	0.02	0.054	0.031	0.116		4.254	
NA																			0.004	0.642

Table 6: 1998 Modem Traffic Matrix, Bytes.

	20	21	25	53	80	110	119	137	161	443	514	1058	1352	1431	1650	2916	6667	5151	other	NA
20																			0.031	
21																			0.033	
25												0.002							0.316	
53								0				0.001	0	0.001	0	0			0.384	
80												0.049	0.017	0.016	0.009	0.003			23.32	
110												0.007	0.001	0	0.001				1.645	
119												0.024				0			3.669	
137				0				0.211												
161																			0.091	
443												0.001							0.912	
514											0.195									
1058			0.002	0.001	0.051	0.005	0.025			0.001									0.428	
1352				0	0.019	0.001													0.254	
1431				0.001	0.016	0													0.203	
1650				0	0.009	0.001													0.233	
2916				0	0.002		0												0.175	
6667																			0.52	
5151						0.003												26.57		
other	0.02	0.033	0.35	0.52	24.18	1.766	3.159		0.091	1.09		0.438	0.405	0.102	0.237	0.158	0.548		4.875	
NA																			0.012	2.532

Table 7: 1998 Modem Traffic Matrix, Packets.

Appendix B contains a list of all applications that occur in the Tables 4 and 5, and a few more that were prominent in other recent datasets.

The application ESRI_SDE, while prominent in the 1998 Table 4, was not a good candidate for further investigation, because its traffic seemed to be generated by just a few users. While there were about 20 to 25 IP addresses using this portnumber (talking to each other), we believe that the total number of users is smaller. Our understanding is that the IP addresses of “dial–in users” we see are temporary IP addresses, assigned when the users dial in and freed up when the users closes the dial–in connection. This is an example of the care that is needed in interpreting measurements. This seems a good reason not to worry about the “other” categories in the 1999 data in Tables 2 and 3.

	20	21	25	53	80	110	119	137	161	443	514	1058	1352	1431	1650	2916	6667	5151	other	NA
20																			0.043	
21																			0.024	
25												0.002							0.116	
53								0.003				0.012	0.003	0.005	0.002	0.001			4.847	
80												0.075	0.027	0.02	0.009	0.001			32.42	
110												0.005	0.001	0.001	0.001				1.847	
119												0.004				0.001			1.377	
137				0.002				0.11												
161																			0.027	
443												0.002							0.513	
514											0.001									
1058			0.002	0.012	0.078	0.005	0.004			0.002									0.004	
1352				0.003	0.027	0.001													0.002	
1431				0.005	0.018	0.001													0.001	
1650				0.002	0.009	0.001													0.004	
2916				0.001	0.002		0.001												0.001	
6667																			0.015	
5151						0.046												0.027		
other	0.043	0.02	0.127	6.526	35.07	6.376	1.365		0.027	0.544		0.002	0.002	0.001	0.004	0.001	0.015		7.745	
NA																			0.022	0.251

Table 8: 1998 Modem Traffic Matrix, Flows.

4 Detailed Traces of WWW Flows

We use two ways to make it easier to visualize the behavior of a WWW session.

One is to show the time period over which TCP connections are in existence. We call these “existence plots”. An example is Figure 3. The other is to give for TCP connections the cumulative number of bytes as function of time. We call these “activity plots”. An example is Figure 4. We see that TCP connections can alternate between activity and inactivity, while remaining in existence.

These examples Figures 3 and 4 are not for actual customers: they were obtained by the second author of this report accessing the WWW, and monitoring the resulting packet flows using TCPDump. The following series of URLs was visited:

- http://www.yahoo.com
- photography
- photographers
- masters
- robert capa
- robert capa photographs
- next page
- exhibitions
- kenro izu

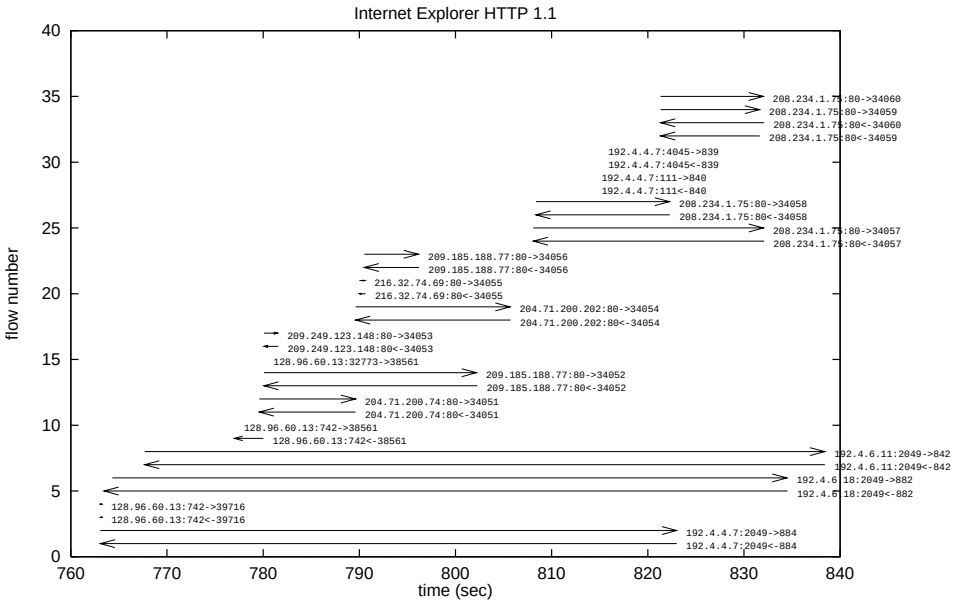

Figure 3: Internet Explorer, HTTP 1.1, existence

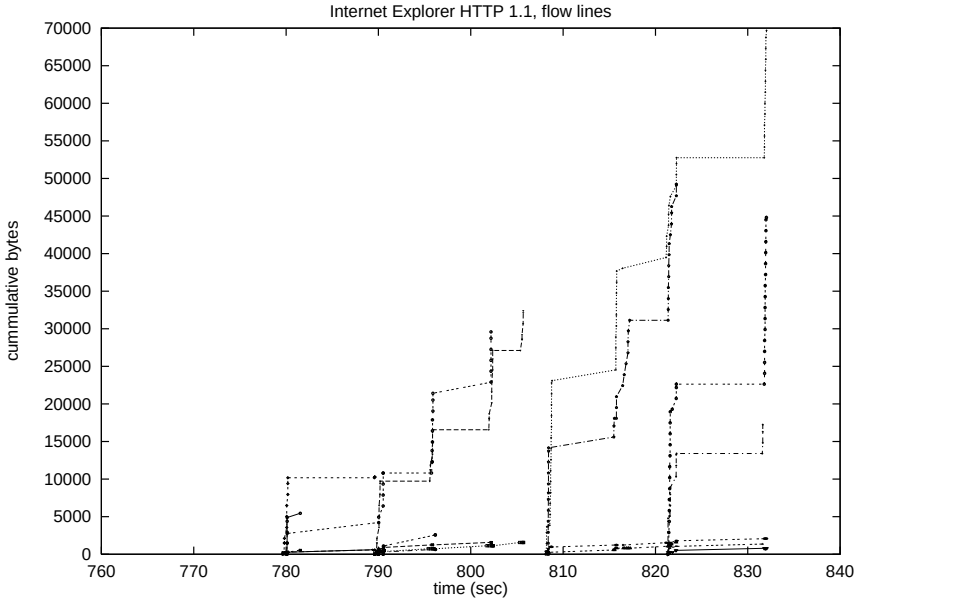

Figure 4: Internet Explorer, HTTP 1.1, activity

Figure 3 shows various TCP connections come into existence and be terminated. We see that TCP connections are in pairs: one from the server to the client ($\rightarrow$) and one from the client to the server ($\leftarrow$). The Figure also gives the address of the server and the portnumber on the server and on the client. Since this is HTTP traffic the portnumber on the server often, but not by a long way always, is 80. For example, we often see portnumber 53 (Domain Name Server), this means the client is trying to get the address of the server for a new URL. Other portnumbers also occur. Those TCP connections are not active all the time. Figure 4 depicts the activity of the various TCP connections: For every packet seen, it gives both timestamp and cumulative bytecount (including headers) until that packet. We see that TCP connections alternate between activity (steep increase of the activity plot) and inactivity (the activity being essentially flat).

Figures 3 and 4 are for an "HTTP 1.1" transaction, obtained using Internet Explorer, version 6.0 (which has HTTP 1.1). This means that for a specific WWW session in a client there is no hard upper limit on the number of simultaneous pairs of simultaneous TCP connections that can exist to a specific server (but the recommended upper limit is 2 TCP connections per server), and that there is "persistence": a TCP connection is kept in existence when not in use, in case another file from the same server needs to be transported. The TCP connection usually is discarded when the WWW session in the client accesses a new server. TCP connections for advertisements seem to often remain in existence even when the client accesses a new server. Figure 4 shows that quite often multiple TCP connections are simultaneously active.

A result of this simultaneous activity is that if the throughput bottleneck of those flows is in the MaxWindow (TCPWindow) or in the "backbone" network,

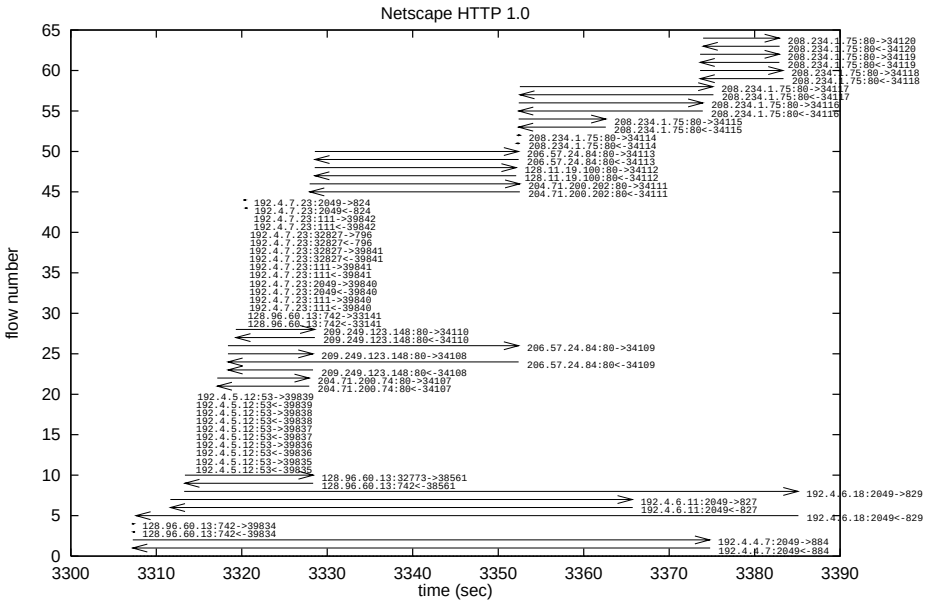

Figure 5: Netscape, HTTP 1.0, existence

the WWW session gets higher throughput. If the bottleneck is in the access line, these multiple parallel TCP connections are competing with each other and having multiple parallel connections does not really make sense.

Figures 5 and 6 similarly give existence and activity for “the same” session, but now using Netscape version 4.61 and HTTP 1.0.

These figures were obtained pointing the Netscape browser at the same sequence of URLs as those in Figures 3 and 4.

Netscape has the newer version of HTTP 1.0, where at most 4 simultaneous TCP connections can be open between a WWW session in a client and a specific server, and those up to four connections have “keepalive” (i.e., persistence) until the client moves to another server. We see that the activity lines are fairly similar. There are differences in DNS accesses, these may be due to luck

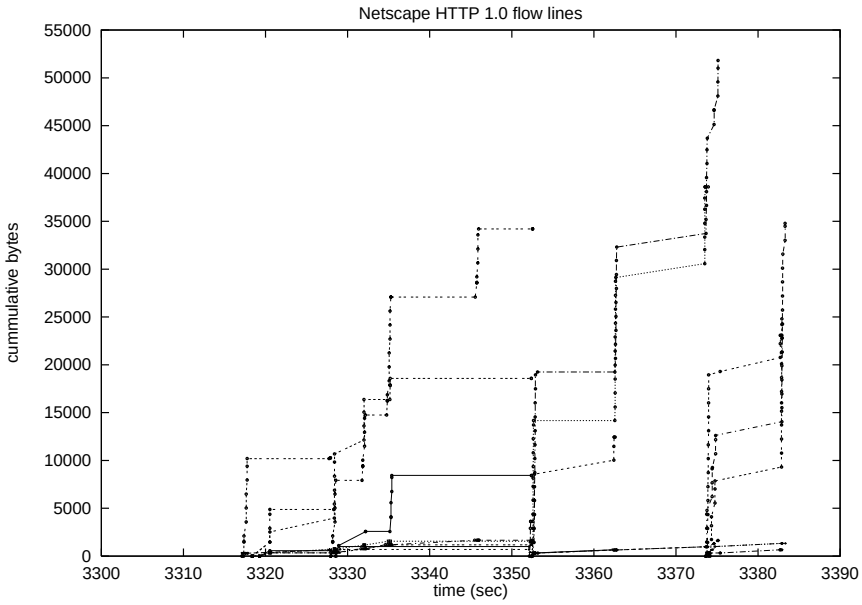

Figure 6: Netscape, HTTP 1.0, activity

(or the opposite) in address caching.

Figure 7 gives the activity for an actual customer. This customer happens to be a dial–in customer using HTTP 1.0.

We observe that this specific customer has simultaneously active connections, but not as pronounced as happened in our measurements at Telcordia. We also see that this actual customer does not get the high bandwidth (steep almost vertical stretches in the activity curves) we observed in our workstation and PC at Telcordia. This must be due to the fact that our (Telcordia's) internal network is 10 Mbit/sec and up (mostly ethernet), and the access line from Telcordia to its ISP also is about 10 Mbit/sec (7 $\times$ T1). Modem customers have bandwidths no higher than 64 Kbit/sec. In the next section we will see that Frame Relay customers have at most about 250 Kbit/sec, and that while a few DSL customers have bandwidths over 1 Mbit/sec, almost all have band-

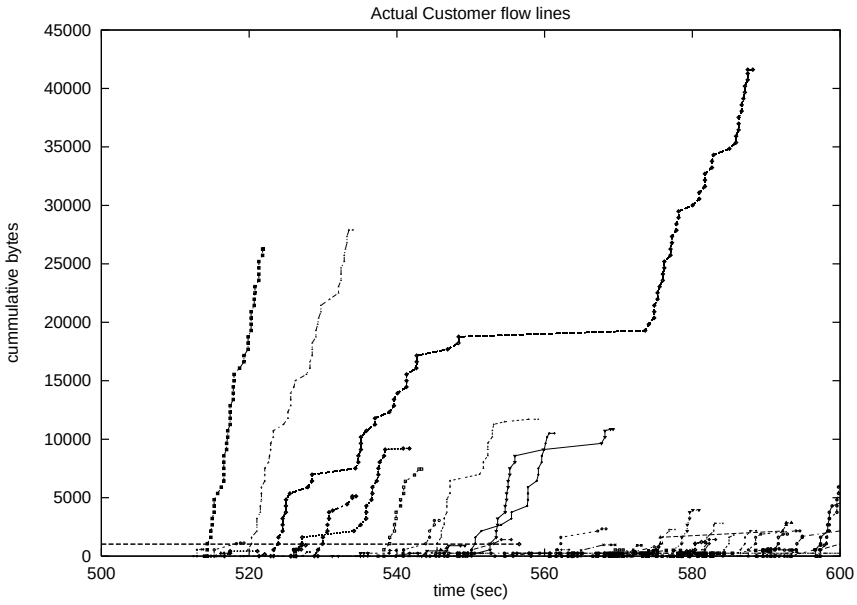

Figure 7: Actual Customer activity plot

widths below 700 Kbit/sec (even in the “downward” direction, toward the DSL customer).

5 Access Line Speeds

Plots as in Figure 7 show how fast a file was moved from the client to the server, and thus give an indication of (in fact a lower bound for) the bandwidth of the access line of the customer (in the direction from network to customer). When that access line is not the bottleneck, the lower bound is a very low estimate, and also when two or more files are transported in parallel (over competing TCP connections) the lower bound becomes a very low estimate.

Such estimates can be significantly improved on, for example as follows: For a given length of time, say T seconds, find that time interval of length T for which the sum of all amounts of traffic the client received from all servers together is maximal. Let that amount be M bytes. Then, $\frac{8M}{T}$ is a plausible estimate for the access linespeed of the client (bits/sec, in the "downward" direction). A problem is of course the choice of length T. This depends on the estimated linespeed. A sensible choice seems to be to restrict attention to clients that at some time received a file of at least 50 KBytes, and take for T half the time it took to download that file (i.e., T depends on the client). Instead of all intervals of length T we can restrict ourselves to intervals of the form $(x, x+T]$, where x is incremented by values of the order of $T/10$.

In preparing this paper we did not do this more sophisticated analysis. Instead we took all files and subtracted the packet sizes of the first seven packets to take care of windowing effects. If the resulting file size was 50 KBytes or larger we divided the new file size by the time it took to move the file (not including the first seven packets). This results in a distribution of speeds over files (all files larger than 50 KBytes). The taildistributions are given in Figures 8 (for dial-in data) and 9 (for DSL data). Figure 10 gives both

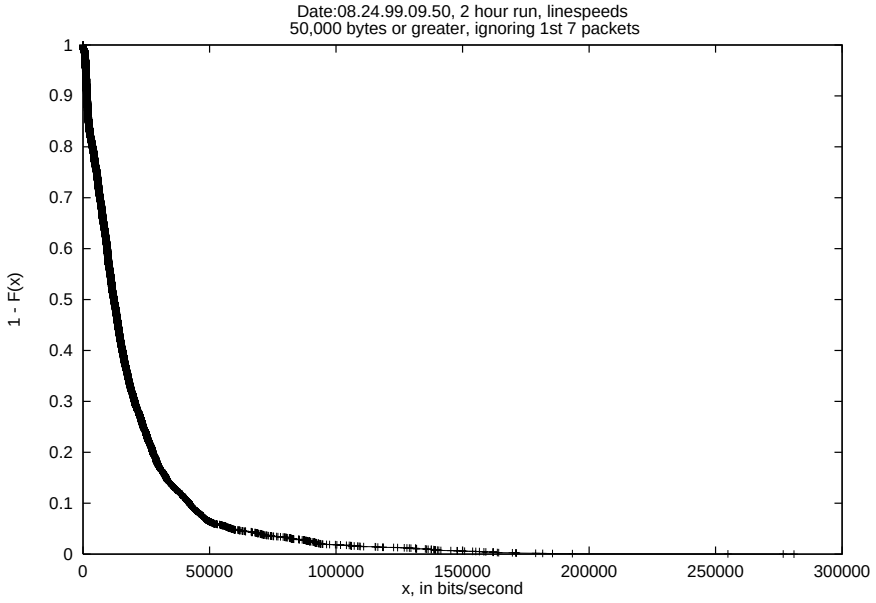

Figure 8: Distribution of “File Speeds”, modem customers

distributions in one figure.

While the estimated speeds are below linespeeds, for higher speeds they must give a reasonably good picture. Figure 8 shows that many of the “dial–in” customers have linespeeds well over 100 Kbits/sec. This is explained by the fact that there are Frame Relay customers among the dial–in customers.

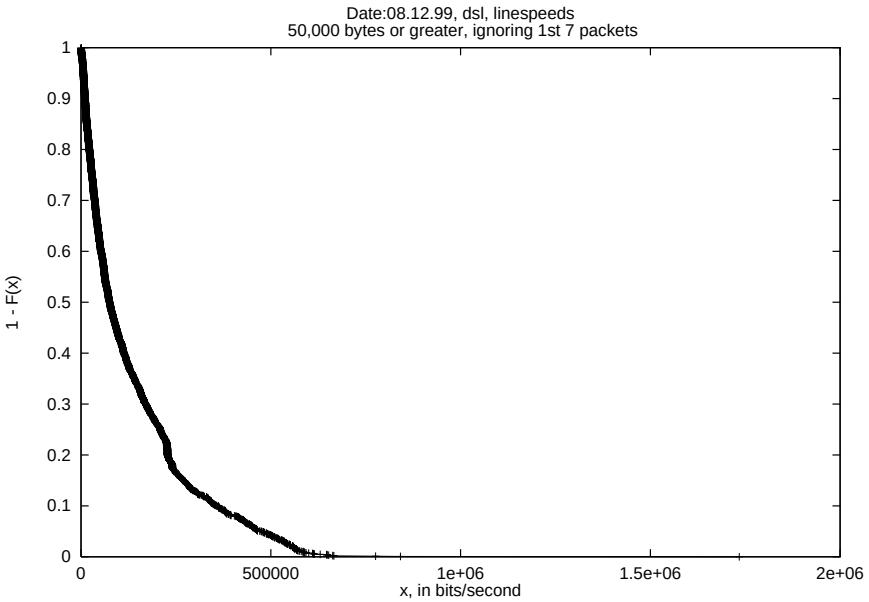

Figure 9: Distribution of “File Speeds”, DSL customers

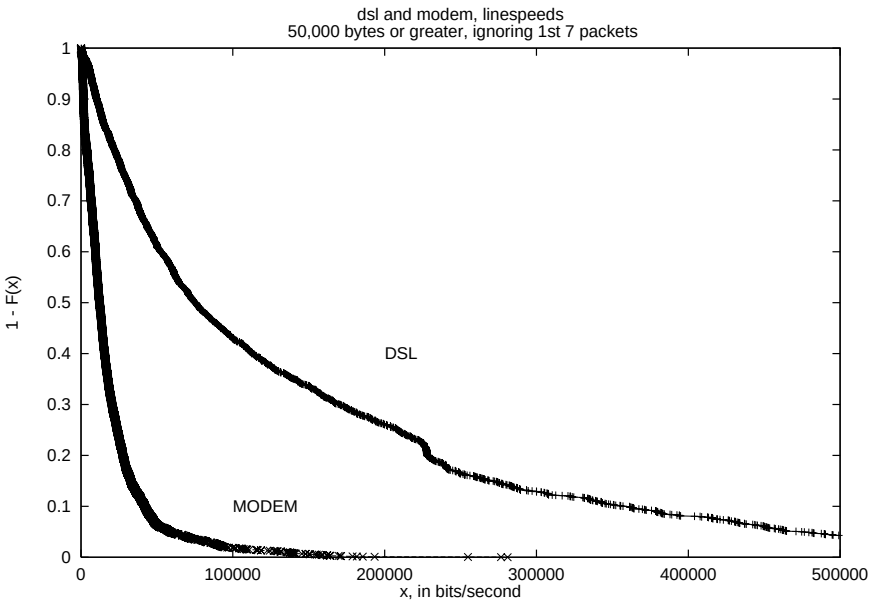

Figure 10: Distribution of “File Speeds”, modem and DSL customers

6 HTTP, Methods and Gets

This section concentrates on the HTTP aspects of the traffic. Whenever the client (computer) makes a request to the server (computer), we call this a “hit”. This name is somewhat misleading: The request can indeed be caused by the customer typing in a URL, or clicking on an item. It can, however also be the result of an “autonomous” action on the part of the browser. Later we will distinguish between “human clicks” and “computer clicks”. First we describe the breakdown between HTTP 1.0 and HTTP 1.1.

Table 9 gives the number and percentage of HTTP 1.0 requests v.s. HTTP 1.1 requests (“clicks”) for modem customer for a 24 hour period. These are “human clicks” as well as “computer clicks”.

HTTP	hits	fraction
1.0	470392	.50790
1.1	455760	.49210
total	926152	

Table 9: HTTP 1.0 vs 1.1 Modem customers, 24 hour period

Table 10 gives the number and percentage of HTTP 1.0 requests v.s. HTTP 1.1 requests for DSL customers for a 2 hour period.

HTTP	hits	fraction
1.0	68111	.33567
1.1	134797	.66433
total	202908	

Table 10: HTTP 1.0 vs 1.1 DSL customers 2 hour period

Table 11 gives the numbers and percentages of HTTP 1.0 requests v.s. HTTP 1.1 requests for dial–in customers, for a 6 hour run on a Friday night.

HTTP	hits	fraction
1.0	122,905	.738
1.1	43,606	.262
total	166,511	

Table 11: HTTP 1.0 vs 1.1 Modem

We see that DSL customers (not surprisingly!) have more HTTP 1.1 than dial–in customers. We also see that during the evening there are fewer HTTP 1.1 users than in average over the day. Apparently, “Friday evening Surfers” have older equipment than “Daytime Surfers”.

When a “click” occurs as above, i.e. when the client computer makes a request to the server computer, the click or request is of one of a number of “methods”. See IETF RFC 1945 and RFC 2068 for more details on HTTP methods. It must be noted that the client that makes the request can be either a human customer, or the browser on behalf of the human customer. When the human customer explicitly clicks on a URL, we call this a “human click”, or “human request”. When the browser, “on behalf of the human” requests a file (say some graphics) we call this a “computer click” or “computer request”. Later we will discuss how to guess, from the data, which clicks are “human” and which are “computer”. Tables 12 and 13 give the frequencies of the various methods for two different time periods. Both tables are for dial–in customers. In these tables we do not differentiate between “human” and “computer” generated requests.

HTTP method	hits	fraction
POST	793	.01976
GET	39211	.97753
HEAD	84	.00209
PUT	24	.00059
DELETE	0	0
TRACE	0	0
CONNECT	0	0
TOTAL	40112	

Table 12: HTTP method, 2 hours on a Tuesday morning

HTTP method	hits	fraction
POST	2707	.01574
GET	168433	.97945
HEAD	620	.00360
PUT	206	.00119
DELETE	0	0
TRACE	0	0
CONNECT	0	0
TOTAL	171966	0

Table 13: HTTP method, 6 hours on a Friday night

6.1 Gets

We see that “Get” is by far the most frequently occurring HTTP Method. Henceforth we only consider “Gets”. There is a large number of different types of “Gets”:

- asf. Active Streaming Format for multimedia. The ASF specification defines a structure to combine miscellaneous file types such as audio, video, scripts, ActiveX controls and HTML documents. See http://www.microsoft.com/mind/0997/netshow/netshow.htm for details.
- asp. Active Server Page. Webpages that utilize ASP technology end with the extension .ASP, rather than the common .HTML extension. These pages contain code which pulls information from database files and then displays them to your browser. ASP pages are similar to CGI scripts, but typically contain Visual Basic or Jscript code.
- asx. ASX Files are text files that act as a link from a Web page to an ASF file on a (HTTP or NetShow) server. See http://www.microsoft.com/netshow/howto/asx.htm for details.
- cd. When a customer changed directories. This is indicated when the last character in the GET method string is a ”/”. i.e. the URL http://dir.yahoo.com/Arts/ would bring the user to yahoo’s arts and humanities page.
- cd_init. When a customer goes to a site’s root directory ”/”. i.e. going to the URL ”www.yahoo.com” goes to the yahoo root directory.
- gif. A picture file format created by CompuServe.

- jpg. A picture file format created by the Joint Photographic Experts Group.
- mov. Apple Computer QuickTime video file format. Mov is an abbreviation for Movie. See http://www.apple.com/quicktime/ for details.
- mp3. MP3 is an open audio compression codec that was developed by the Motion Picture Experts Group (MPEG). The abbreviation ”MP3” stands for MPEG Layer 3. MP3 is currently the de facto Internet audio standard. MP3 audio is near-CD quality and several hundred thousand song files are currently posted online. See http://www.mp3.com for details.
- pl. Perl. Perl is a high-level programming language derived from C, sed, awk, and the Unix shell. See http://www.perl.com/ for details.

Table 14 gives a distribution of the filesizes that are transported from the server to the client in response to the various types of “Gets”. In this table we do not yet worry about which gets are human generated, and which are computer generated. The data in Table 14 is for dial–in customers.

get type	events	frac/ev	bytes	ave	var	s.d	coef of var
asf	5	0.0002	94476	18895.2	1.34021e+09	36608.8	3.75378
asp	93	0.0028	706459	7596.33	1.14556e+08	10703.1	1.98523
asx	6	0.0002	3213	535.5	40864.6	202.15	0.142504
cd	525	0.0158	5336750	10165.2	3.18113e+08	17835.7	3.07855
cd_init	717	0.0216	7328936	10221.7	2.3023e+08	15173.3	2.20352
cgi	5820	0.1755	42646805	7327.63	1.25908e+09	35483.6	23.4492
class	125	0.0038	509720	4077.76	5.03964e+07	7099.05	3.03079
css	177	0.0053	155763	880.017	3.23005e+06	1797.24	4.17088
gif	18406	0.5550	41318858	2244.86	5.56819e+07	7462.03	11.0493
giv	3	0.0001	1251	417	288	16.9706	0.00165623
htm	2008	0.0605	16435269	8184.89	2.64846e+08	16274.1	3.95337
jpg	5067	0.1528	34131328	6736	1.71488e+08	13095.4	3.77946
js	135	0.0041	197520	1463.11	9.09118e+06	3015.16	4.24684
mov	3	0.0001	530772	176924	4.7187e+10	217226	1.50747
mp3	14	0.0004	307733	21980.9	1.12064e+09	33476	2.3194
pl	13	0.0004	187743	14441.8	2.83534e+08	16838.5	1.35945
txt	46	0.0014	281536	6120.35	6.85607e+08	26184.1	18.303
TOTAL	33163						

Table 14: HTTP GET method breakdown, all

6.2 Human Gets

For simulation purposes, we decided to differentiate between "human" requests and "computer" requests. We classified all non-jpeg and non-gif gets as human clicks. This decision was based on manual analysis of the TCPDump traces in the Figures 3 – 6, and of about 10 other traces from our data. Of the jpeg and gif gets, we classify those that result in a file from server to client of less than or equal to 20,000 bytes as "computer clicks", but those that result in a larger file from server to client as a "human click". The reason for this choice is that quite often a number of "thumbnail" graphics files are requested automatically by the browser. Experimentation by one of the authors of this paper backs up the limit of 20 KBytes.

Thus, a "human get" results in transport of one file from the server to the client that the human customer explicitly asked for, plus a number of other files that the browser requests (hopefully "on behalf" of the human customer, but actually often against that human user's desire(!), more on behalf of an advertiser).

Table 15 gives the distribution of types and resulting "bunch" sizes for "Human Gets". For example, 2008 times a human explicitly asked for an htm or html file. These htm files, plus the files that the browsers then autonomously asked for, represent a total of 30,343,246 bytes (including packet headers). Comparing Tables 14 and 15 we see that while there were a total of 18406 requests for a gif file, only 229 of these were above 20KBytes. The 717 cd_init files in Table 14 together contained 7,328,936 bytes, but Table 15 shows that these 717 cd_init human gets, together with the computer gets that followed them, represented 10,462,262 bytes.

get type	events	frac/ev	bytes	ave	var	s.d	coef of var
asf	5	0.0005	94476	18895.2	1.34021e+09	36608.8	3.75378
asp	93	0.0090	1299324	13971.2	4.66281e+08	21593.5	2.3888
asx	6	0.0006	4524	754	326938	571.785	0.575073
cd	525	0.0508	8547962	16281.8	7.10958e+08	26663.8	2.68187
cd_init	717	0.0693	10462262	14591.7	4.59588e+08	21438	2.15852
cgi	5820	0.5628	62906977	10808.8	1.36331e+09	36923	11.6693
class	125	0.0121	1019275	8154.2	2.8106e+08	16764.9	4.22704
css	177	0.0171	840959	4751.18	1.17513e+08	10840.4	5.20575
gif	229	0.0221	11438223	49948.6	2.89771e+09	53830.3	1.16147
giv	3	0.0003	92898	30966	8.80787e+08	29678.1	0.918546
htm	2008	0.1942	30343246	15111.2	6.62357e+08	25736.3	2.90065
jpg	423	0.0409	20045338	47388.5	1.00251e+09	31662.4	0.446419
js	135	0.0131	1595535	11818.8	3.96683e+08	19916.9	2.83987
mov	3	0.0003	530772	176924	4.7187e+10	217226	1.50747
mp3	14	0.0014	342355	24453.9	1.16167e+09	34083.2	1.9426
pl	13	0.0013	197506	15192.8	2.95238e+08	17182.5	1.27908
txt	46	0.0044	373059	8109.98	6.98899e+08	26436.7	10.6261
all	10342	na	150134691	14517	1.1937e+09	34549.9	5.66424

Table 15: HTTP human GET method breakdown, “bunches”

7 Customer Behavior

In this section we describe “customer behavior”, of course specialized to HTTP. This will be in terms of thinktime distributions, filesize distributions, etc. for customers surfing the web.

In the remainder of this section we will describe the empirical distributions we found. We used the data for dial–in customers to obtain “theoretical distributions” that later were used in simulations. We decided not to do an extremely refined analysis, but to just capture what we think are the main aspects of the behavior.

The model we would have liked to achieve is a “thinktime - bunch - thinktime - etc” model: The “real” customer (the human) thinks, then requests a file. That file, when it arrives at the client (the client computer, let’s call it the browser), causes the browser to request a few more files. Finally, transfer stops, and the “real” customer (the human) starts thinking again. If the data had born out this model, we could have characterized the model by thinktime distribution (time from end of transfer until next human click), and entities such as the distribution of the initial filesize (requested by the human), and distributions of the number and sizes of the later files in the bunch, and the distribution of the total bunch size.

Such a model would have the advantage of being inherently closed loop: When the network gets congested, it takes longer to transport a “bunch” (based on the TCP protocol), and the amount of service the customer gets decreases when congestion increases, and even decreases sufficiently fast.

Such a model would of course be an oversimplification: In reality, thinktime distributions change when congestion increases and effective transfer rates

decrease. Quite possibly, also filesize distributions change, and certainly at some point the customer reneges (walk away to have coffee, or to write an angry letter, or logs off).

The reason we could not use this oversimplified model is that, certainly for the “dial–in” customers, different bunches overlap.

Ideally, the human customer waits until after transmission of the previous bunch has finished, before generating the next “human click”. This would generate a positive thinktime. It is also satisfactory if the next human click causes all TCP connections from the previous “bunch” to be closed. This would, somewhat misleadingly, be interpreted as a zero thinktime.

However, moderately often, the filetransfer caused by the next human click is (partially) simultaneous with a file transfer from the previous bunch.

In order to model this (somewhat roughly) we organize human clicks into three classes, or **types**.

- **type 1** These are human clicks that come after the end of transmission of the previous bunch, or at least cause immediate termination of all remaining TCP connections from previous bunches.
- **type 2** These are human clicks, after which at least one previous HTTP TCP connection remains in existence, and the new server is different from those of all those persistent older connections.
- **type 3** These are human clicks, after which at least one previous TCP connection remains in existence, and the new server has one of those persistent older connections.

The next subsection gives information about frequencies of the various types of human clicks.

7.1 Types of Human clicks

Tables 16 and 17 give total numbers and fraction of type 1, 2, 3 “human clicks” for respectively dial–in customers and DSL customers.

type	hits	fraction
type1	6995	.67637
type2	1206	.11661
type3	2141	.20702
total	10342	

Table 16: Total number of types 1, 2 and 3, dial–in

type	hits	fraction
type1	45224	.98939
type2	244	.00534
type3	241	.00527
total	45709	

Table 17: Total number of types 1, 2 and 3, DSL

Tables 18 and 19 give numbers and frequencies of transitions between types for dial–in customers,

We see that 84% of all type 1 human clicks is followed by another type 1 human click, and .94% of all type 1 human clicks is the last human click in the HTTP session.

	goto type1	goto type2	goto type3	goto end
type1	5876	515	538	66
type2	442	579	177	8
type3	588	112	1426	15

Table 18: Transitions between human get types, raw, dial–in

	goto type1	goto type2	goto type3	goto end
type1	0.8400	0.0736	0.0769	0.0094
type2	0.3665	0.4801	0.1468	0.0066
type3	0.2746	0.0523	0.6660	0.0070

Table 19: Transitions between human get types, fractions, dial–in

Tables 20 and 21 give numbers and frequencies of transitions between types for DSL customers.

	goto type1	goto type2	goto type3	goto end
type1	44623	171	167	263
type2	165	73	1	5
type3	166	0	72	3

Table 20: Transitions between human get types, raw, dsl

We see that while for “dial–in” customers only 68% of all human clicks are type 1, for DSL customer that percentage is 99%. Thus, the conceptual model we would have liked is reasonable for DSL lines, but in case of dial–in lines the transmission is sufficiently slow that human do not wait for completion, and make a new request while the old file (or advertisement) is still rolling out. For dial–in customers, there seems to be a mild degree of “diagonal domination” in the transition matrix Table 19. This may indicate that certain customers tend

	goto type1	goto type2	goto type3	goto end
type1	0.9867	0.0038	0.0037	0.0058
type2	0.6762	0.2992	0.0041	0.0205
type3	0.6888	0.0000	0.2988	0.0124

Table 21: Transitions between human get types, fractions, dsl

to stick with one type of “human click”.

Tables 19 and 21 show that the overwhelming majority of “human clicks” is not the last human click in the HTTP session. In other words, the total number of “human clicks” or “bunches” in an HTTP session usually is very large.

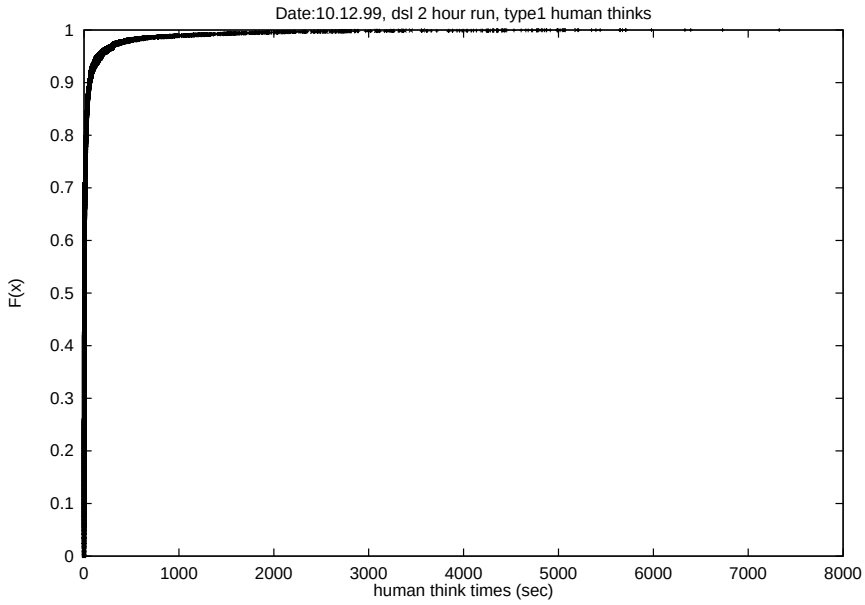

Figure 11: human think characteristics, modem and dsl

7.2 Think Times

Figures 11 and 12 give the distribution function ($F(x)$ versus x) of the "human think times" preceding a type 1 human click, for "dial–in" as well as "DSL" customers, first over the whole range and then then up to one second. Because Figure 11 is not informative, Figures 13 and 14 give the same data as $\log(1 - F(x))$ versus x, and Figure 15 again gives the same information, but now in a plot of $\log(1 - F(x))$ versus $\log x$.

Figure 16 repeats Figure 15 for dial–in customers only, with an indication of the distributions we use to simulate the data. Figure 17 repeats repeats Figure 12 for dial–in customers only, with an indication of the distributions we use to simulate the data.

Figures 11, 13, 14 and 15 indicate that apart from the first few seconds,

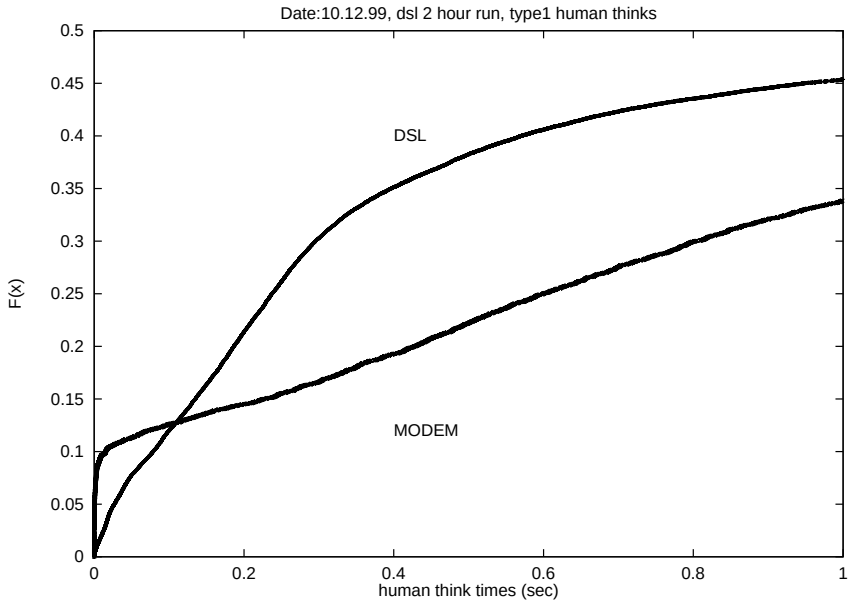

Figure 12: human think characteristics, modem and dsl, detail

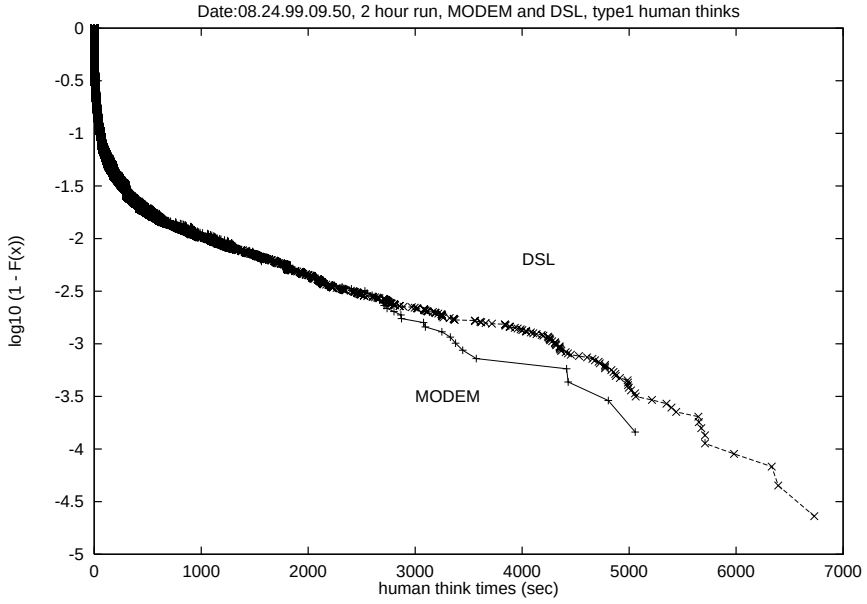

Figure 13: human think times, modem and dsl

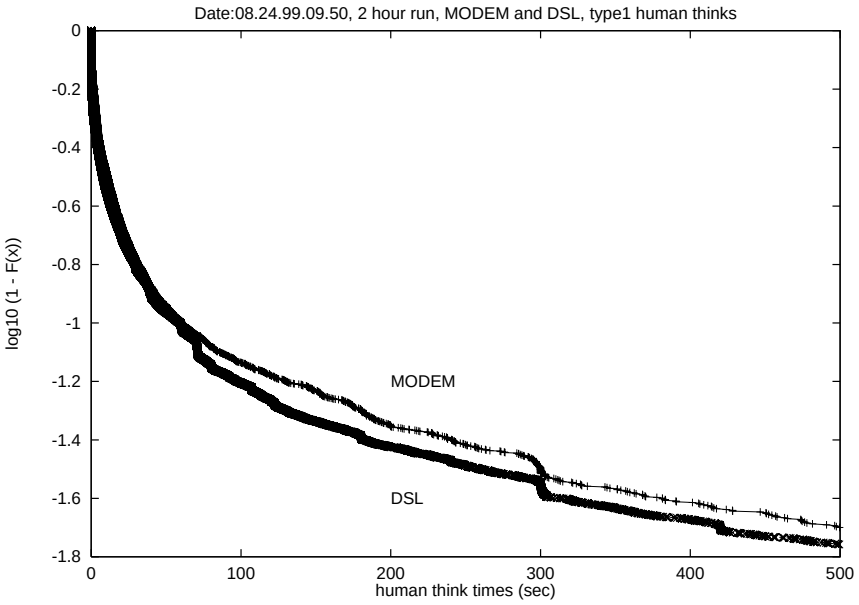

Figure 14: human think times, modem and dsl, 0–500 sec

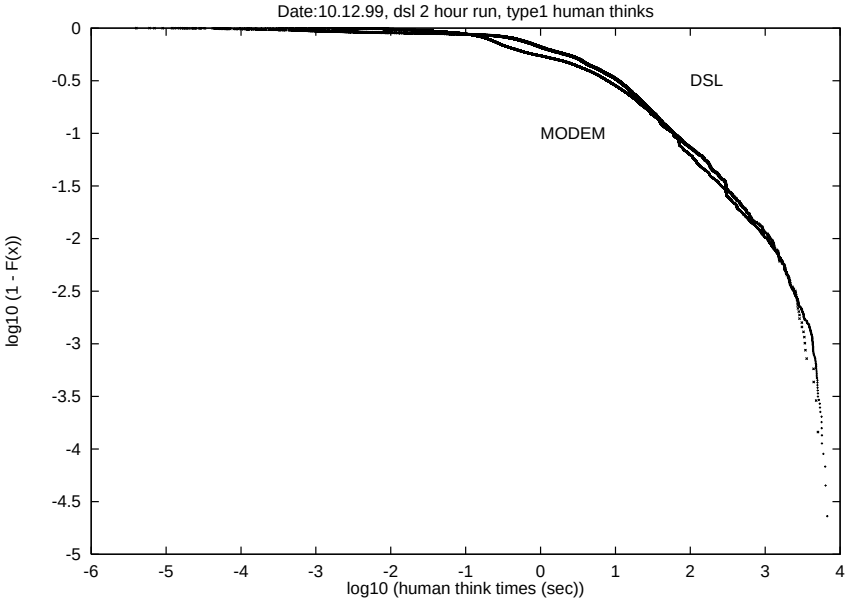

Figure 15: human think characteristics, log log plot, modem and dsl

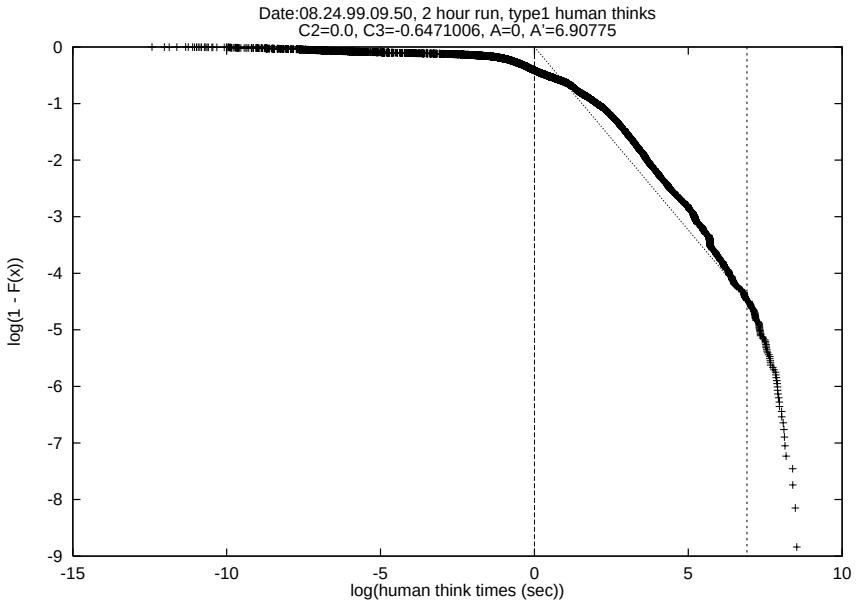

Figure 16: Human think time distribution, dial–in

the think time distributions for “dial–in” and “DSL” customers are quite close. In the simulations we use the data for the “dial–in” lines. Figure 12 shows that quite often (almost 10% of the time) dial–in customers request a new file before the previous “bunch” is completed. The same figure shows that DSL customers hardly ever do this. This strongly suggests that the zero think-times of dial–in customers are due to impatient customers hitting the “reload” button. DSL customers have faster downloads and receive the whole “buch”, without getting impatient. This agrees with the findings in Tables 16 – 21.

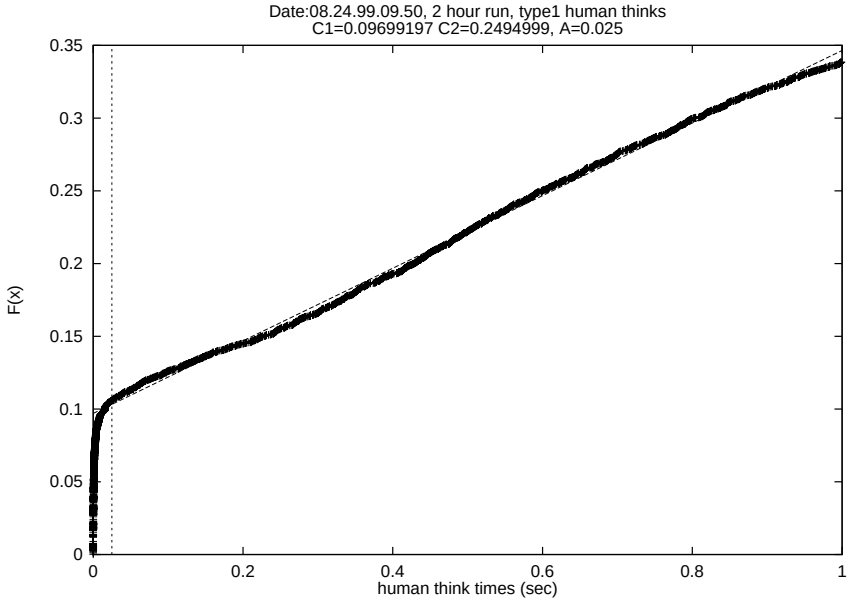

Figure 17: Human think time less then one second, dial–in

Figures 13 and 14 indicate that thinktimes have a distribution with an exponential tail: after slightly less than 100 seconds, the "hazardrate" of the human customer requesting a file becomes constant. We did not utilize this fact in our simulation.

Based on the data, we decided, in the simulations, to generate human think times for type 1 human clicks as follows:

From various regression analyses we obtain

$$p_0 = .097, c_1 = .2495, c_2 = .0, c_3 = .647\ . \tag{7.1}$$

We compute U^* by

$$U^* = 1 - \left({c_1}^{\frac{c_3}{1+c_3}}\ e^{\frac{c_2}{1+c_3}} \right) \tag{7.2}$$

Generate a random variable U, which is uniformly distributed over the interval $(0, 1)$. Then the thinktime X is defined as

$$X = 0 \ \text{ if } \ U \leq p_0, \tag{7.3}$$

$$X = \frac{U - p_0}{c_1} \ \text{ if } \ p_0 \leq U \leq U^*, \tag{7.4}$$

$$U = \max \left(\frac{U - p_0}{c_1}, e^{\frac{c_2}{c_3}} (1 - U)^{-\frac{1}{c_3}} \right) \ \text{ if } \ U > U^*. \tag{7.5}$$

The coefficient $c_3 = .647 < 1$ would indicate that the thinktime distribution is not only heavy–tailed, but does not even have a finite first moment. Figure 16 already shows that this conclusion is questionable. In fact, Figure 13 indicates

that an exponential tail for the thinktime distribution fits the data better. This would be consistent with previous findings, that the think time distribution has a decreasing hazardrate which at some point becomes constant. Figure 13 actually seems to indicate that after 4000 seconds the hazardrate starts increasing. This is probably due to the fact that the observation periods span only two hours.

For the purposes of this paper, we felt it was excusable to take liberties with the thinktime distribution, see SubSection 9.1.

7.3 Human Gets

Figures 18, 19, 20, 21 and 22, give the "human get size" distribution for both the modem and DSL customers. These are the distributions of the sizes of the files that are transmitted as a direct consequence of the "human click", i.e. not including the files requested by subsequent computer gets. These distributions are not differentiated with respect to the type of the human get.

Figure 23 repeats some of this information for dial–in customers only, and adds some information on how we later simulate those distributions.

Based on these plots, we decided to generate, in the simulation, those "human gets" as follows:

From varies regression lines we obtained a number of constants, as follows:

$$c_1 = 1.648, c_2 = .328, c_3 = 17.72, c_4 = 1.92\ . \tag{7.6}$$

Generate U Unif$(0,1)$. The human get size X is obtained from

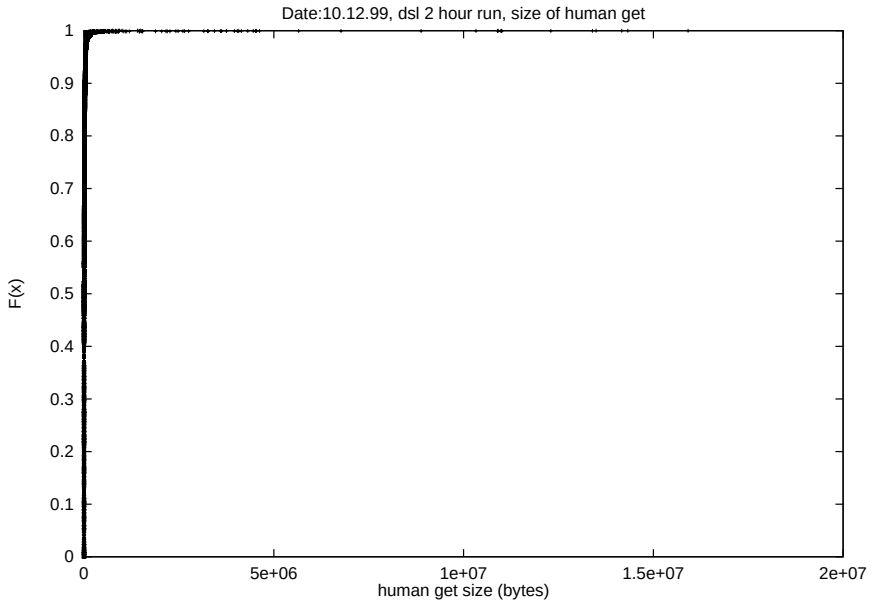

Figure 18: human get sizes for modem and DSL

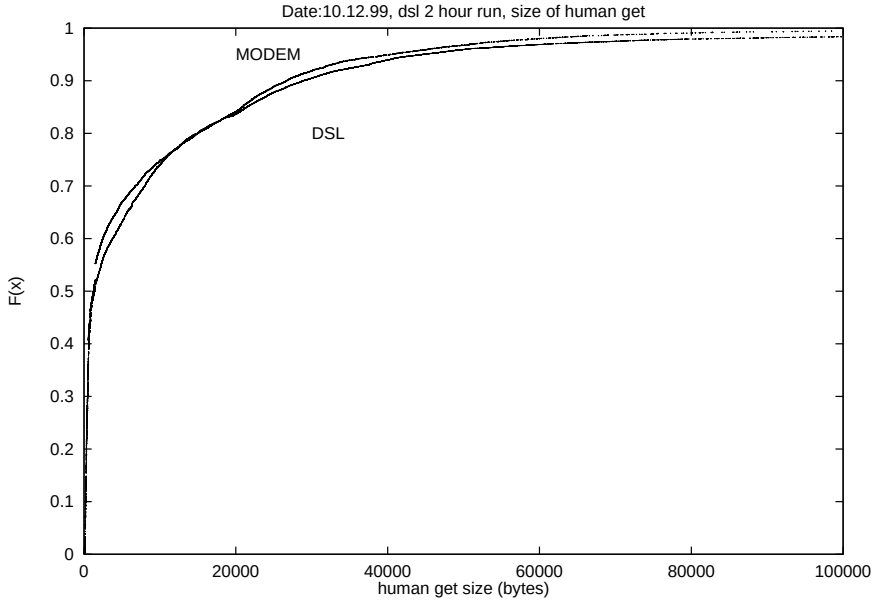

Figure 19: human get sizes for modem and DSL, detailed view

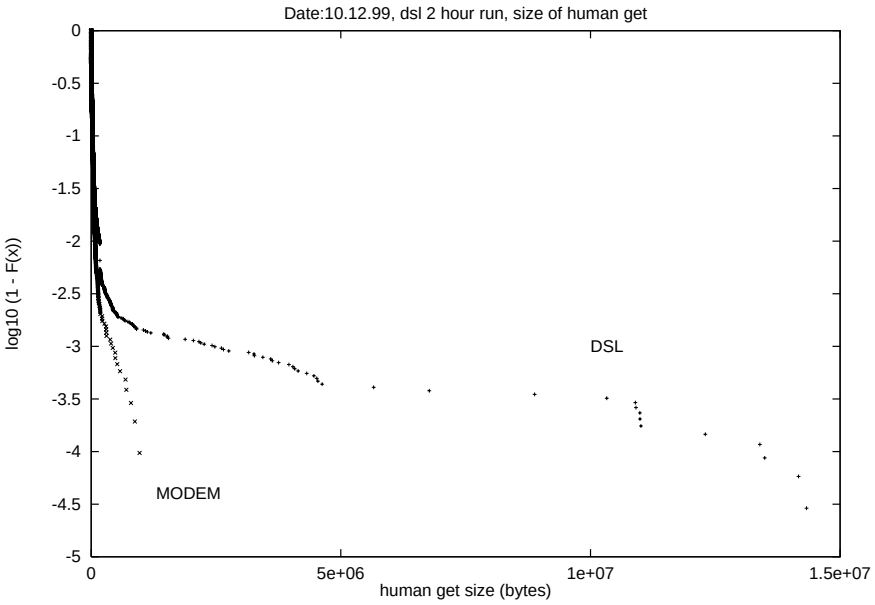

Figure 20: human get sizes for modem and DSL, log plot

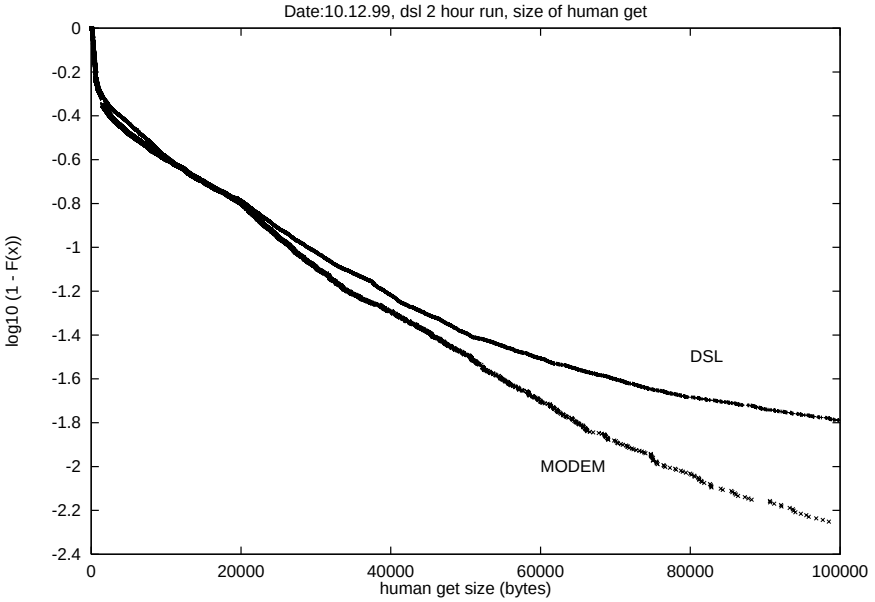

Figure 21: human get sizes for modem and DSL, log plot, detail

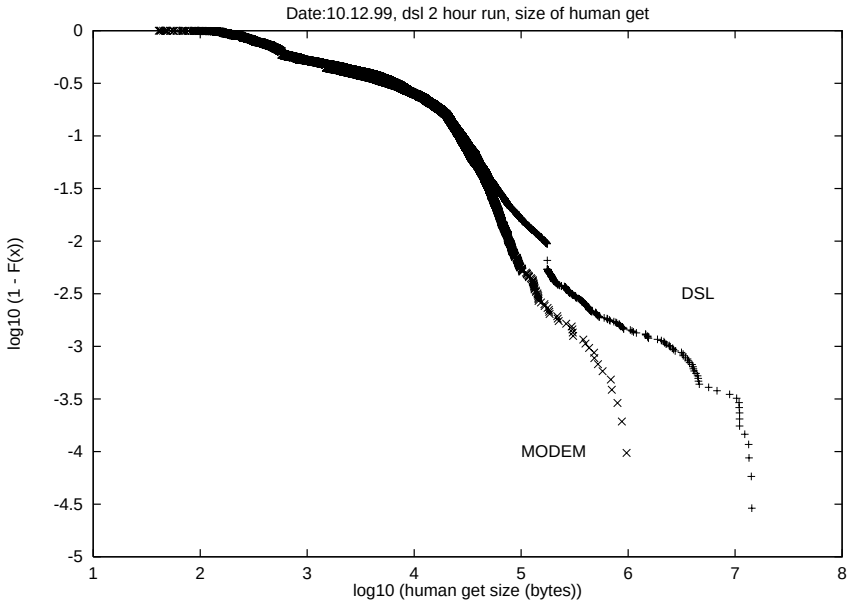

Figure 22: human get sizes for modem and DSL, log log plot

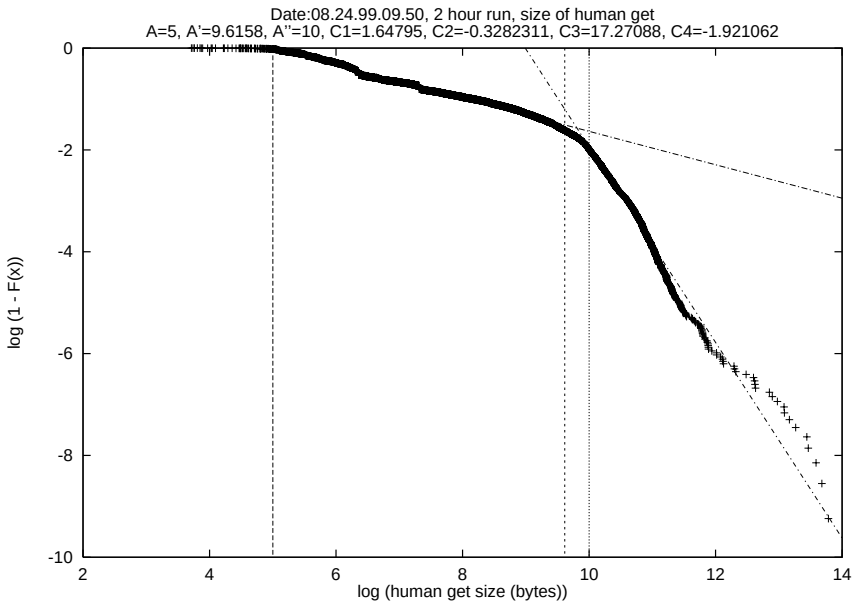

Figure 23: Size of a human get distribution

$$X = \min\left(e^{\frac{c_1}{c_2}}(1-U)^{-\frac{1}{c_2}}, e^{\frac{c_3}{c_4}}(1-U)^{-\frac{1}{c_4}}\right) \tag{7.7}$$

The coefficient $c_4 = 1.92 < 2$ in (7.6) indicates the distribution of the human get filesizes is heavy tailed, with a finite first moment but an infinite variance. A second look at Figures 23 and 22 indicates that for dial–in customers this conclusion is warranted, while for DSL customers the distribution is even worse heavy tailed. We did not obtain the (tail) shape parameter for DSL human gets.

7.4 Computer Gets

Figures 24, 25 and 26 gives the computer get size distribution for both the modem and DSL customers. These are the distributions of the sizes of the files that are transported to the client in response to a "computer get".

Figure 27 repeats some of this information for dial-in customers only, and adds some information on how we plan to simulate these distributions.

Based on these data, we decided to generate, in the simulations, those "computer gets" as follows: From a regression analysis we determined a few coefficients:

$$c_1 = 3.664, c_2 = .665\ . \tag{7.8}$$

Draw a random variable U, Unif$(0,1)$. Then X, the size of the computer get is first computed as

$$X = e^{\frac{c_1}{c_2}}(1-U)^{-\frac{1}{c_2}}. \tag{7.9}$$

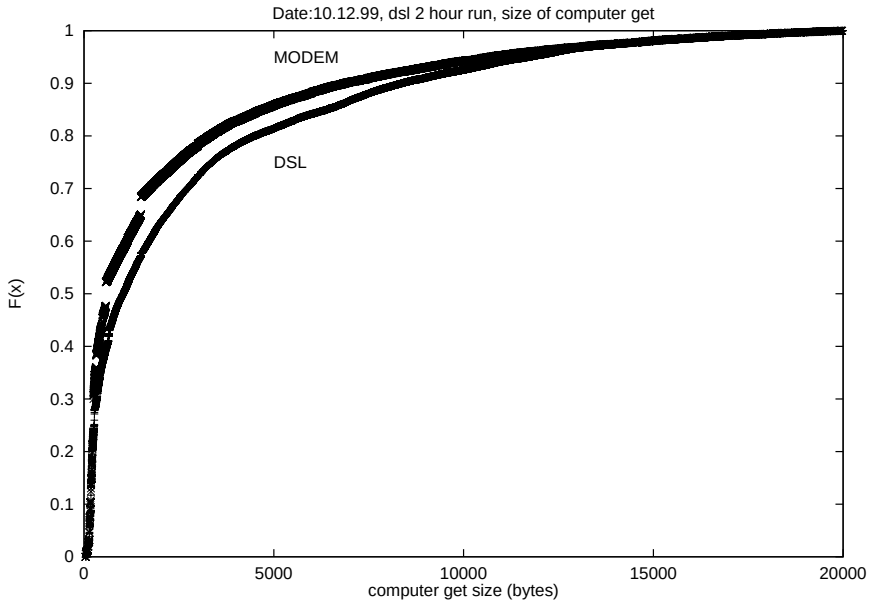

Figure 24: Computer Get Size Distribution for modem and DSL

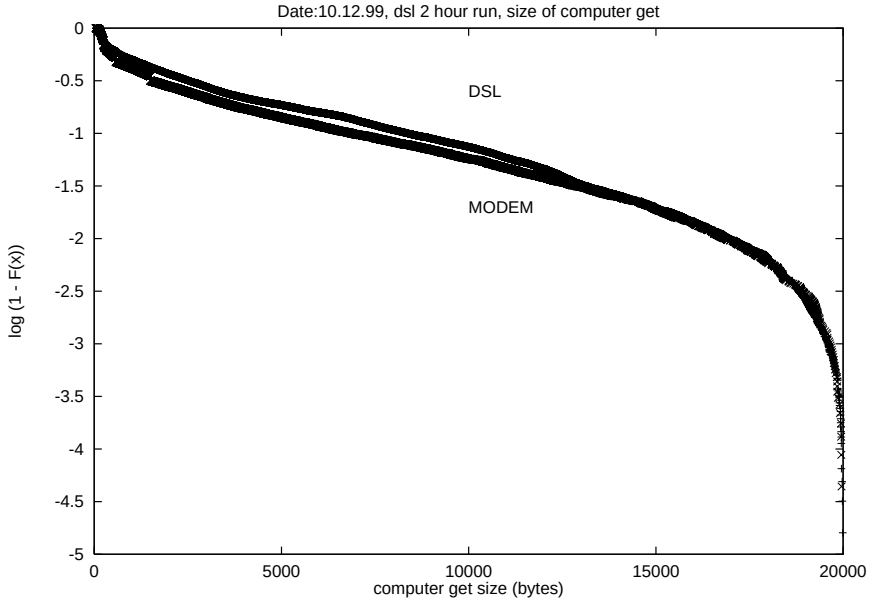

Figure 25: Computer Get Size Distribution for modem and DSL, log plot

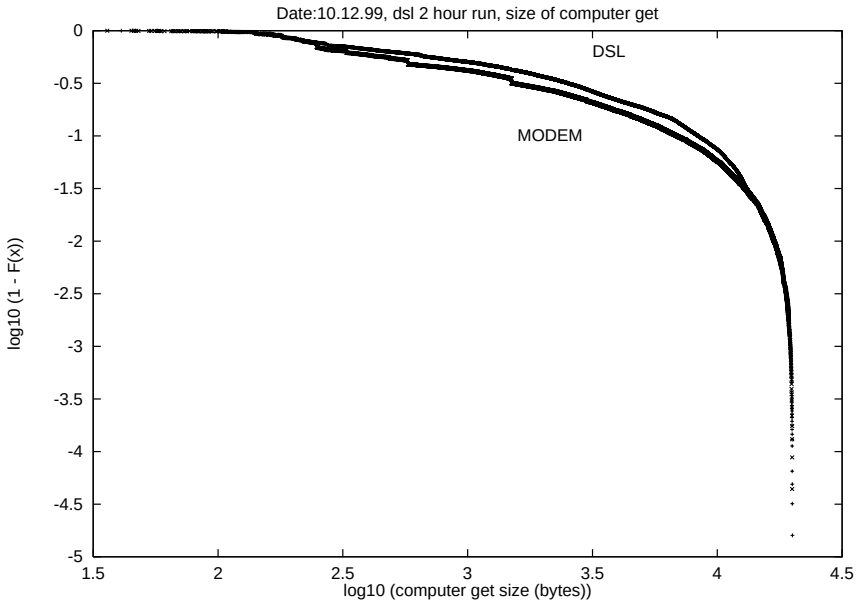

Figure 26: Computer Get Size Distribution, log log plot

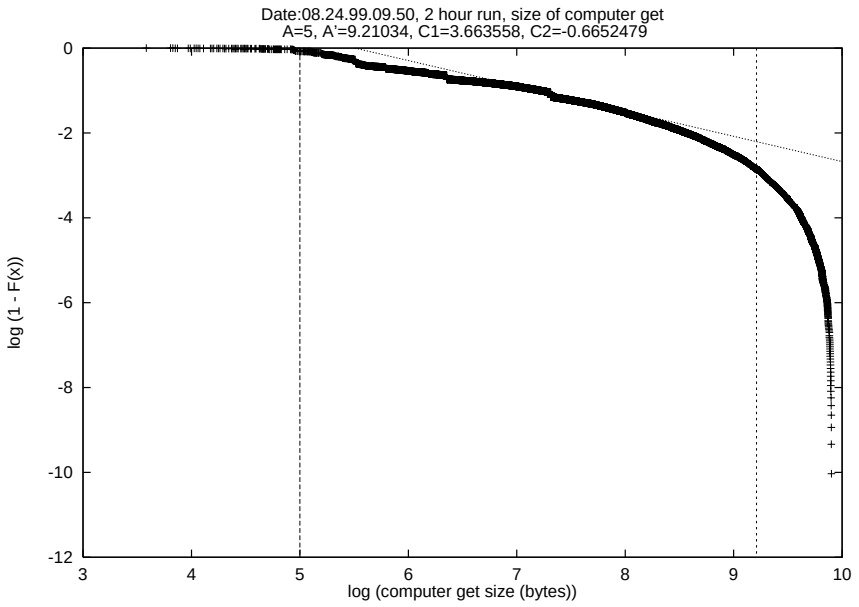

Figure 27: Size of Computer Gets distribution, dial–in

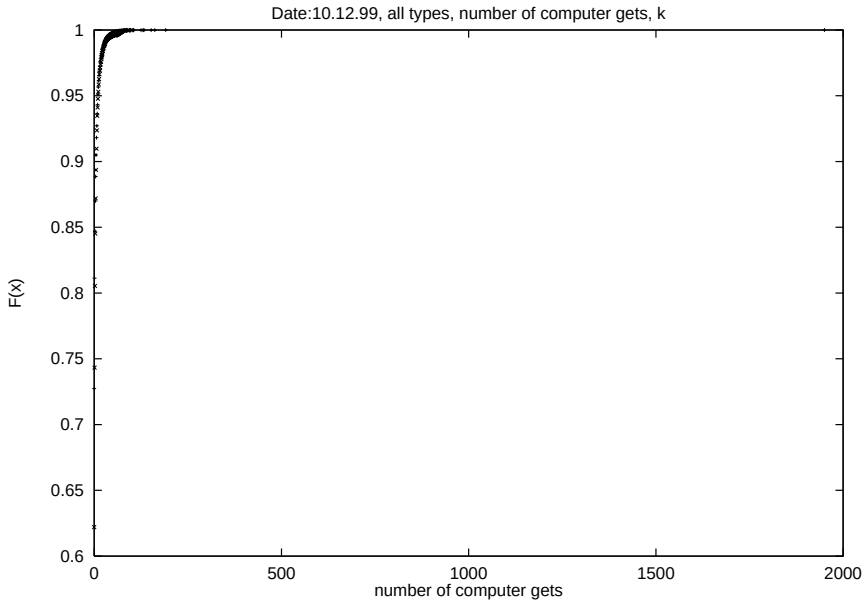

Figure 28: Number of Computer Gets per Human Get

However, if this computation results in $X > 20$KBytes, we reset X to 20KBytes.

Because of this maximum of 20KBytes, the distribution is not heavy tailed.

7.5 Number of Computer Gets per Human Get

Figures 28, 29, 30, and 31 give the distributions of the numbers of "computer gets" that are triggered by a single "human get".

Figure 32 repeats some of this information for dial–in customers only, and adds information on how we plan to simulate these data.

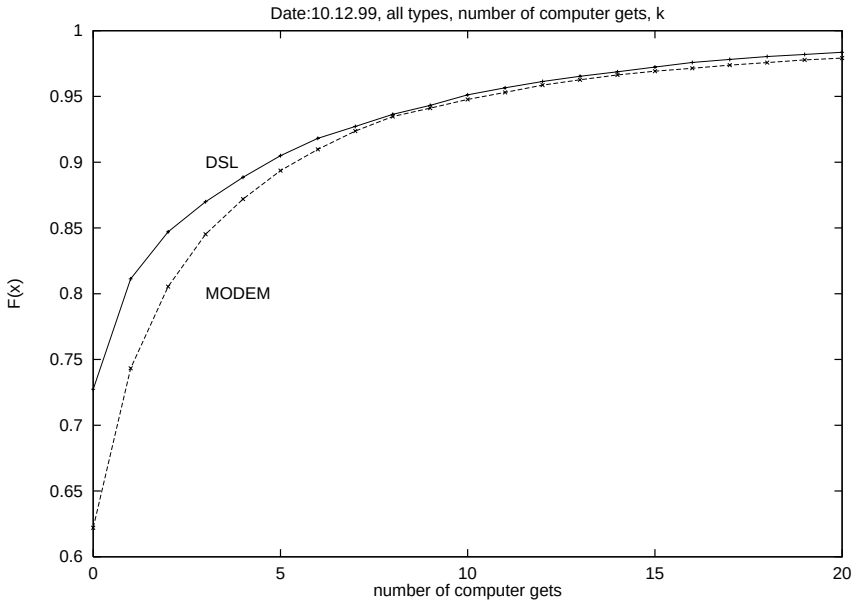

Figure 29: Number of Computer Gets per Human Get, detailed view

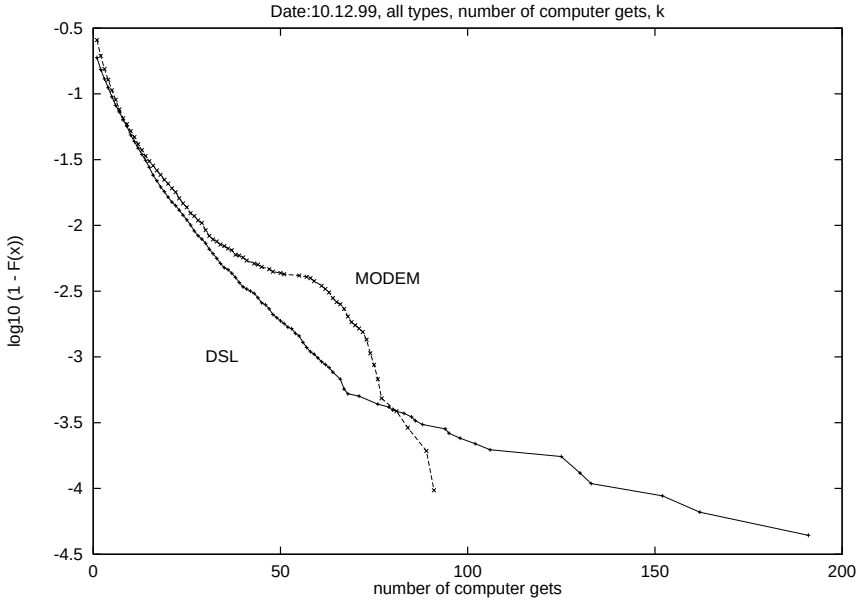

Figure 30: Number of Computer Gets per Human Get, log plot

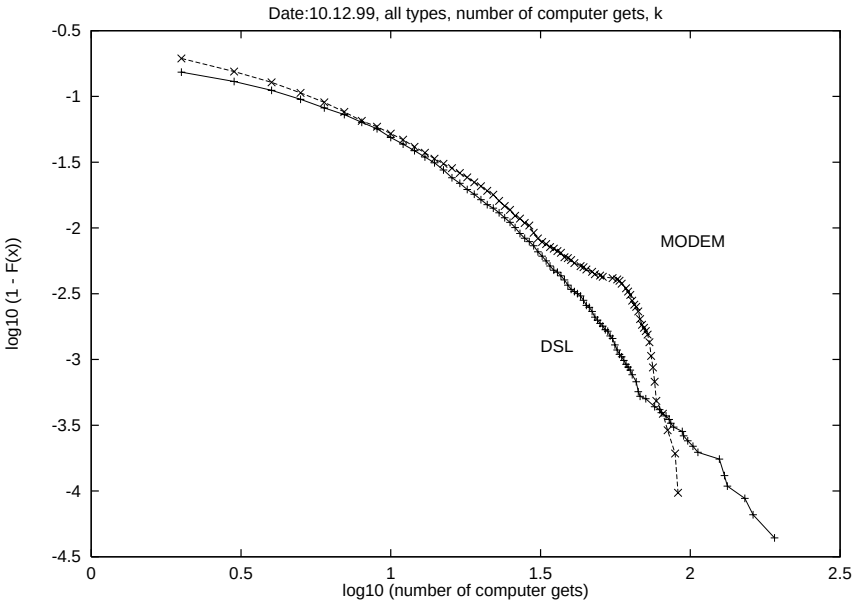

Figure 31: Number of Computer Gets per Human Get, log log plot

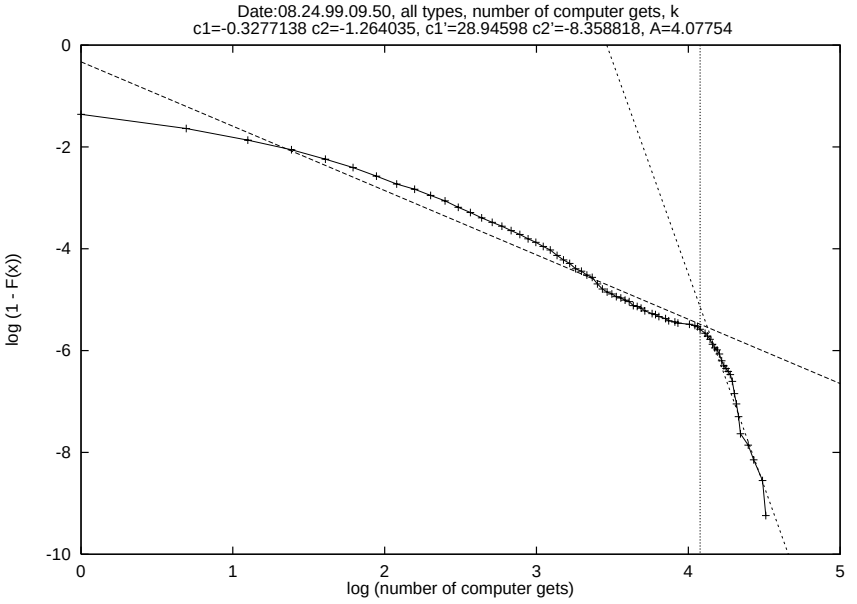

Figure 32: Number of Computer Gets per Human Get

Based on these data, we decided to generate the number of "computer gets" triggered by one "human get" as follows: First, by regression we determine a number of coefficients:

$$c_1 = -.3277, c_2 = 1.264, c_3 = 28.95, c_4 = 8.359. \tag{7.10}$$

Generate U Unif$(0,1)$. The number of computer gets generated by a human get is now obtained from

$$K = \min\left(e^{\frac{c_1}{c_2}}(1-U)^{-\frac{1}{c_2}}, e^{\frac{c_3}{c_4}}(1-U)^{-\frac{1}{c_4}}\right). \tag{7.11}$$

The coefficient $c_4 = 8.359$ indicates that the distribution of the number of computer gets per human gets, for dial–in customers, has a polynomial tail, but has finite moments up to the eighth moments. It is possible the number of computer gets per human get for DSL customers might be heavy-tailed. In the simulation we used the distribution above, based on dial–in customer data.

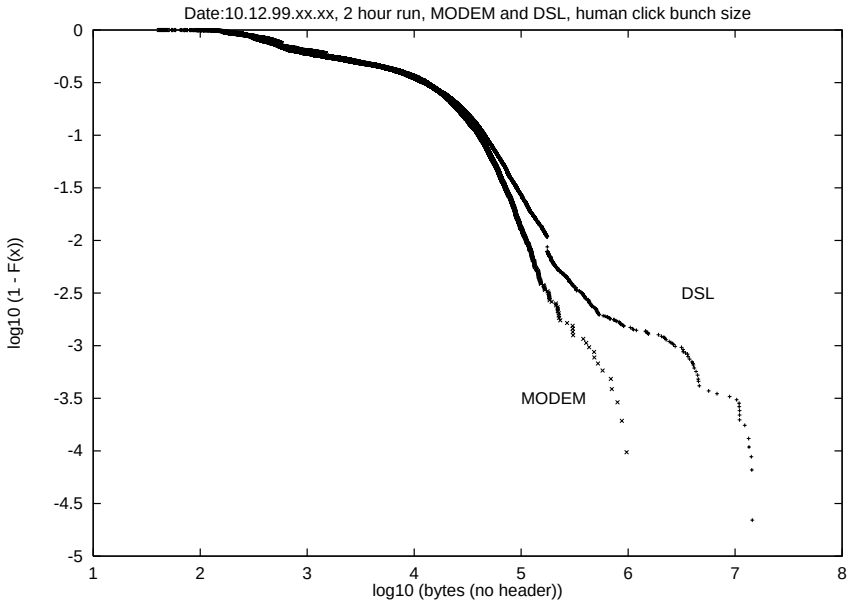

Figure 33: Human click bunch sizes for modem and DSL, log log plot

7.6 Total Bunch Sizes

Figure 33 gives the sizes of the “total bunch sizes” for both modem and DSL. These are the distributions of the total numbers of bytes that follow a human click, in the first file and later computer–requested files together.

8 Fit of Simulated and Real Data

Tables 22 and 23 give means and standard deviations for the simulated (computed) and observed values.

	mean	standard deviation
human thinks	50.6737	254.714
number computer gets	2.20636	6.48535
computer gets	2238.19	3579.77
human gets	9581.93	31034.5

Table 22: Observed data from modem customers

	mean	standard deviation
human thinks	49.0845	353.39
number computer gets	2.0158	5.18086
computer gets	2574.14	4872.15
human gets	8729.03	23679.7

Table 23: Computed values based on modem customers data

Table 22, together with the preceding figures with distributions, is a compendium of results on the observed data.

We see that for the “human gets” the data show higher variance than the simulations. Since the “best” theoretical distribution fitting the data has infinite variance, this could have come out either way.

9 Source Models and Provisioning

In this section we show how knowledge of customer behavior, in principle, can be used to determine how heavily a link can be loaded before there is impact, or serious impact, on the QoS received by the customers.

The source model used in this section is a model for behavior of HTTP customers and is based on the data described in previous sections. The source model will be explained in detail in the first sub-section. In later sub-sections, simulation will be used to determine the throughputs various customers can achieve under various load scenarios. Our philosophy is that any customer who is transporting a large file to his "client" computer wants the last mile access link to be the bottleneck. A customer who sees lower throughput than his or her access line allows will feel he or she is not getting what he or she paid for, and will be unhappy. Thus, the bandwidth of the access line is of importance.

In the simulations to be used, there essentially is a constant number of logged in customers, all surfing the web, all going through random thinktimes and random bunches of files. The fact that the number of WebSurfing customers is constant makes the achievable throughput unrealistically high. Further study is needed to find the additional back-off needed to protect against a varying number of connected customers, and against a varying number of customers that are connected but away for significant amounts of time, or who are doing other applications than WebSurfing. As long as much of the load is generated by dial–in customers with access lines of 128 Kbit/sec or less, the next step in the analysis depends on whether there is competition for dial–in ports. If there is no such competition, the number of connected customers is

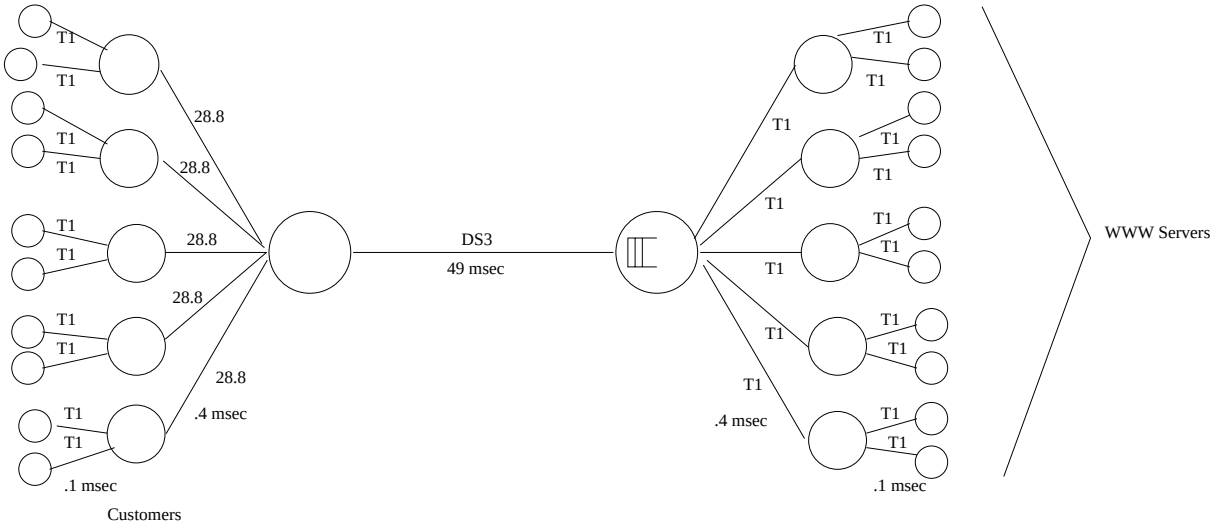

Figure 34: Network with DS3 Bottleneck, 0 persistent TCP connections

more variable than if there is port congestion. Thus we are justified in solving a simplified problem, that only shows how important availability of source models is.

9.1 The Simulation Model

Consider the network in Figure 34. In this figure there is a large number of "WebSurfers" (at the left). Each of those WebSurfers is connected through the Internet with one WebServer (at the right). We mimic HTTP 1.1 in that between the Surfer and the Server there can be two (pairs of) TCP connections. We simulate this by having two "TCP endstations" per client and per server: one for each possible TCP connection.

The first of these connections always transports the "human get" (the file explicitly requested by the human customer). The other TCP connection immediately starts on the computer gets. When (or if) the first connection finishes

the “human get”, it starts transporting computer gets, if there is any the other connection has not yet started on.

This, we feel, mimics the behavior of HTTP 1.1 with a maximum number of simultaneous connections equal to 2.

In the simulation, when the next human get is of type type 1, we wait until all previous file transfers are completed. Then we generate and wait one “human thinktime”. At this point we ran into a fairly serious complication. Due to memory constraints in our computers, and due to the fact that we need two TCP connections per client–server pair, we could simulate with at most 6000 client–server pairs. Hence, in order to generate sufficient load for our studies, we generated think times which are one tenth the size of those generated as in SubSection 7.2. This means that in our simulation the fraction of active (i.e. “non–thinking”) clients is too high, and the number of active clients is too close to deterministic.

This was the reason we felt at liberty not to look for a perfect fit in the thinktime distribution.

Next we generate one “human get size” from the fitted distribution, a “number of computer gets” from the fitted distribution, and the “sizes of the computer gets” from the fitted distribution. Next, we transmit the file of the human get over the first TCP connection of the client–server pair, and the files of the computer gets over both connections. We also draw the type of the next human get from the transition matrix 19. If this is type 2 or type 3, we draw human file size and number and sizes of computer gets, and immediately make them available for transmission.

We simulated this system with up to 6000 customers, i.e. 6000 clients,

6000 servers, 12000 TCP connections client → server, and 12000 TCP connections server → client. We did this “without further complications” and later with “further complications (in fact: additional connections). The results for the runs without further complications are not interesting by themselves, but will be presented to clarify the approach.

The results for the runs “without further complications” are first shown in Figures 35 - 38. We show Carried Load (on the bottleneck link), Loss Rate, “Normalized Load”, and “Offered Load” as function of the number of sources. “Offered Load” is the number of bytes (including headers) that arrives, from the servers, at the bottleneck port at the right, trying to get to the clients at the left. “Carried Load” similarly is the number of bytes actually carried. “Normalized Load” is essentially the same as carried load, only bytes (packets) that were transmitted more than once (due to the TCP Protocol) are counted only once. The “Loads” are given as fraction of the maximum the bottleneck can carry (fraction of the bottleneck linespeed). Somewhat surprisingly, the offered load never increases above the bottleneck linespeed: to have the “offered load” (carried load plus blocked load) even approach .9 of link capacity, the “inherent load” in terms of the number of active customers indicates an overload of (about) 1.5. At that load, packet loss is sufficiently large to keep even the offered load below the link bandwidth. We see that for high utilization offered load *is* larger than carried load. The difference is the lost packets.

One lesson from this is that measuring carried load or offered load can be quite misleading: there can be a wide range of different “inherent loads” that result in essentially the same carried and offered loads.

Another lesson is that for essentially constant offered load and carried load, traffic characteristics can change considerably, even if the traffic remains gen-

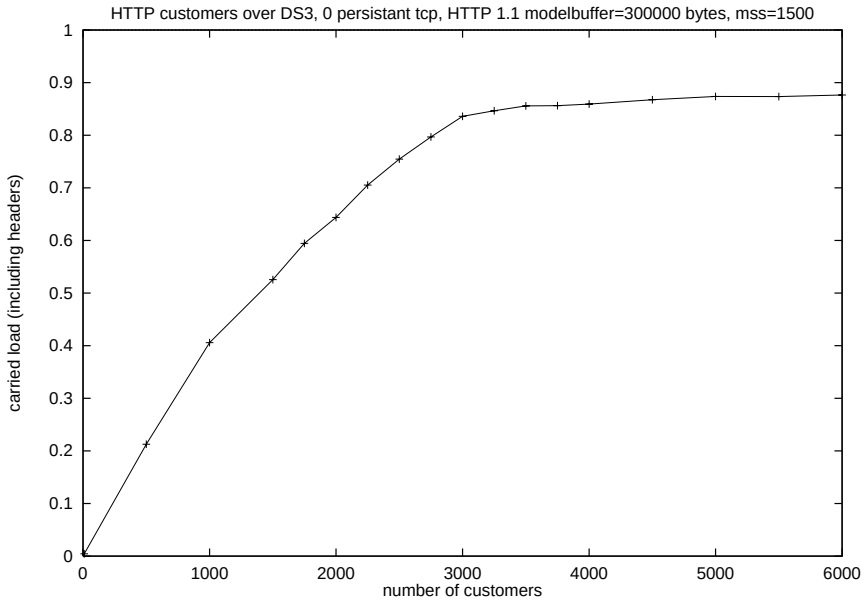

Figure 35: Carried Load as function of Number N of Connections

erated by similar customers. In Figures 35 and 38, if the number of "customers" increases from 3000 to 6000 the carried load and offered load remain essentially constant, but clearly average congestion window sizes are decreasing considerably, and hence "gross" traffic characteristics must be changing. It is quite likely that for example the Hurst parameter is different at 3000 customers than at 6000 customers. This is an interesting problem for future research. This kind of problem should be studied with more realistic thinktime distributions.

A third lesson is that once the utilization is sufficiently high, measuring carried load, or even carried load and offered load, does not give a clear picture of the level of congestion. Measuring carried and offered loads, while simultaneously measuring or computing the loss rate may give enough information.

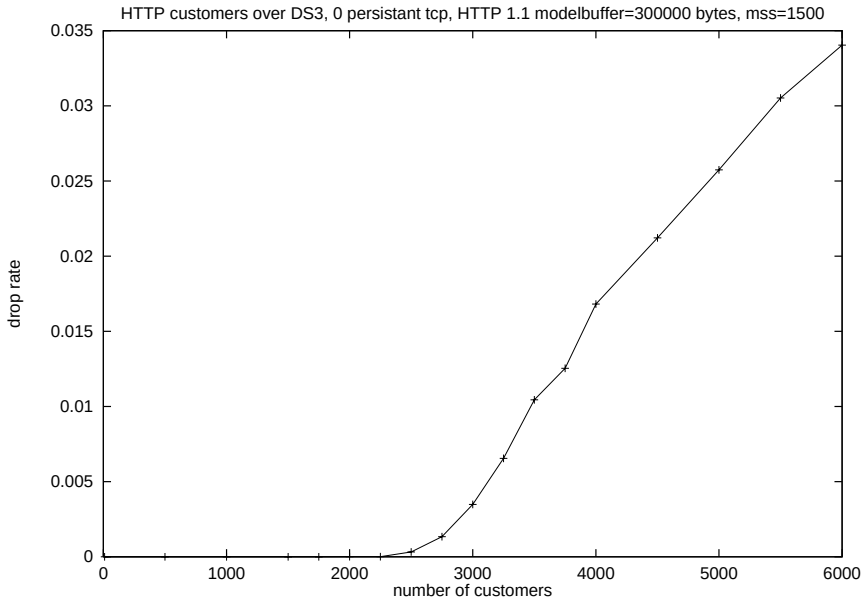

Figure 36: Loss Rate as function of Number N of Connections

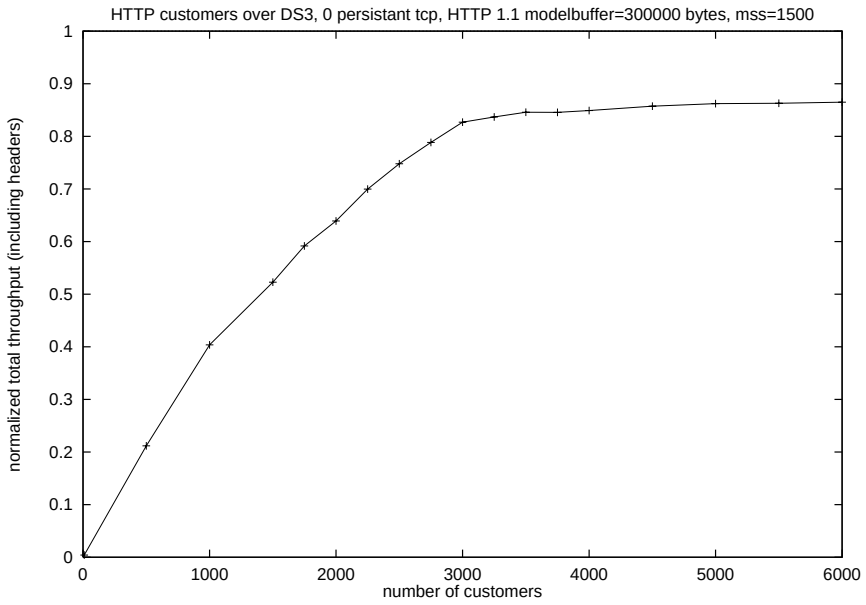

Figure 37: Normalized Throughput as function of Number N of Connections

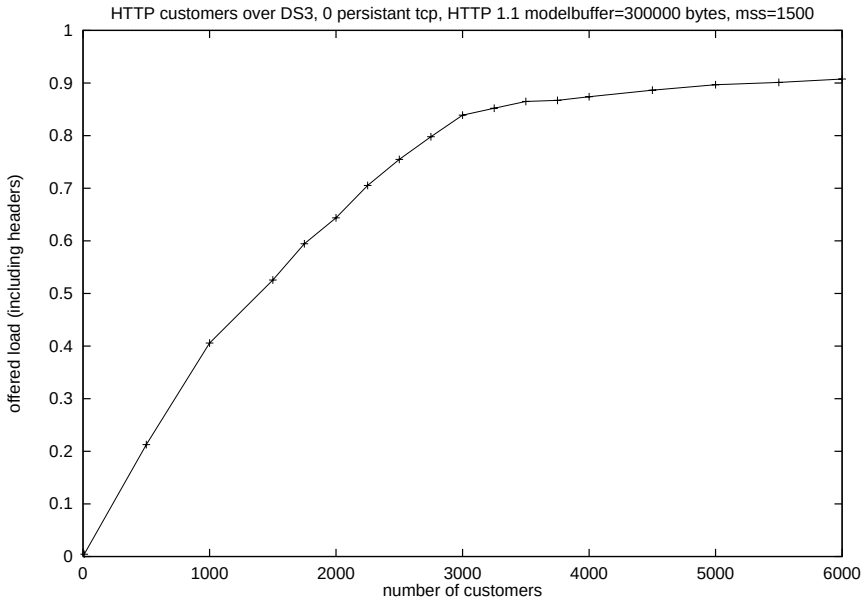

Figure 38: Offered Load as function of Number N of Connections

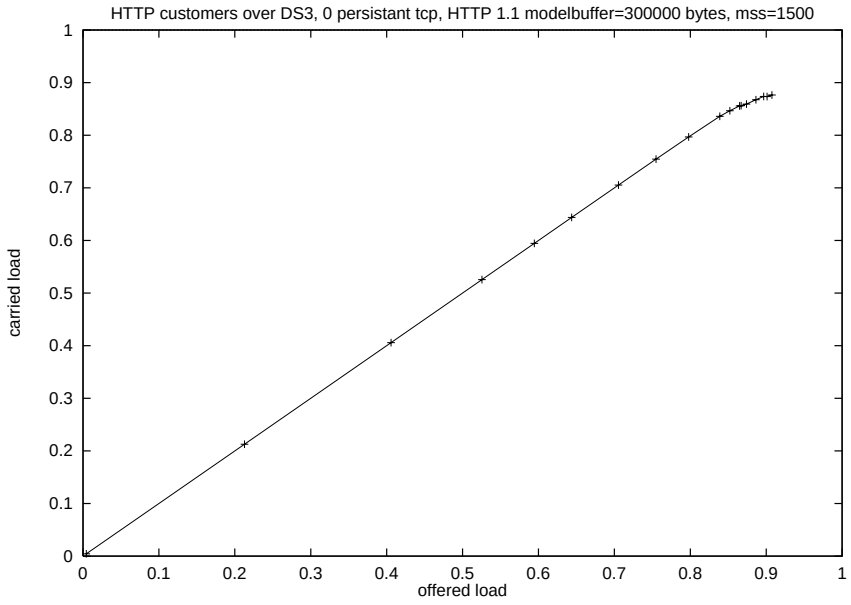

Figure 39: Carried Load as function of Offered Load

Next, we re-arranged the data, giving in Figures 39 and 40 for the same simulation runs the loss rate and carried load as function of the offered load.

Figure 41 gives the number of *active* clients (i.e. clients not in thinking mode) over time, for the simulation run with a total of 2000 customers. We see this number is too close to constant. We also see an interesting effect of choosing as theoretical thinktime distribution one with an infinite mean ($c_3 = .647 < 1$ in Equation (7.1)): Over time, a larger and larger fraction of clients gets hung up in a very long thinktime, and the number of active clients decreases over time.

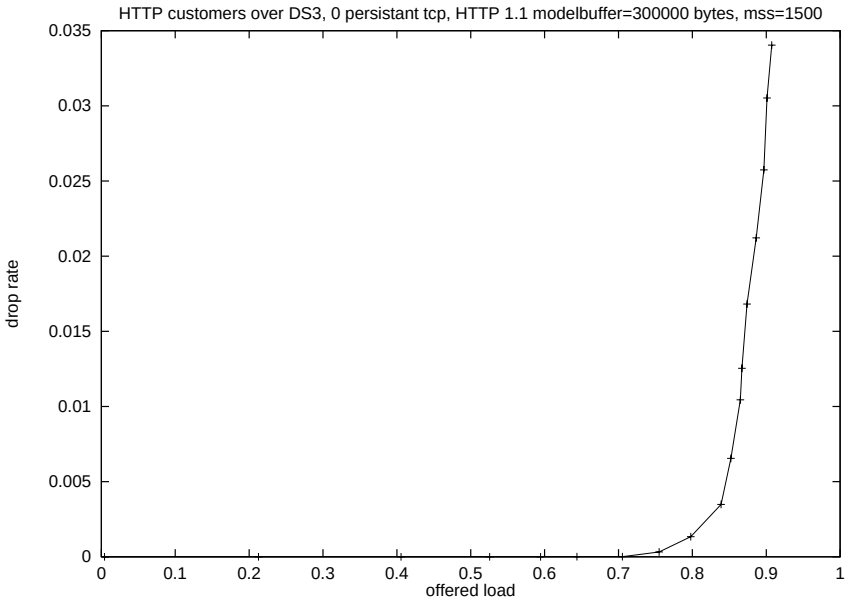

Figure 40: Loss Rate as function of Offered Load

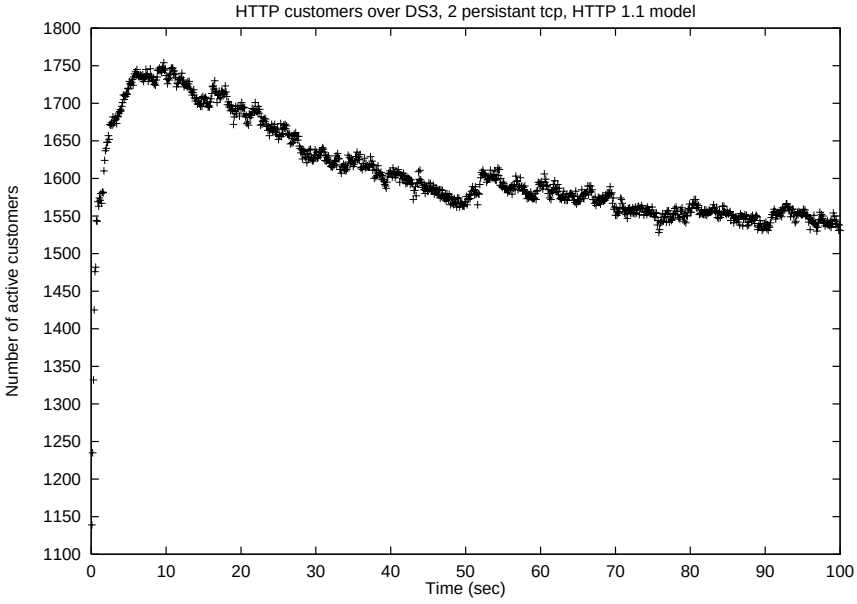

Figure 41: Number of Active Customers among 2000

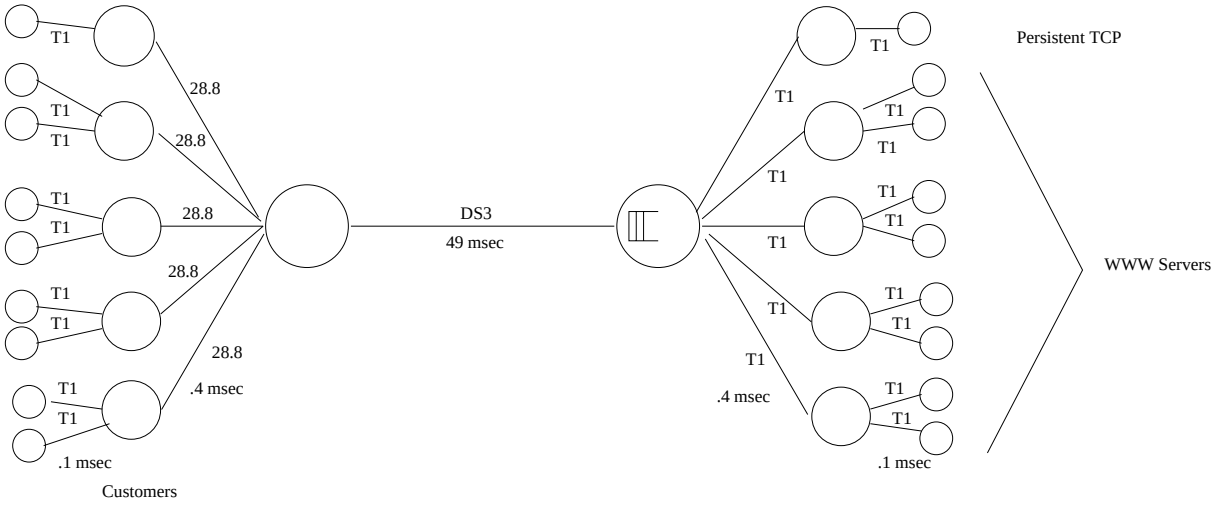

Figure 42: Network with DS3 Bottleneck, 1 persistent TCP connection

10 The Simulation Results

This section contains the actual simulation results. We study two situations: That with one additional ("persistent") connection, and that with two such persistent connections. In this context, a persistent connection is a TCP connection transporting an "infinitely" large file. This allows us to gauge the impact of a large number of "typical" HTTP WebSurfers on a single customer (WebSurfer or otherwise) who is moving a large file.

10.1 One Persistent Connection

The network used in this subsection is shown in Figure 42. In addition to a large number of regular HTTP WebSurfers, there is one customer with a persistent connection. That customer has a access channel of 28.8 Kbits/sec: the customer is a dial–in customer using a 28.8 Kbit/sec modem.

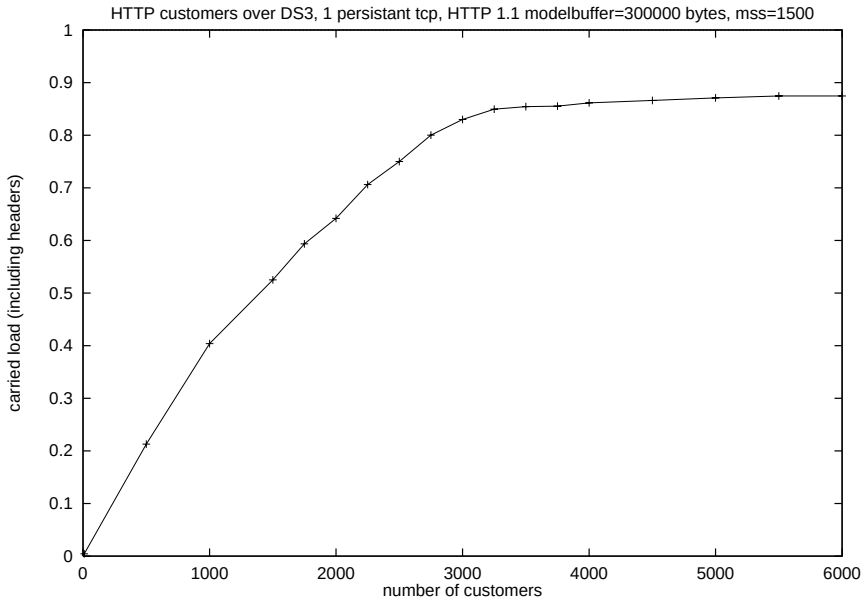

Figure 43: Carried Load as function of Number N of Connections

Figures 43 – 46 give carried load, loss rate, normalized throughput, and offered load as function of the number of simulated HTTP WebSurfers.

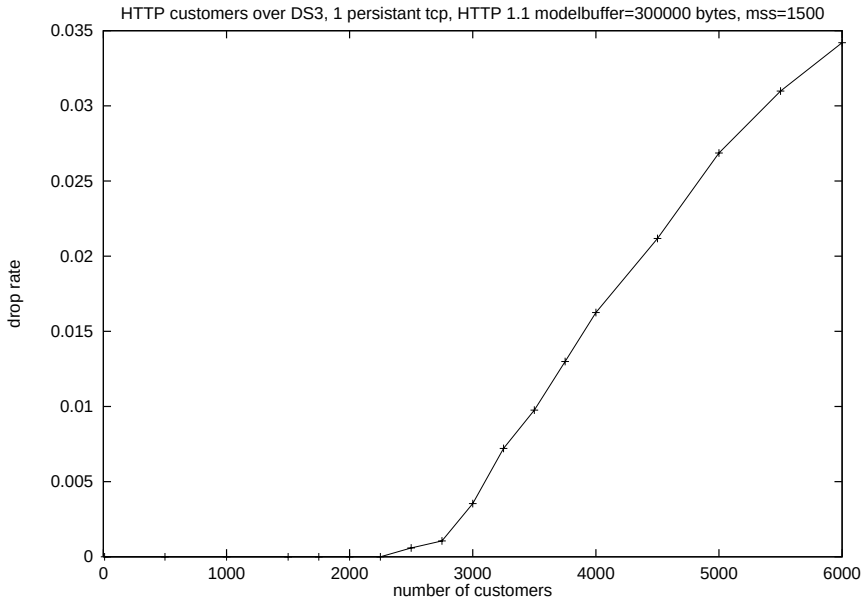

Figure 44: Loss Rate as function of Number N of Connections

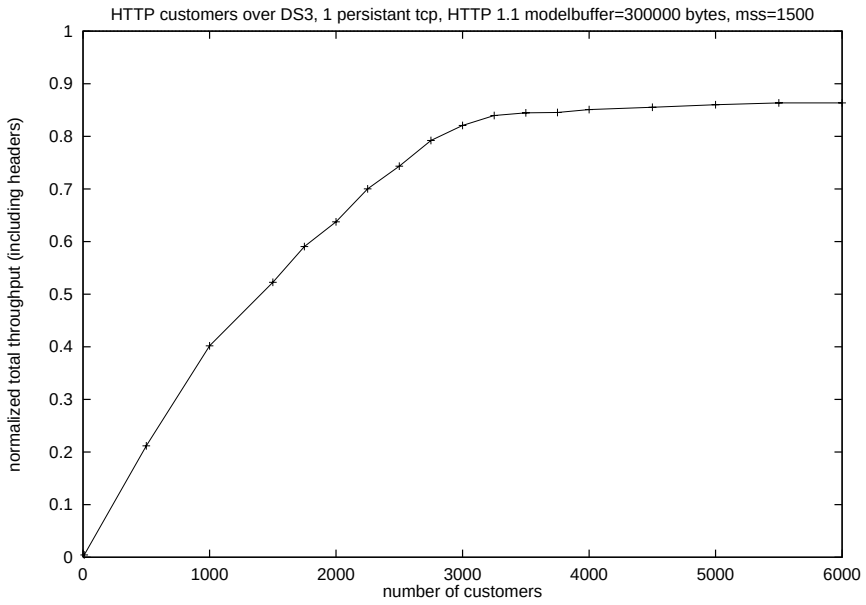

Figure 45: Normalized Throughput as function of Number N of Connections

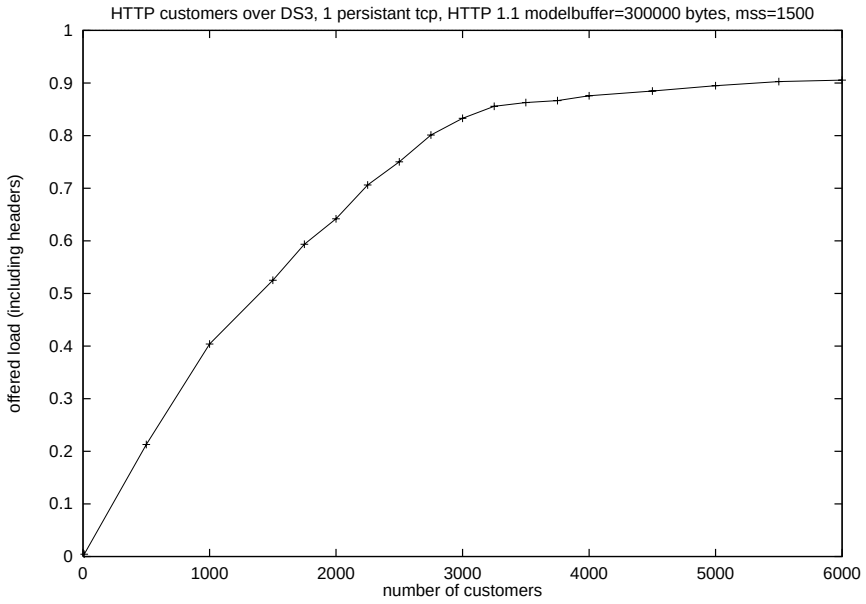

Figure 46: Offered Load as function of Number N of Connections

Next, we give for the same simulation results as in Figures 43 – 46 the carried load and lossrate as function of the offered load. The results are given in Figures 47 and 48. In Figure 47 we give carried load as function of (total) offered load. We give total carried load as fraction of the bottleneck bandwidth, and the carried load for the persistent 28.8Kbit/sec customer as fraction of 28.8Kbit/sec. To our surprise we see the latter carried load occasionally to be larger than 1 (!). This is of course physically impossible. The explanation is that we discard the first fifteen seconds of the simulation. The seemingly large carried load is due to the fact that at 15 seconds the buffer in the left shared router toward the client (receiver) of the persistent connection happens to contain a large number of packets, while at the end of the simulation that buffer is essentially empty. For the persistent connection, "carried load" is really "normalized load" and is measured at the destination.

We see that up to an offered load of over 85% of the bottleneck linkspeed, the persistent customer gets all the throughput his or her access line permits. We could interpret this as meaning this link can be loaded, in the peak fifteen minute of the week, up to 85% of the bottleneck linkspeed. This assumes we use as engineering requirement that "28.8Kbit/sec" type customers must almost always be able to get full use out of their access line.

While this specific conclusion is too optimistic (see arguments earlier in the section), a refined version of this approach will lead to insight into acceptable load levels. The fact that the throughput of the persistent connection recovers at higher loads is probably due to a synchronization effect in the simulation, and to randomly other connections encountering time-outs, thus freeing bandwidth for the test connection. This shows the need for care in interpreting results.

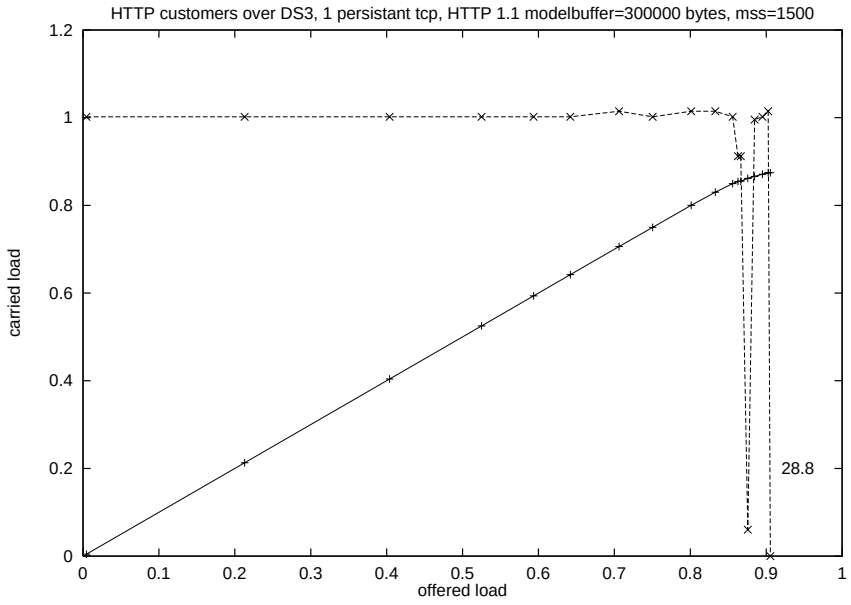

Figure 47: Carried Load as function of Offered Load

Figure 48 gives loss rate, separately for all sources together and for the persistent connection. We see that for high offered load the lossrate for a single connection can vary wildly.

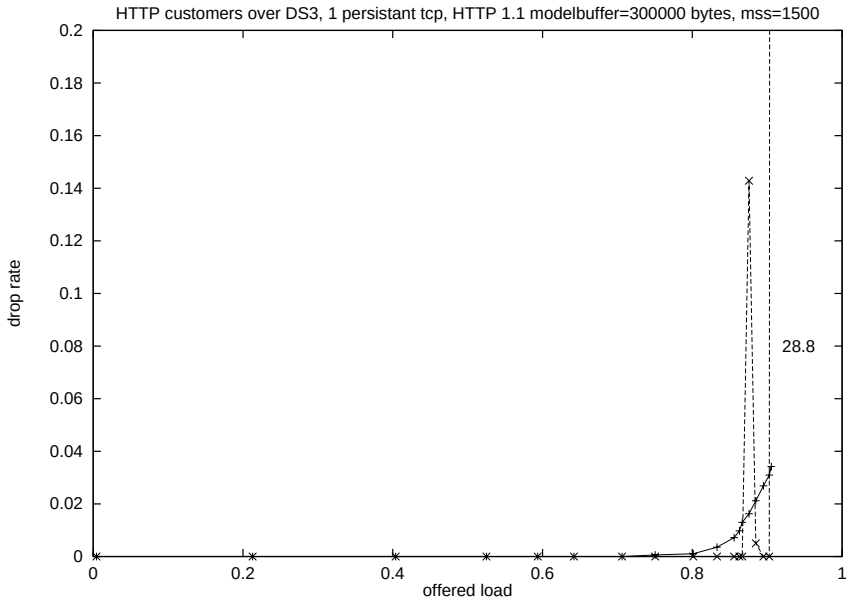

Figure 48: Loss Rate as function of Offered Load

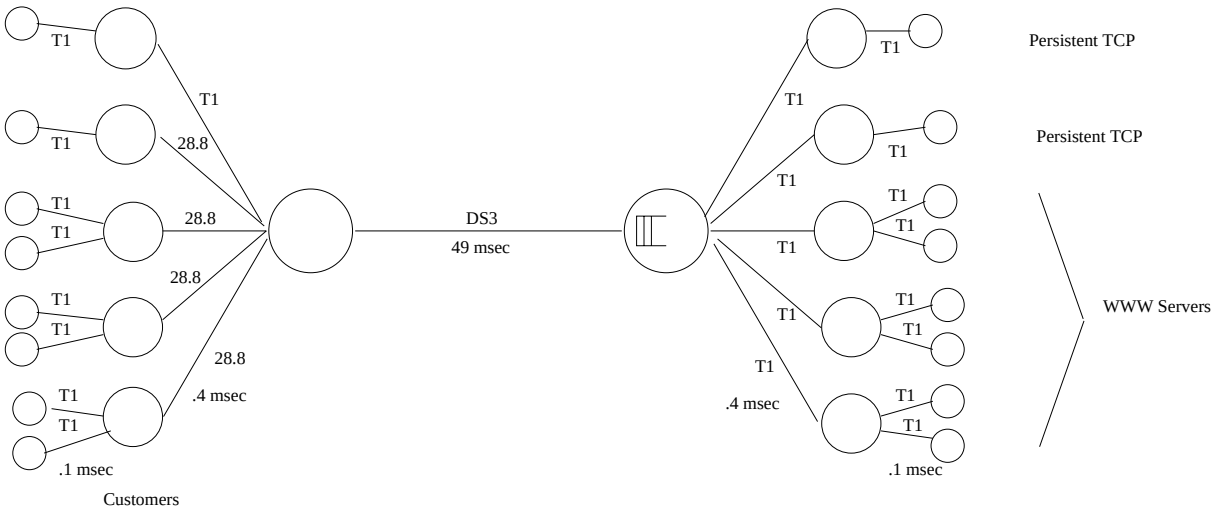

Figure 49: Network with DS3 Bottleneck, 2 persistent TCP connections

10.2 Two Persistent Connections

The network simulated in this subsection is shown in Figure 49. It has two persistent connections, one with an access channel of 28.8 Kbits/sec, the other with an access channel of 1.5Mbit/sec, say a DSL or cable modem access link.

In this network we do similar simulations as in the previous subsection, but now we only show, in Figures 50 and 51, carried load and loss rate as function of the offered load. In Figure 50 we show total throughput, throughput of the 28.8 persistent connection, and throughput of the T1 connection, respectively as fraction of the total bottleneck bandwidth, of 28.8 Kbit/sec, and of 1.5 Mbit/sec. Similarly, Figure 51 gives total loss rate and loss rates of the two persistent connections.

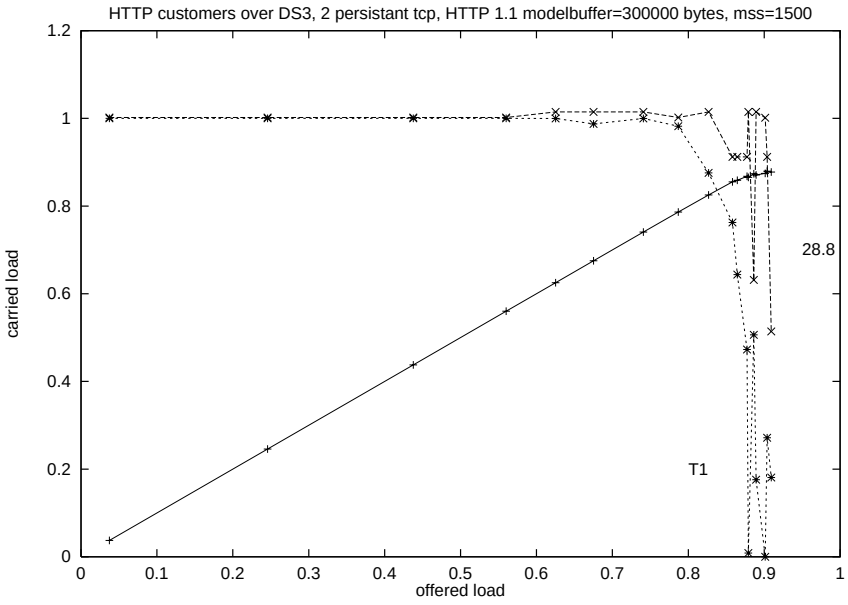

Figure 50: Carried Load as function of Offered Load

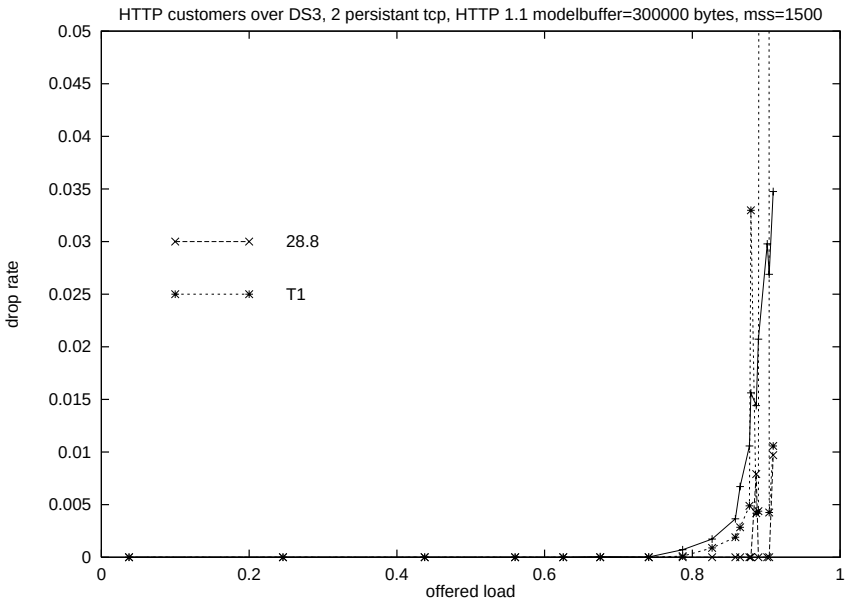

Figure 51: Loss Rate as function of Offered Load

In Figure 50 we see that the throughput of the persistent T1 connection collapses at a lower offered load than that of the 28.8 persistent connection. This is due to the fact the T1 connection, to get T1 throughput, needs a larger congestion window than the 28.8 connection does in order to get 28.8 throughput. A quantitative analysis is given in [6] (the square root formula for TCP). We see that maximal acceptable offered load is lower when we engineer for "T1" customers than when we engineer for "28.8Kbit/sec" customers.

It is interesting to note that in a previous (1998) study, see Tables 4 etc, we also fitted a source model to measurements and analogous to Figures 50 and 51 we then obtained Figures 52 and 53.

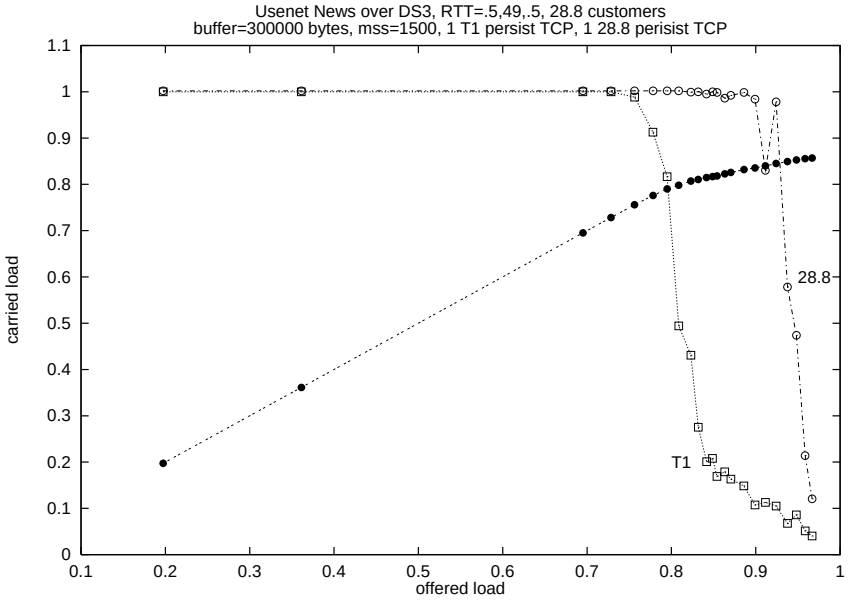

Figure 52: Carried Load as function of Offered Load, 1998 data

For the 1998 source model we notice that the throughput of the persistent T1 connection collapses for an offered load below 80%, while the throughput of the persistent 28.8 connection keeps going strong to about 90%. There clearly

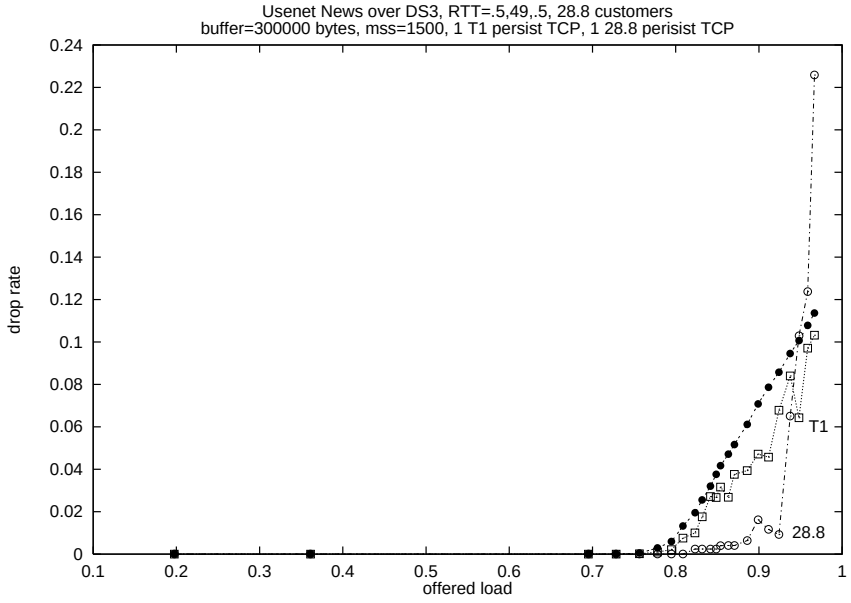

Figure 53: Loss Rate as function of Offered Load, 1998 data

is a difference between Figures 50 and 51 on the hand hand, and Figures 52 and 53 on the other. While it is possible the smaller difference between "loads of collapse" between T1 and 28.8 in 1999 than in 1998 is due to the unrealistically small thinktimes in the 1999 simulation, we still must conclude that the source model not only makes a difference for "safe operating levels", but also in the differential between QoS for customers with different access bandwidths.

11 Conclusions

On the basis of the measurements, our analysis, and the simulations we ran, we came to the following conclusions:

- Traffic generated by WebSurfers still is “heavy tailed”. In particular, the sizes of files requested directly by human customers have, for dial–in lines, a (tail) shape parameter $\sim 1.92 < 2$. For DSL customers the same distribution has a (tail) shape parameter even lower, but we have not determined it in this paper.

  The distribution of the number of “computer gets” per “human get” has, for dial–in customers, a (tail) shape parameter of 8.359, so that any well-fitting theoretical distribution has finite first and second moments. For DSL lines the similar distribution has a lower shape parameter. It is conceivable that distribution is heavy tailed, we did not verify this.

  We did not determine the shape of the distribution of the number of “human gets” per WebSurfing session. It is clear that the average number of human gets per WebSurfing session is large. It is quite possible that this gives another source of long range dependence.

- Our simulation results indicate that use of adequate source models can indeed lead, with simulation, to insight into maximal acceptable load levels on specific links, in say the busy 15 minutes of the week. However, at this point simulation technology has become a bottleneck.

- There is need for further work on “Large Scale Simulation”. One of the problems with our simulation runs was that we could not include sufficiently many sources to create realistic heavy loads on (say) a DS3 link, at least not without making thinktimes unrealistically small.

  Such research in large scale simulation technology must take into account that most of the “other” (say background) traffic has closed loop control (TCP), so that the traffic characteristics depend heavily not only

on the actual utilization of the link, but also on something we call the "inherent load", see SubSection 9.1. This is illustrated by the results in SubSection 9.1.

- Knowledge of the bandwidths of the access lines of the customers is important in engineering of the network, (probably) mostly on the engineering of links relatively close to the edge of the network, such as links internal to ISPs and links between ISPs and backbone carriers.

- While for most or all "routine" traffic monitoring use of "sniffers" that only produce byte counts and packet counts of certain classes of flows over say 15 minuter periods is adequate and probably preferable, use of tools like TCPDump that collect entire traces is necessary to build reliable source models. They also are a useful tool in estimating the distribution of access bandwidths of active customers.

- We saw that access bandwidths of customers are important for engineering. This is true for several reasons. The one discussed in this paper is that higher bandwidth customers need larger congestion windows to fully utilize their access lines, and therefore need lower loss rates. Another relevant factor is that larger bandwidth customers generate wilder fluctuating traffic. This effect was not studied in this paper. We showed that the kind of measurements we used in this paper can be used to estimate the access bandwidth distribution of active customers.

An issue not addressed in this paper is traffic forecasting. It is to be expected that stratified forecasting (seperate forecasts for several categories, based on customer type, application, and possibly source and destination

(sub)–networks) will improve forecast accuracy. Stratification will certain facilitate tracking of forecasts and recognizing shifts in trends. The type of analysis done in this study is an integral part of traffic stratification.

Appendix A An old WWW model

For some time we have been using a Source Model for WWW traffic that is based on the data obtained in 1995 at Boston University by taking traces of Web Accesses by students. This set of measurements we henceforth denote as the "Bestavros and Crovella" data. We call the model the "Pseudo–Web Model". Because of the time elapsed since the measurements were taken, the model must be considered out–dated. The model described in the Sections ... above for the time being is a valid replacement. The old model still is a useful reference model, and is given below. Changes must be due to many factors, including more modern Browser Technology, a different test population, more servers with a larger choice of Web pages, more graphics, etc.

More on the changing nature of network traffic can be found in [4] and [5]

We now call our old model for WWW traffic the "Pseudo – Web" model.

In the Pseudo – Web model, a source goes through a thinktime with a given distribution (in seconds, see below) and then sends a file of which the size (the number of bytes) follows a different given distribution, see below. Once the last byte of the file has been acknowledged, the source resets its Congestion Window *cwnd* to one MSS (Maximum Segment Size) and its Threshold *ssthresh* to 64 KBytes, and when the next file is ready to be transported (one think time later) it starts in "slowstart". The simulation does not do the SYN and FIN

packets of real TCP.

Thinktime distributions and Filesize distributions were obtained from a study of the Bestavros and Crovella data . This work was done mostly by our colleague Arnold Neidhardt at Telcordia.

The Bestavros and Crovella data were obtained by monitoring the behavior of students at Boston University who were given free access to the Web.

For file sizes in the Bestavros and Crovella data we found a simple distribution (F denotes file size, in bytes):

$$P\{F > f\} = (1 + (\frac{f}{\mu})^{\alpha})^{-1}, \text{ where} \qquad \text{(Appendix A.1)}$$

$$\mu \text{ is the median,}$$

$$\alpha > 1,$$

$$\mu \times \alpha^{-1} \times \Gamma(\frac{1}{\alpha}) \times \Gamma(1 - \frac{1}{\alpha}) \text{ is the mean.}$$

If $0 < \alpha \leq 1$ Appendix A.1 still is a probability distribution, but that distribution has no first moment.

For the Bestavros and Crovella data we found

$$\mu = 2190,\ \alpha = 1.15066\,.$$

Hence, the distribution Appendix A.1 has a finite mean but infinite variance.

The mean of the distribution Appendix A.1 is 14,954.22 bytes.

For the “thinktimes” T (in seconds) we found the distribution

$$P\{T > t\} = p_h(1 + (\frac{t}{\mu_h})^{\alpha_h})^{-1} + p_l(1 + (\frac{t}{\mu_l})^{\alpha_l})^{-1}, \qquad \text{(Appendix A.2)}$$

where

$$p_h + p_l = 1, \text{i.e. a mixture of two distributions as in AppendixA.1, and}$$

$$p_h = 0.4953916,\ p_l = .5046084$$

$$\mu_h = 10,\ \alpha_h = 1.243437,$$

$$\mu_l = 0.245032,\ \alpha_l = 3.252665.$$

The distribution in Appendix A.2 etc is a mixture of two distributions, one ("h" for high) long tailed and one ("l" for low) with a finite variance and even a finite third moment. It is attractive to believe that the "l" thinktimes occur when the computer (actually: the browser) chooses the next file to be downloaded while the "h" thinktimes occur when it is the human customer who makes the decision. A further discussion of when "thinktimes" are generated by humans and when they are generated needs understanding of how browsers work, this leads us too far afield.

The mean of the distribution Appendix A.2 is 21.83705 seconds. The two distributions that make up the mixture have expected values $E[T_h] = 43.78729$ seconds, resp $E[T_l] = 0.2877263$ seconds.

Appendix B List of Applications

The following is a list of "Applications" (such as WWW, NNTP, FTP etc) that occurred in either the measurements used in this document or in other re-

cent measurements, with brief indications of portnumber used and further documentation available.

- **TCPMUX**
  TCPMUX is a protocol to contact multiple services on a single well-known TCP port using a service name instead of a well-known number. In addition, private protocols can make use of the service without needing an official TCP port assignment. TCP port 1. See RFC 1078 for details.

- **FTP** (File Transfer Protocol)
  This is the protocol for (anonymous or not) transfer of files between users. TCP ports 20 and 21. See RFC 765 for details.

- **TELNET (Telnet Protocol)**
  Telnet's primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other, TCP port 23. See RFC 854 for details.

- **SMTP** (Simple Mail Transfer Protocol)
  TCP port 25. See RFC 821 for details. This is the mail protocol between mail servers.

- **WHO IS**
  Also known as NICNAME, TCP port 43. See RFC 812 for details.

- **NI-FTP (NI File Transfer Protocol)**
  The NI File Transfer Protocol uses TCP port 47. We never saw this in any of the data we used in this document.

- **DNS** (Domain Name System)
  This is the distributed Name/Address mechanism in the Internet. TCP and UDP port 53. See RFC 883 for details.

- **WWW-HTTP** (World Wide Web)
  This is the well known Web traffic. It uses only TCP port 80. See http://www.w3.org for details.

- **HOSTS2-NS (HOSTS2 Name Server)**
  The HOSTS2 Name Server uses TCP port 81.

- **POP3** (Post Office Protocol, version 3)
  This is a protocol for fetching Email from a remote mailbox. It is the protocol used when a user get its mail out of a mailserver. TCP port 110. See RFC 1225 for details.

- **NFS (Network File System)**
  A distributed file system developed by Sun Microsystems which allows a set of computers to cooperatively access each other's files in a transparent manner. TCP and UDP on RPC port 111. See RFC 1094 for details.

- **AUTH** (Authentication Service)
  This provides a means to authenticate the identity of a user of a TCP connection. TCP port 113. See RFC 931 for details.

- **NNTP** (Network News Transfer Protocol)
  This is a protocol for distribution, inquiry, retrieval, and posting of news articles. It uses TCP port 119. See RFC 977 for details.

- **NTP (Network Time Protocol)**
  NTP provides the mechanisms to synchronize time and coordinate time

distribution in a large, diverse internet. UDP port 123. See RFC 1305 for details.

- **Net Bios** (Network Basic Input Output System)
  This is a protocol designed for groups of PCs, sharing a broadcast medium. It uses both TCP and UDP, port 137. See RFCs 1001 and 1002 for details.

- **SNMP** (Simple Network Management Protocol)
  UDP ports 161, 162, 164, 165. See RFC 1448 for details.

- **HTTPS** This is the “secure” version of HTTP. TCP portnumber 443. See the Internet Drafts on Transport Layer Security at http://www.ietf.org for details.

- **Route**
  UDP port 520. This is a variation on the Xerox NS Routing Information Protocol (RIP).

- **ICMP (Internet Control Message Protocol)**
  ICMP is the protocol used to handle errors and control messages at the IP layer. ICMP is actually part of the IP protocol. The port varies. See RFC 792 for details.

- **SYSLOG**
  Syslog log file, UDP port 514.

- **NIM**
  TCP port 1058. Network Installation Management (IBM).

- **Lotus Note**
  TCP port 1352.

- **RGTP**

  TCP port 1431. Reverse Gossip Transport.

- **NKD**

  TCP port 1650.

- **Elvin**

  TCP port 2916. Push News.

- **ESRI_SDE**

  UDP port 5151. Spatial Database Engine.

- fcp-addr-srvr1

  TCP port 5500.

- **IRC**

  TCP port 6667. Internet Relay Chat, see http://www.mirc.com for details.

- **Real Audio**

  UDP port 6970.

References

[1] Cunha, C.R., Bestavros, A, and Crovella, M.E. (1995) “Characteristics of WWW Client-based Traces”. Tech. Report BUCS-TR-95-010, Boston University, CS Dept. Boston, MA.

[2] Almeida, V., Bestavros, A., Crovella, M.E. and de Oliveira, A. (1996) “Characterizing Reference Location in the WWW”. *Proceedings of of PIDS:* The IEEE Conference on Parallel and Distributed Information Systems, Miami Beach, Florida. December 1996.

[3] Crovella, M.E. and Bestavros, A. (1997) “Self Similarity in World Wide Web Traffic: Evidence and possible causes”. *IEEE/ACM Transactions on Networking*, December 1997.

[4] Feldmann, A., Gilbert, A.C., and Willinger, W. (1998) Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic. *Computer Communication Review*, Vol. 28, No. 4, *Proc. of the ACM/SIGCOMM’98*, Vancouver, Canada, pp. 42-55, Sept 1998.

[5] Feldmann, A., Gilbert, A.C., Willinger, W. and Kurtz, T.G. (1998) The changing nature of network traffic: Scaling Phenomena. *Computer Communication Review*, Vol. 28, No. 2, pp. 5-29, April 1998.

[6] Ott, T.J., Kemperman, J.H.B., and Mathis, M. (1996) The Stationary Behavior of Idealized TCP Congestion Behavior.
ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps