A Measurement Study of Internet File Transfer Traffic

David J. Ewing
Richard S. Hall
Michael F. Schwartz

CU-CS-571-92 January 1992

Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309
(303) 492-3902
{ewing,rshall,schwartz}@cs.colorado.edu

Abstract

The lack of a topology-based information distribution mechanism in wide area networks causes users to waste a large amount of bandwidth on repeat transfers of widely accessed files. As a first step towards designing such a mechanism, in this paper we present measurements of the Internet File Transfer Protocol, gathered over two weeks at the main gateway to the University of Colorado network. We found that nearly 40% of all transfers were duplicate transmissions, accounting for over 54% of the file transmission traffic. A small proportion of files were significantly larger and more frequently transferred, with 6.30% of the files accounting for 19.30% of the transfers. 8.21% of duplicate file transmissions were caused by user errors when transferring binary data, underscoring the need to insulate users from such details. We also found that file transfers occurred in only 44% of the FTP connections. Other connections were probably directory-only requests, underscoring the need for better resource discovery support in the Internet. We present measurements of a number of other statistics as well, including distributions of file sizes, file types, peak transfer times, and sources and destinations.

1. Introduction

In recent years the number of hosts on the Internet has increased dramatically, causing concomitant increases in network traffic. A substantial portion of this traffic is due to the large number of files that are transferred around the global Internet using the File Transfer Protocol (FTP) [Postel & Reynolds 1985]. Some files get transferred multiple times between many different sources and destinations, without regard for network topology. We propose that a topology-based distribution and caching mechanism be provided for the Internet, to more efficiently support disseminating and searching large amounts of widely shared information.

As a motivation, consider the problem of large software releases. When MIT released the latest revision of the X window system (X11R5), they manually arranged to have the data available by "anonymous FTP" from over 20 sites around the world, to help distribute Internet load. Users then chose among the copies, using simple heuristics (for example, preferring to retrieve files from geographically nearby sites). Unfortunately, as networks become increasingly complex, manually chosing where to place and retrieved shared data becomes difficult. Optimizing for network bandwidth, cost, and other factors will need support from the network layers responsible for routing, flow control, accounting, and policy considerations. A similar problem arises when distributing widely accessed documents [Malamud 1991, Postel 1982], directory information [CCITT 1988, Danzig et al. 1991, Schwartz 1990], and generally in any circumstance where a large volume of information is circulated among many recipients. Caching becomes particularly appealing if the information contains a moderate sized subset that is required at a large proportion of sites. For example, we conjecture that the popular "Archie" anonymous FTP directory service [Emtage 1991] answers a large proportion of its search requests from a small proportion of its database. If that is the case, distributing and caching copies of the heavily accessed parts of the database could allow the system to operate much more scalably than it presently does, without forcing the data to be structured in a way that limits how it can be searched.

Distributing and caching information according to network topology and usage patterns can potentially reduce network load substantially, since data transport activities typically account for the majority of bytes transmitted across operational networks [NSFNET Service Center 1991]. Beyond files and directory information, the mechanism could be used to support other forms of information dissemination, such as network news and mailing lists. As a longer term goal, the mechanism could underly a network-transparent mode of resource sharing, whereby resources are named and accessed independently of the particular hosts on which they reside.

The details of the proposed mechanism are still being worked out [Schwartz 1991a]. To help establish the need for the mechanism and to better understand what types of information access it should optimize for, we have begun a series of network measurement studies. In the current paper we present measurements of file transfers between the University of Colorado and the rest of the Internet, to determine the extent of duplicate transmissions. To carry out this study, we implemented software to monitor Internet packets using Sun Microsystem's Network Interface Tap, and tracked the names and access counts of the 10,000 most frequently transferred files from December 6-19, 1991.

The data we collected clearly raise privacy concerns. We treat all of the data as private information, publishing measurements only in global statistical terms, divorced from the actual sites that make up the underlying data points.

In Section 2 we discuss related work. In Section 3 we discuss our data collection methodology. In Section 4 we present our results. We offer our conclusions in Section 5.

2. Related Work

A number of studies have measured lower layer network characteristics such as packet traffic and protocol usage [Heimlich 1990, Horvath 1990, NSFNET Service Center 1991]. In contrast, few measurement studies have considered aspects of operational wide area networks above the level of the network routing and transmission control. To the best of our knowledge, no measurements have been made of Internet file transfer activity.

A number of studies have passively monitored wide area networks to collect data. Jain and Routhier measured network traffic to characterize their packet train traffic model [Jain & Routhier 1986]. Caceres et al. performed similar measurements, but monitored particular applications instead of lower level traffic [Caceres et al. 1991]. Pu et al. developed a methodology called Layered Refinement that measures end-to-end performance and availability in a wide area internet affected by a large volume of concurrent usage, and by evolution in the underlying hardware and software [Pu, Korz & Lehman 1991]. Schwartz and Wood collected data about electronic mail traffic, to analyze the organizational structure that arises naturally when people communicate, and the extent to which one can derive clues about this structure from a communication graph [Schwartz & Wood 1991].

Several other studies have used directed probes to collect measurements. Lottor used software that recursively descended the Internet Domain Naming tree, using "zone transfers" to retrieve information about a large number of Internet hosts, to measure characteristics such as the number and top-level domain distribution of hosts, the number of hosts running various operating systems, and popular host names [Lottor 1990]. Long et al. probed over 100,000 hosts twice over a period of several months using Sun RPC [Sun Microsystems 1988] and ICMP [Postel 1981] echo requests, to test standard assumptions about the probabilistic distribution of host failure and repair rates [Long, Carroll & Park 1990]. Mills probed over 100,000 hosts to measure the timing accuracy of ICMP, TIME, and the Network Time Protocol [Mills 1990]. Schwartz is carrying out a longitudinal measurement study of changes in service-level reachability in the global TCP/IP Internet, to uncover changes in how sites resolve the problem of supporting open network usage while ensuring reasonable security [Schwartz 1991b].

3. Measurement Methodology

Several complications arose in gathering the data for this study. The first complication was FTP’s use of separate communication ports for control and data. Since each file transfer uses a different data port number, the control packets had to be gathered and parsed from one port while monitoring data on other ports.

The second complication was attempting to limit resource consumption. Keeping track of all file transfers over a long period of time requires a large amount of storage. Therefore, we used statically allocated tables, and dumped the information to disk once per hour. While packets can be lost during disk dumps, in practice this turned out not to be a problem. Moreover, having hourly dumps allowed us to plot various time-series distributions of the measured data.

The third complication was keeping pace with network traffic. Although the measurements were collected on a dedicated, reasonably powerful machine (a SparcStation IPC with 24 megabytes of RAM and a local disk), in our measurement configuration we saw up to 150 FTP packets per second. We used in-memory hash tables to minimize administrative overhead. Even so, with this traffic rate some packets were dropped because of kernel buffer overflows. We discuss how dropped packets affect the data in Section 4.

The final complication was keeping track of repeat files. To do this it was necessary to have some way to identify particular files. It is not possible to use file names, since files are not named uniformly across Internet hosts, and relative names are not even unique across a single node. Instead, we used the file size and twenty bytes selected from throughout each file, sampling every $\frac{n}{20}$th byte for a file of size n. This *file signature* algorithm is more robust than using file names. Given two files of size n that differ by m bytes[1] and sampling s bytes from each file, the probability of incorrectly identifying two files as matching when they actually differ is

$$P(incorrect\ \ match) = \frac{\binom{n-m}{s}}{\binom{n}{s}}$$

Figure 1 plots this function against the percentage by which two files differ ($100\frac{m}{n}$), for $n = 10{,}000$ and three values of s. The curves shift very slowly up as n increases. For example, for 10% file difference and $s = 20$, $P(incorrect\ \ match) = .095$, .121, and .122 for $n = 100$, 10,000, and 1,000,000, respectively. Although the probability of an incorrect match with $s = 20$ is significant for files for which the percentage difference is less than 20%, in practice it is unlikely that two different files will have the same size and differ by fewer than 20% of their bytes.

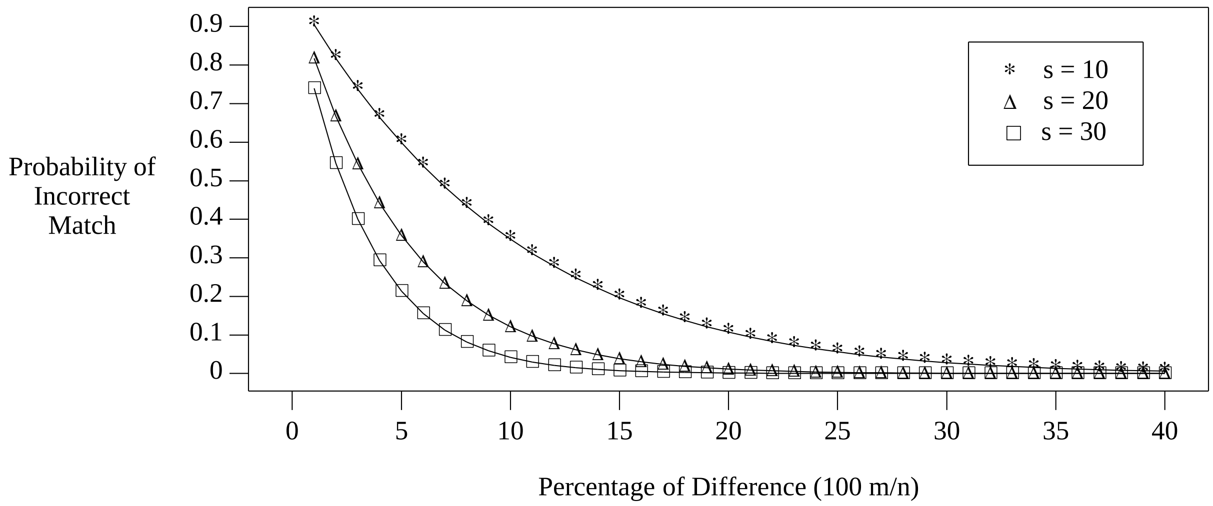

Figure 1: Probability of an Incorrect Match

To test the likelihood that two different files of the same size might differ by fewer than 20% of their bytes, we compared 5,062 pairs of randomly chosen same size files from one of the authors' workstation. In 210 (4.15%) of the cases, the file pairs had different names but identical contents (which would be correctly identified by our signature algorithm). On average, pairs of non-identical files differed by 86.06% of their bytes. 279 (5.51%) of the non-identical pairs differed by fewer than 20% of their bytes. 259 of these pairs (5.12% of the total) were UNIX[2] executable programs (which we found are not transferred very often by FTP; see Table 3). The remaining 0.40% of the pairs consisted of files that had been copied from one directory to another and modified slightly at some point. This final category represents the files most likely to be incorrectly indicated as matching by our signature algorithm. As an interesting side note, the fact that UNIX executable programs differ by so few bytes indicates that a large amount of disk space is wasted on the average UNIX file system, even though the measured executables used shared libraries.

---

[1] Bytes must be in the same position and have the same value to be considered the same.

[2] UNIX is a trademark of AT&T Bell Laboratories.

4. Results

Raw Data Statistics

Table 1 summarizes the raw data. A non-trivial amount (12.34%) of network traffic was caused by protocols other than IP. Yet, 75.22% of the total network traffic was caused by TCP/IP packets, of which 18.62% (14.00% of total packets) were FTP packets. We return to these figures in Section 5, where we use them to extrapolate our results to the NSFNET backbone.

Execution Time	13.9 days
Total Packets	369,581,096
Number of Packets Dropped by Network Interface Tap	159,206
Number of IP Packets	323,830,604
Number of TCP Packets	277,873,134
Number of FTP Packets	51,735,135
Number of FTP Bytes	8,321,000,202
Number of FTP Connections	89,230
Reset FTP connections	2,595
Average Duration of Connections	83.2 sec.
Number of Files Transferred	39,324
Number of Files Dropped	4,141
Number of Files of Unknown Size	822
Number of Different Hosts Involved in Transfers	≥ 2,125[4]

Table 1: Raw Data Statistics

The Network Interface Tap dropped 0.04% of the packets that passed by the host, because of kernel buffer overflows while scheduling the application level data collection process. Since 14.00% of the packets were FTP packets, the percentage of traffic caused by FTP has a maximum error of ±0.01%. The lost packets caused the monitoring software to be unable to detect the identities of 4,141 files (10.53%), because it missed some packets needed by the file signature algorithm described in Section 3. The file loss rate is much higher than the packet loss rate because bursty traffic can cause several packets in a row to be dropped, increasing the chance that the monitoring software will miss a needed sampling point (and hence an entire file's signature). Since running this study we have modified the software so that it samples 32 bytes, and only requires that a subset of size 20 be present, to increase resilience to dropped packets. We will use this new algorithm for future measurements. These modifications were intended primarily for monitoring a network with more traffic (such as the NSFNET backbone). Losing information about a moderate number of file transfers does not affect the validity of the study, since the collected data is only a sample of all traffic in any case.

To determine file size, the monitoring software looked in the packet containing the FTP server's response to a client's file retrieval request. Because this is not a standard feature of FTP, the monitoring software was unable to determine the sizes of 822 files. It might have been possible to determine file size by looking at packet sequence numbers, but we did not attempt to do that, since our method worked in 97.91% of the cases.

During data collection, 89,230 FTP connections were monitored, the average duration of which was 83.2 seconds. Since the average file transfer size 211,601 bytes (see Table 2), the average throughput was 20.35k

[4] For duplicate transfers, only the first source and destination IP addresses were recorded.

bits/second. Interestingly, only 44% of the FTP connections resulted in a file transfer. Other connections either experienced network problems, or performed non-file retrieval operations, such as directory retrievals. Since connection resets happened in only 2,595 (2.91%) of the FTP connections, network problems were probably not the reason for this low average. Directory listing-only requests probably accounted for most of the non-file retrieval operations. One wonders how much of this traffic is due to "Archie"-like FTP directory services, which automatically and periodically retrieve directory listings of many "anonymous FTP" servers around the Internet [Emtage 1991]. Even if the directory list-only requests were done manually (by users exploring FTP sites), this statistic underscores the need for better resource discovery support in the Internet.

Duplicate Transmission Measurements

Table 2 presents statistics on duplicate file transfers. Over the monitoring period, 39,324 transfers of 23,622 different files were observed. Of these files, 4,982 (21.10%) files were transferred more than once, resulting in 15,702 (39.93%) transfers of files that had previously been transferred. Looking at duplicate transfers by file size rather than count, 4,523,742,089 (54.37%) of the 8,321,000,202 total bytes were transmissions for files that had previously been transferred. Figure 2 shows that these percentages represent steady state averages. Given that FTP accounted for 14% of the network traffic by packet, duplicate transfers accounted for approximately 7.61% of the packets transmitted. This is approximate, since not all of FTP's traffic is caused by file transfers.

Total File Transfers	39,324
Number of Transfers of Files Previously Transferred	15,702
Number of Different Files Transferred	23,622
Number of Files Transferred More Than Once	4,982
Total Bytes Transferred	8,321,000,202
Bytes Transferred of Files Previously Transferred	4,523,742,089
Average Transfer Size over all Transfers [bytes]	211,601
Average Transfer Size for Files Transferred Once Each [bytes]	121,362
Average Transfer Size for Files Transferred More Than Once [bytes]	263,691

Table 2: FTP Duplicate File Transfer Measurements

Since so much of the file transfer traffic consisted of duplicate transmissions, the question that arises is how much file data should be cached. While the duplicate transmission set is 21% of all files, a small number of files account for most of the duplicate transfer activity. Figure 3 shows a histogram giving the distribution of repeat counts for files transferred more than once, with a logarithmic Y axis. (Files transferred exactly once are not included, even though the X axis starts at 0.) As an example of how caching could reduce network load, of the 4,982 files transferred more than once, only 314 (6.30%) were transferred 10 or more times each, and these files accounted for 19.30% of the file transfers. By caching a small number of files at strategic network locations, a large number of repeat transfers could be avoided.

Table 2 also shows that files transferred more than once were significantly larger on average than files transferred only once. Figure 4 shows that these percentages represent steady state averages. During the first few days of data collection, the sizes and counts of duplicate vs. other files varied by quite a bit, as can be seen in Figures 2 and 4. After that point in time, the percentage of duplicate transfers rose significantly, and the curves began to level off to steady state averages. This change was caused because the first few days of data collection fell during a relatively low activity period (a weekend; see Figure 6).

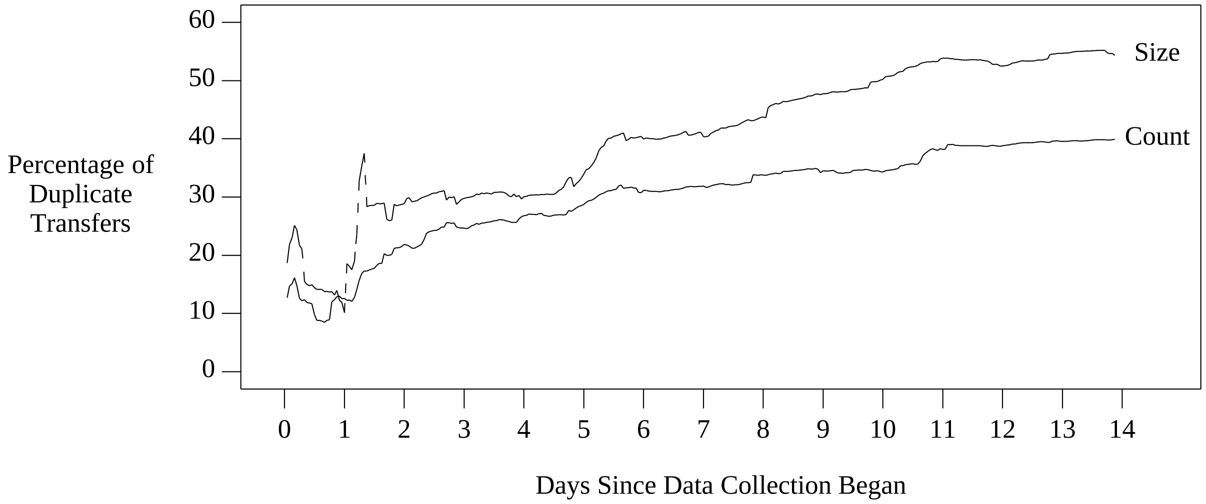

Figure 2: Duplicate Transfers over Time

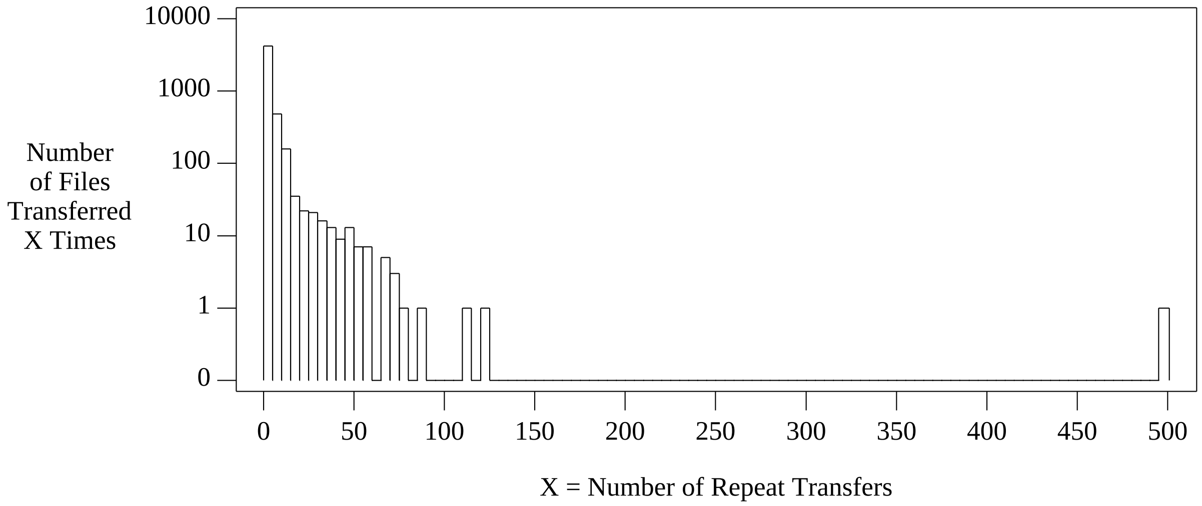

Figure 3: Distribution of Transfer Repeat Counts

Distributions of Transfer Sizes, Retrieval Times, Transfer End Points, and File Types

Figure 5 shows histograms of transfer sizes, with a log-log scale. The smallest ratio of duplicate to single transfer counts was for moderate sized (10k-320k) files, ranging from 2.8 to 6.4%. The ratio was highest (and hence caching would be most effective) for small (0-10k) and large (320k-40.96M) files, ranging from 14 to 28%. Note also that the smallest files (0-10k) accounted for the largest number of files. This observation is similar to Ousterhout et al.'s finding that most files are small [Ousterhout et al. 1985], except that the environment we considered is more heterogeneous than just Berkeley UNIX systems.

Figure 6 plots the number of file transfers as a function of time and day of the week.[5] Interestingly, the time of

[5] The X axis is labeled as DayHour, where Hour is either 6 (6 AM) or 18 (6 PM).

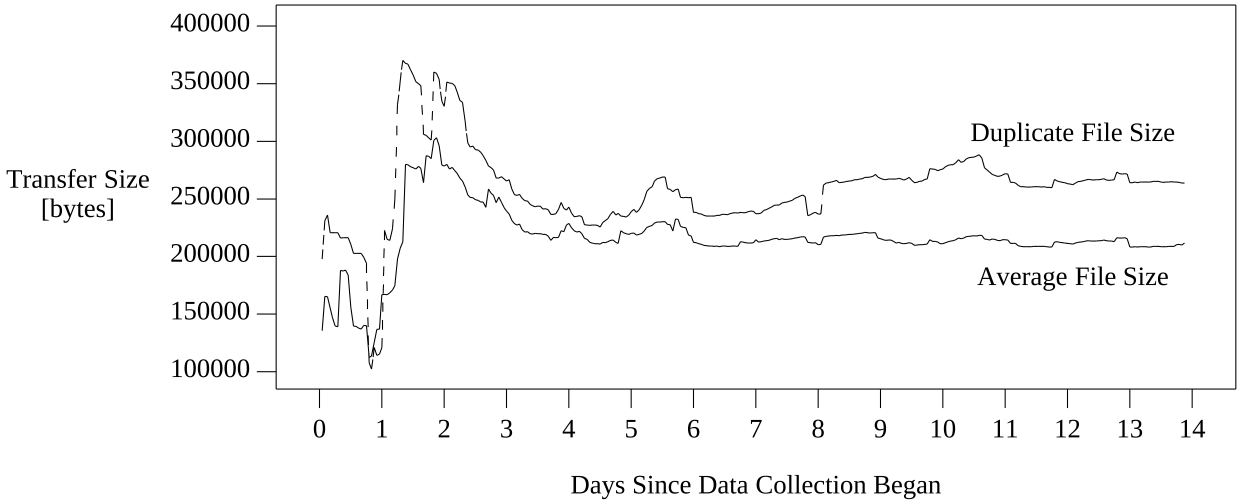

Figure 4: Transfer Sizes over Time

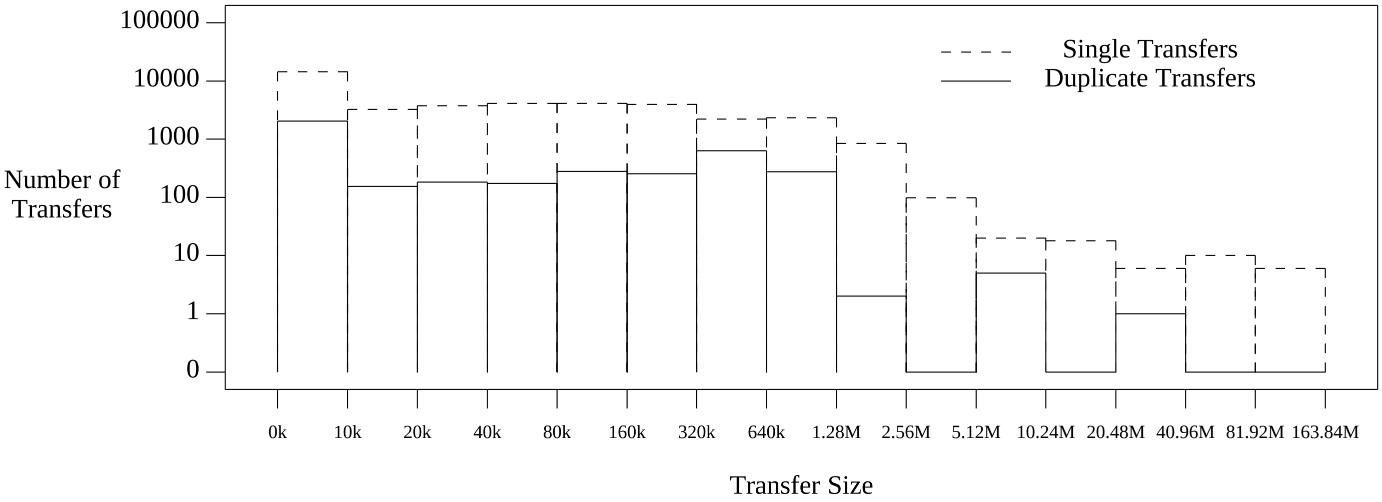

Figure 5: Distribution of Transfer Sizes

the month had more effect on the peak transfer rate than did the day of the week, with increasing activity as the Christmas holiday season approached. Also, we were surprised by how pronounced the peaks and valleys were in this plot. Given the international scope of the Internet, we expected files to be accessed more consistently around the clock. Probably the spikes would be less pronounced for the corresponding NSFNET backbone data.

Figure 7 shows histograms of the number of sources and destinations of all (not just duplicate) FTP transfers, by transfer count, with a logarithmic Y scale. These plots show that a relatively small number of sources transferred files to a relatively large number of destinations. A topology-based distribution and caching mechanism would be effective in these circumstances.

Table 3 shows the types of the most commonly transferred files, as inferred by file name. To build this table, we combined the 20 most frequently transferred types with the 20 heaviest users of network bandwidth by file size. (The table accounts for 65.18% of all file transfers and 82.80% of all the data transferred over the monitoring

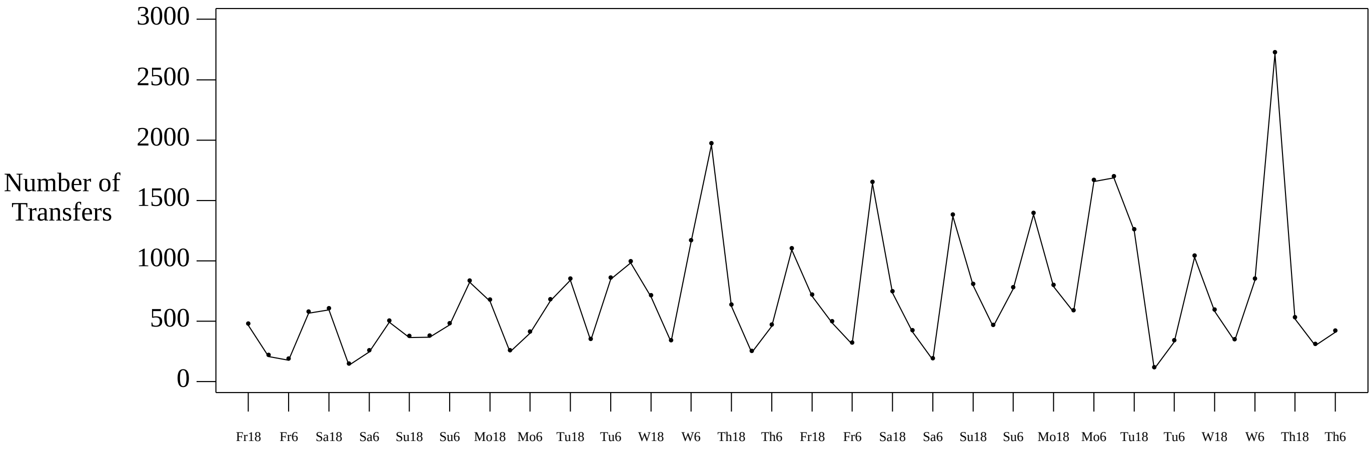

Figure 6: Files Retrieved Per Time Period

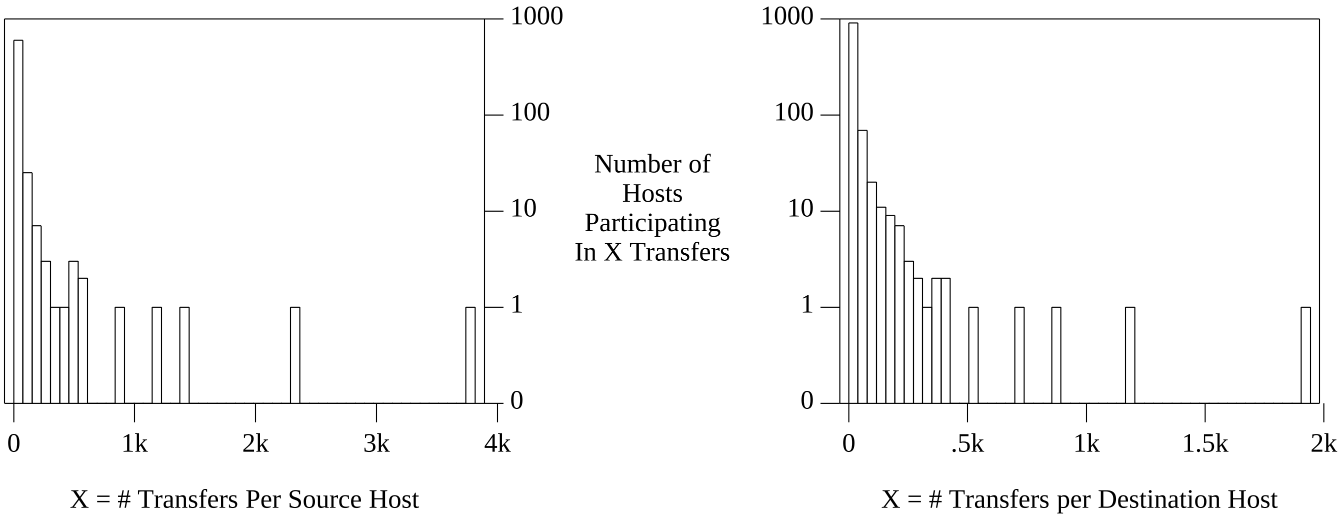

Figure 7: Source and Destination FTP Count Histograms

period.) Name comparisons were done insensitive to case. This breakdown of file types is imperfect, since naming conventions are not standardized. Nonetheless, it is informative. For example, it is interesting to note how high up in the table graphical images reside, both because image data transfers outstrip text transfers in network usage, and because of the controversy that arose a few years ago over the transmission of pornographic "gif" files across the Internet.[6] Table 3 also clearly indicates that personal computers are making significant use of Internet bandwidth for transferring executables and other types of data specific to those systems.

[6] From browsing the list of file names it is clear that such files are still being transferred. However, many innocuous "gif" files were also in the list, such as photographs from the Voyager probe.

% by File Count	% by File Size	Number of Transfers	Average File Size [bytes]	File Name	Probable Meaning
13.860	10.512	5,450	160,490	*.z, *.z?of*, *.z;*	Lempel-Ziv Compressed
9.442	28.642	3,713	641,889	*.zip	ASCII encoded, compressed archive (IBM PC)
8.721	5.049	3,430	122,496	*.gif*, *.gl, *.jpeg*, *.jpg, *.pbm, *.ras*, *.tif*, *.ppm, *.pic*, *.icon, *.plot	Graphical image
8.414	3.342	3,309	84,036	*.asc*, *.descrip*, *.doc*, *.guide, *.not*, *.text*, *.txt*	ASCII document
4.018	0.089	1,580	4,664	*.c, *.cc, *.h, *.lex, *.yac, *.yacc	C program source
3.939	16.255	1,549	873,207	*.*dat*	Binary data
3.250	2.193	1,278	142,804	*.*ps*	PostScript document
2.584	1.850	1,016	151,489	*.*hqx	ASCII encoded, compressed archive (Macintosh)
2.357	0.209	927	18,781	*.f, *.for, *.fortran	Fortran program source
2.245	6.552	883	617,405	*.arj	Compressed
1.989	0.022	782	2,396	*readme*	Description of directory contents
1.541	0.141	606	19,395	*.*tex*, *.dvi	TeX formatting source and output
1.325	0.617	521	98,539	*.me, *.ms, *.tbl, *.troff	Troff formatting source (UNIX)
1.101	0.236	433	45,289	*.wp5	Word Perfect formatting source
0.900	0.364	354	85,462	*.rtf	Microsoft's Rich Text Format
0.776	0.450	305	122,750	*.d	Binary data
0.645	0.245	254	80,220	*_uu, .encoded, *.uue*	ASCII encoded (UNIX)
0.615	0.521	242	179,073	*.lzh	Compressed (various PCs)
0.585	0.063	230	22,948	*.bin	Executable program (non UNIX)
0.557	0.667	219	253,441	*.0	First file in a sequence
0.409	1.140	161	589,351	*.dms	Unknown
0.379	0.462	149	258,031	*.list	Unknown
0.153	2.556	60	3,544,066	*.tar	UNIX file archive
0.150	0.413	59	582,700	*.a01	Unknown
0.074	0.374	29	1,073,403	*.v1j77	Unknown
0.003	1.738	1	144,601,200	*.fzx	Unknown

Table 3: Most Commonly Transferred File Types

Looking at file types also gives an indication of the proportion of files that are transferred in compressed form. If all ".z", ".gif", ".zip", and ".hqx" files are counted as being compressed, only 32.16% by count, and 45.24% by size, of the listed files were transferred compressed. If all files had been transmitted compressed and compression reduced data size by an average of 40%[7], network load caused by FTP could decrease by 27.14%, decreasing the overall load at the point of measurement by 4.00%. This amount would presumably be magnified on the NSFNET backbone, since a higher preportion of traffic on the backbone is from file transfers (23.43% as opposed to 14.00% at the point of measurement).

It is also interesting to note that some very large files were transferred. The last 4 file types listed in Table 3 accounted for 0.38% of the file transfers, but accounted for 5.08% of the data transferred during the monitoring period. In fact, there was one transfer of a 144.6 megabyte file, singularly accounting 1.738% of the data

[7] A typical figure for Lempel-Ziv compression [Welch 1984].

transferred.

Table 4 lists the types of files transferred multiple times, for which at least 10 different files were multiply transferred, along with the number of different files with these types that were transferred multiple times. These files represent the most popular files to be transferred multiple times, and hence the types of files that might benefit most from a topology-based distribution and caching mechanism.

File Name	Number of These Files Multiply Transferred	File Name	Number of These Files Multiply Transferred	File Name	Number of These Files Multiply Transferred
*.z	595	*.dat	64	*.s	16
*.gif	328	*.wp5	62	*.makefile	16
*.txt	299	*.readme	58	*.out	15
*.zip	208	*.h	54	*.dms	14
*.hqx	177	*.bin	50	*.shstat	13
*.c	161	*.0	38	*.v1j77	12
*.troff	130	*.lzh	34	*.d	12
*.f	127	*.jpg	30	*.au	12
*.asc	123	*.arj	30	*.drw	11
*.ps	119	*.for	27	*.com	11
*.tif	116	*.arc	26	*.a	11
*.doc	96	*.snd	21	*.1	11
*.tex	80	*.index	19	*.exe	10
*.rtf	72	*.cs	17		
*.eps	71	*.4	17		

Table 4: Popular Duplicate Transfer File Types

User Errors With File Data Conversion

A final interesting note concerns user errors when transferring binary data. We found that 1,892 files (transferred 4,059 times) had the same names and sizes, but different sampled contents. This surprised us. While it is possible for two files to have the same name and size but different contents, it is highly unlikely, particularly for large files. By studying the data we realized that many of these cases were probably caused by the following problem. FTP supports file data conversion, with the default being to translate data into and out of 8 bit ASCII data as a "network standard" representation. To transfer a file without this conversion, the user must precede the file transfer request with a command telling FTP to turn off data conversion. A common mistake is to transfer binary data without first turning off conversion. When this happens, the file is transferred with garbled data, and must be retransferred.

We believe this data conversion error is responsible for many of the cases of files with identical names and sizes but different sampled contents, based on two observations. First, if this error happens, a file will show up with the same name and size, but exactly two different sets of contents (corresponding to a user retrieving a file first garbled with improper conversions, and then again with the proper conversion). For 1,850 (97.78%) of the 1,892 files with identical names and sizes but differing sampled contents this was the case. Second, 1,262 (68.22%) of these 1,850 files contained binary data (based on their file names).

This observation means that 4.70% of all file transfers had to be redone because of user errors when requesting the file transfer. Put another way, 8.21% of files retrieved more than once were caused by these user errors. This observation suggests that one way to reduce wasted Internet load would be to make FTP more cognizant of file conversion needs. If the file being transferred resides on a typed file system (such as VMS), FTP could check the file type before selecting a conversion. For non-typed file systems like UNIX, file types could be checked heuristically (e.g., using the UNIX "file" command, which examines a file's contents). This check could then be used to advise users how to set file conversions. The best solution would probably be to integrate file transfer support into individual applications in a fashion that hides the details from users. A more short term aid might be to check the host and operating system types of both ends of the transfer, and if compatible (e.g., both are Suns running UNIX), transfer the file with no conversions. Given the number of data compatible systems in the Internet, this solution might solve the problem for a large number of cases.

5. Conclusions

As operational networks grow in usage and connectivity, an increasingly important problem is supporting effective means of disseminating and sharing large amounts of information among many recipients. In applications such as software distribution, document dissemination, and resource discovery, it is often the case that a small proportion of all data is required at a large proportion of sites. To support such applications, we propose a topology-based mechanism for distributing and caching information in response to usage patterns across wide area networks. Such a mechanism could potentially reduce network load substantially, since data transport activities typically account for the majority of bytes transmitted across operational networks.

To help establish the utility of such a mechanism and to better understand what types of access it should optimize for, we conducted a measurement study of file transfer traffic between the University of Colorado and the rest of the Internet. Monitoring traffic for two weeks, we found that 14.00% of the packets were caused by FTP traffic, and that 39.93% of the files were transferred multiple times, accounting for 54.37% of the file transmission traffic. This means that duplicate FTP file transfers accounted for 7.61% of the total network load. Moreover, we found that these duplicate file transfers were significantly larger on average than other file transfers. Duplicate file transfers consisted of 21.10% of the files that were transferred, with a small (6.30%) proportion of very popular files generating 19.30% of the file transfers. Because a small number of large files were transferred a large number of times, a topology-based distribution and caching mechanism could reduce Internet load, and generally contribute to more effective dissemination of information.

Since 23.43% of NSFNET backbone traffic is caused by FTP traffic [NSFNET Service Center 1991], we extrapolate our measurements to $\frac{23.43}{14.00} * 7.61 = 12.74\%$ of backbone packets being used for duplicate file transmissions. If other types of data transport are taken into account (such as resource discovery and network news applications), the packet waste would be higher still.

We also found that 8.21% of duplicate file transmissions were caused because of user errors when transferring binary data, because of the default ASCII conversion mechanism of FTP. Moreover, only 32.16% of files were transferred compressed in some form. If FTP were modified to automatically compress files, a reduction of 4.00% in total network load would result. These two observations indicate that Internet data dissemination could be done more effectively if support for compression and data conversion were integrated into applications in a fashion that hides the details from users.

A final interesting statistic is that only 44% of FTP connections resulted in file transfers. Most other connections appear to have been directory requests. Whether caused by manual requests or "Archie" like directory services, this statistic underscores the need for better resource discovery support in the Internet.

Future Work

A future version of this study might incorporate more detailed measurements about the distribution of sites transferring duplicate files. For example, given the IP network numbers for each file transfer (rather than just one pair for each file, as we currently recorded) and a map of network topology, we could map out an effective cache placement scheme. Eventually, it might be possible to use collected data to automate the process, so that the topology-based distribution and caching mechanism migrates data adaptively, according to FTP access history. More detailed information could also help to determine the number of different hosts or networks offering each file, the differential cost of multiple transfers across differing gateways (needed for mapping out cache placement), and "hot spots" that host or retrieve many copies of particular files.

We are currently exploring the possibility of running this study on the NSFNET backbone. Doing so would give us a better measurement basis for designing a topology-based distribution and caching and search mechanism.

We are also interested in conducting another study, to determine the validity of our conjecture that the popular "Archie" anonymous FTP directory service answers a large proportion of its search requests from a small proportion of its database. If this conjecture is true, distributing the heavily accessed parts of the database could allow the system to operate much more scalably than it presently does, without forcing the data to be structured in a way that limits how it can be searched. We believe such a mechanism can form the basis for scalably supporting many forms of resource discovery, in addition the usual data transport applications.

Acknowledgements

We would like to thank David Wood for allowing us to monitor network traffic at the primary gateway into/out of the University of Colorado.

This research is based on work supported in part by the National Science Foundation under grants DCR-8420944 and NCR-9105372, and a grant from Sun Microsystems' Collaborative Research Program.

6. References

[CCITT 1988]
CCITT. The Directory, Part 1: Overview of Concepts, Models and Services. ISO DIS 9594-1, CCITT, Gloucester, England, Dec. 1988. Draft Recommendation X.500.

[Caceres et al. 1991]
R. Caceres, P. B. Danzig, S. Jamin and D. J. Mitzel. Characteristics of Individual Application Conversations in TCP/IP Wide-Area Internetworks. *Proc. SIGCOMM Conf.*, pp. 101-112, Zurich, Switzerland, Sep. 1991.

[Danzig et al. 1991]
P. B. Danzig, J. Ahn, J. Noll and K. Obraczka. Distributed Indexing: A Scalable Mechanism for Distributed Information Retrieval. Jan. 1991.

[Emtage 1991]
A. Emtage. Personal Communication. McGill Univ., Montreal, Canada, Mar. 1991. Electronic bulletin board posting on comp.archives about Archie anonymous FTP site directory server.

[Heimlich 1990]
S. A. Heimlich. Traffic Characterization of the NSFNET National Backbone. *Proc. SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp. 257-258, Boulder, CO, May 1990.

[Horvath 1990]
S. M. Horvath. NSFNET Usage by Service. Message sent to nsfnet-reports@merit.edu electronic mail distribution list, Aug. 1990.

[Jain & Routhier 1986]
R. Jain and S. A. Routhier. Packet Trains - Measurements and a New Model for Computer Network Traffic. *IEEE J. Selected Areas in Commun.*, SAC 4, pp. 986-995, Sep. 1986.

[Long, Carroll & Park 1990]
D. D. E. Long, J. L. Carroll and C. J. Park. *A Study of the Reliability of Internet Sites.* Dept. of Computer &

Information Sciences, Univ. of California - Santa Cruz, Sep. 1990.

[Lottor 1990]
M. Lottor. Personal Communication. Discussion of Internet measurements to count number of domains and machines in the Internet. Apr. 1990.

[Malamud 1991]
C. Malamud. ITU Adopts New Meta-Standard: Open Access. *ConneXions - The Interoperability Report*, 5(12), pp. 19-21, Interop, Inc., Dec. 1991.

[Mills 1990] D. L. Mills. Measured Performance of the Network Time Protocol in the Internet System. *Computer Commun. Review,* 20(1), pp. 65-75, Jan. 1990.

[NSFNET Service Center 1991]
NSFNET Service Center. *NSFNET T1 Backbone Traffic Distribution by Service.* NSFNET Service Center, Nov. 1991. Available by anonymous FTP from NIS.NSF.NET in the NSFSTATS directory, in the file NSF91-11.PORTS.

[Ousterhout et al. 1985]
J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer and J. Thompson. A Trace-Driven Analysis of the UNIX 4.2 BSD File System. *Proc. 10th ACM Symp. Operating Syst. Prin.*, pp. 15-24, Dec. 1985.

[Postel 1981]
J. Postel. Internet Control Message Protocol. Req. For Com. 792, USC Information Sci. Institute, Sep. 1981.

[Postel 1982]
J. Postel. Request for Comments on Requests for Comments. Req. For Com. 825, USC Information Sci. Institute, Sep. 1982.

[Postel & Reynolds 1985]
J. Postel and J. Reynolds. File Transfer Protocol (FTP). Req. For Com. 959, USC Information Sci. Institute, Oct. 1985.

[Pu, Korz & Lehman 1991]
C. Pu, F. Korz and R. C. Lehman. A Measurement Methodology for Wide Area Internets. Tech. Rep. CUCS-044-90, March 1991.

[Schwartz 1990]
M. F. Schwartz. A Scalable, Non-Hierarchical Resource Discovery Mechanism Based on Probabilistic Protocols. Tech. Rep. CU-CS-474-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, June 1990. Submitted for publication.

[Schwartz 1991a]
M. F. Schwartz. Resource Discovery in the Global Internet. Tech. Rep. CU-CS-555-91, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Nov. 1991. Submitted for publication.

[Schwartz & Wood 1991]
M. F. Schwartz and D. C. M. Wood. A Measurement Study of Organizational Properties in the Global Electronic Mail Community. Tech. Rep. CU-CS-482-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Aug. 1990; Revised July 1991. Submitted for publication.

[Schwartz 1991b]
M. F. Schwartz. A Measurement Study of Changes in Service-Level Reachability in the Global TCP/IP Internet: Goals, Experimental Design, Implementation, and Policy Considerations. Req. For Com. 1273, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Nov. 1991.

[Sun Microsystems 1988]
Sun Microsystems. RPC: Remote Procedure Call Protocol Specification Version 2. Req. For Com. 1057, Sun Microsystems, Inc, June 1988.

[Welch 1984]
T. A. Welch. A Technique for High Performance Data Compression. *IEEE Computer Magazine*, 17(6), pp. 8-19, June 1984.