Eun-Jin Im and Allison Woodruff

An Examination of Information Retrieval Traffic on a Campus Backbone

Abstract

This paper presents an analysis of TCP and IP traffic monitored on the University of California, Berkeley campus backbone. TCP connection-level data is analyzed to show the distribution of bytes, seconds, and connections among protocols. The connection-level data also shows the request/response, topological, and hourly distribution profiles of these protocols. IP packet-level data is analyzed to show the distribution of bytes and packets among protocols. Data regarding packet length for the protocols is also discussed. The analysis focusses on two primary issues, the amount of traffic related to emerging IR protocols such as gopher and WAIS and the characteristics of this traffic. Our study shows that these protocols account for a small amount of TCP traffic: 7 percent of total connections established and 3 percent of total bytes transferred. These IR protocols are characterized by short, small connections, in contrast to protocols such as telnet which have long, large connections. Our study also shows that TCP traffic accounts for approximately half of the bytes transferred by IP traffic, and that IR protocols have a tendency to have large packets, again in contrast to protocols such as telnet.

Motivation

Much work has been done to characterize traffic patterns on both wide and local area networks, often for the purpose of examining and predicting congestion through the use of statistical models, for example in [Drak93], [FaSo92], and [FoLe91]. A somewhat smaller body of literature has examined the characteristics of network traffic according to protocol, as in [CDJM91], [DJCM92], [Paxs93a], and [Paxs93b]. These studies have focussed on protocols which currently constitute a large percentage of network traffic. In these studies, traffic from emerging information retrieval applications such as Gopher and WAIS tends to be classified as "other". However, we hypothesize that the amount of traffic from these applications will grow steadily in the future as they become increasingly popular. Additionally, the characteristics of these protocols may change as they incorporate large numbers of multi-media objects. Therefore, it is useful to establish the current traffic usage of these protocols so that their growth and evolution may be measured over time.

Further, it is interesting to study their characteristics as compared with those of other more well-established protocols such as NNTP, SMTP, or telnet. This will allow projections of impact these protocols may have on overall network traffic in the future.

Monitoring Environment and Procedure

Traffic was monitored on the University of California, Berkeley (UCB) campus FDDI backbone. The machine used for the monitoring, infocal.berkeley.edu, is a DEC 5900 running Ultrix 4.2a. Infocal is a server for several of the protocols being measured.

Packets were collected using tcpdump [JLMc89]. Tcpdump is a network tracing package which collects data on all packets which go across the subnet being monitored, including those with a source or a destination on other subnets. An intuitive method for monitoring traffic with this tool would be to collect all packets which crossed the network and organize them into complete connections. While this method would support analysis of metrics such as bytes/packet, the large number of packets crossing the campus backbone rendered such an approach impractical for several reasons. First, packet loss occurred at several different levels in the monitoring procedure. Some details on how this can occur are included in [CJRS89]. This meant that connections could not reliably be reconstructed at the packet level; the loss of a single packet in a connection would have meant that the entire connection could not be included in the final data set. Further, the packet loss was related to overflows resulting from the large number of packets crossing the backbone, particularly during periods of bursty activity. Therefore, it could be mitigated by limiting the number of packets collected. A secondary consideration was disk space; an hour trace of all packets (minus the lost packets, of course) would require 70-80 megabytes of storage.

Given these constraints, we had several options. We could have monitored a network with less traffic. Because much previous work has focussed on Ethernet LANs with less traffic, we believed that not only would large-scale results be of interest, but that problems encountered in this type of large-scale analysis would be valuable as well. Alternatively, we could have restricted the measurements to specific protocols or to specific hosts. If we had chosen to do this, we could have attempted to measure different protocols at different times, e.g. measure Gopher on Monday and WAIS on Tuesday. This would, however, have limited the amount of comparison we could do among the protocols, since their profiles vary over time.

As a result of these considerations, we chose to accept the packet loss, compromising data completeness to allow broader characterizations. We collected two data sets to measure different characteristics and to allow cross-verification of the algorithms used to process the data.

Data Set #1: Connection-Level

The first data set collected packets which could be used to characterize connections. The beginning of a TCP connection is marked by a packet with a SYN flag sent from one host to another host. Correspondingly, the end of a TCP connection is marked by a packet with a FIN flag sent from one host to another [Info81b]. In our monitoring, tcpdump utilities were used to output only those packets containing SYN or FIN flags. This output was processed in real-time by a simple Perl script which simplified the tcpdump format to minimize disk requirements. The reformatted output was processed later to match SYN and FIN packets which uniquely described a connection. Information which could be gathered from these matched SYN/FIN pairs included connection duration, source, destination, and protocol. SYN/FIN packet data was colleced from 12AM Tuesday, November 16th through 11:59PM on Thursday, November 18th. The resulting trace consists of nearly 2.4 million packets. By way of comparison, [Paxs93a] and [Paxs93b] analyzed 2.5 million connections which occurred over a six-month period.

Network Traffic Metrics and Categories

For each connection, our post-processing script records the total number of bytes transferred in the connection, as well as the duration of the connection in seconds.

Traffic was measured at a high degree of granularity, to minimize packet connection and processing time and to protect individual privacy. Each connection is assigned a value from each of the following categories:

protocol

```
protocol :: gopher | wais | http | dap | ftp_c | ftp_d |
     rlogin | shell | telnet | smtp | domain | finger |
     nntp | other
```

TCP protocols have specific port numbers assigned to them. Many of these port numbers are conventional throughout the Internet [RePo92]. Conversation between one host and another using one of these protocols generally uses an arbitrary port number (one which does not overlap with the conventional port numbers) on the client and the protocol number on the server. Therefore, we identify protocols according to the known port number used by either the source or the destination host. The protocols, their ports, and the services they provide are:

gopher (ports 70, 4324, and 7777): *Gopher information browser* [Ankl93]
WAIS (port 210): *ANSI z39.50 information browser* [Bern93b]
http (port 80): *Hyper-Text Transfer Protocol sometimes used by World Wide Web and Xmosaic browser* [Bern93b]
dap (ports 998 and 999, infocal only): *Phone book queries, only measured on the infocal server at Berkeley* [Kunz93]
ftp_c (port 21):*File transfer (Control)* [PoRe85]
ftp_d (port 20): *File transfer (Data)*
rlogin (port 513): *Remote login*
shell (ports 512 and 514): *Remote process execution*
telnet (port 23): *Remote terminal* [PoRe87]
smtp (port 25): *Electronic mail* [Post82]
domain (port 53): *Domain name service* [Mock87]
finger (port 79): *User information* [Harr77]
nntp (port 119): *Network news transfer protocol* [KaLa86]
other (all other ports): *all other TCP services*

source

```
source :: Infocal | inside UCB | outside UCB
```

The participants in a conversation can be identified from the source and destination host IP numbers output by tcpdump. The server we were monitoring, Infocal, was identified by IP number 128.32.155.4. Hosts with IP numbers of 128.32.*.* are part of the primary UCB domain, and were classified as inside UCB, as were hosts on several networks in more limited use, 192.31.95.*, 192.101.42.*, 192.31.161.*, 192.58.221.*, and 136.152.*.*. All hosts not on one of these networks were considered to be outside UCB.

destination

`responder :: Infocal | other inside UCB | outside UCB`

Destination hosts were categorized exactly as source hosts above.

time of day

`time of day :: 1 | 2 | ... | 24`

Time-of-day was determined by marking the time the packet was seen by the kernel. The connections were categorized according to the hour during which they were terminated.

direction

`direction :: request | response`

In these protocols, a conversation between two machines is comprised of two one-way connections. Therefore, after SYN/FIN packets were matched to identify connections, connections were matched with each other to form complete conversations. The connections within a conversation were categorized as either request or response. For example, suppose a user on machine A initiates a telnet session on machine B. During the conversation, all traffic for that telnet session from machine A to machine B will take place over a single connection, the request connection. All traffic from machine B to machine A will take place over another connection, the response connection.

Connection and Conversation Identification

Connections were identified by matching SYN/FIN flags with identical source and destination hosts and ports. The protocols allow only one such connection to be alive at any given time. Therefore, if no packets were lost, matching of SYN and FIN packets would be sufficient. However, because some packets were lost, there were several possible error conditions which could arise. First, if a SYN packet were lost, two FIN packets in a row could be found for exactly the same hosts and ports (example 1). In this case, the second FIN packet was assumed to be associated with a missing SYN and was discarded from our data. Second, if a FIN packet were lost, two SYN packets could occur in a row (example 2). In this case, the first SYN packet was discarded. Both examples 1 and 2 mean that connections were omitted but did not introduce erroneous data. Third, if both a FIN packet and a SYN packet were lost, an apparently legitimate connection could be formed from packets which were in fact part of different connections (example 3). This was the most serious case, since it introduced the possibility of incorrect data.

example 1: S1 F1 [S2] F2 => S1 F1

example 2: S1 [F1] S2 F2 => S2 F2

example 3: S1 [F1] [S2] F2 => S1 F2

Once the SYN/FIN packets had been matched, information about the connection was recorded. Duration was assigned to be the time consumed between the time of the SYN packet and the time of the FIN packet. The number of bytes transferred is the value of the initial sequence number subtracted from the final sequence number (this byte count does not include bytes transferred for headers). Once these values were ascertained, the connection was categorized according to protocol, source, destination, and time-of-day.

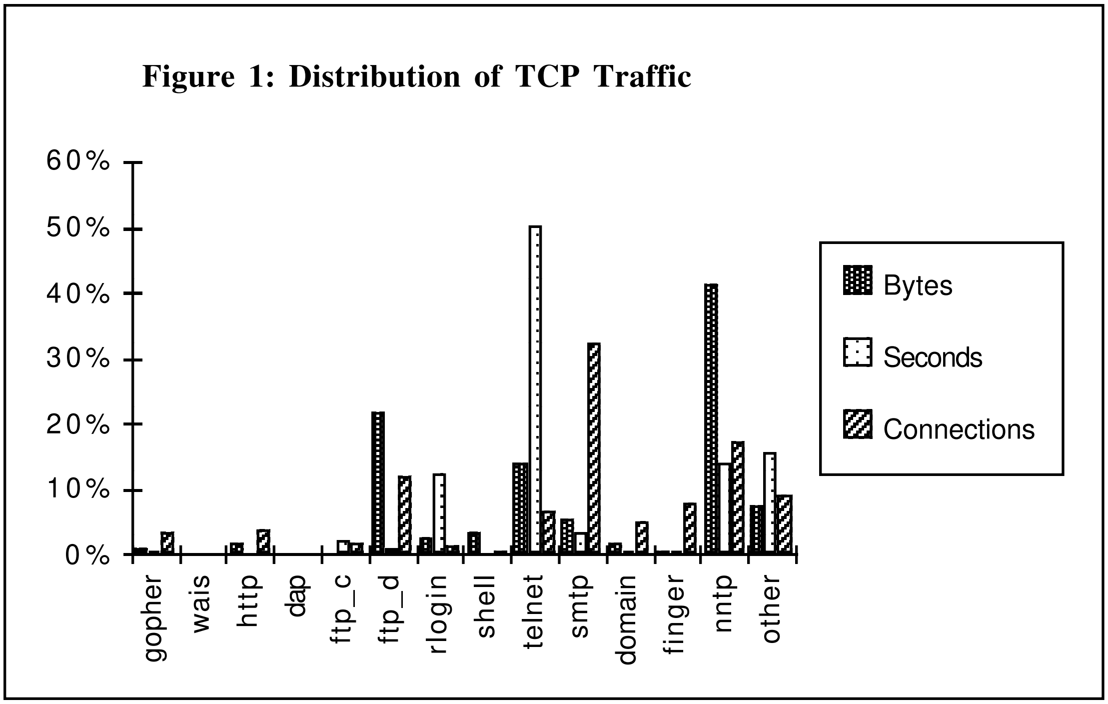
Conversations were identified by matching connections using an algorithm similar to that described above for SYN/FIN packets. The only difference is that instead of matching identical source and destination hosts and ports, the algorithm must match a source in one connection with a destination in another connection, and vice versa. In other words, if there is a connection from A->B, the matching connection will be from B->A. The possible errors were the same: connections could be omitted from further consideration, or connections might be incorrectly matched to form incorrect descriptions of conversations. Note that many connections for which we did not find a match were spurious connections, so conversation matching helped to filter errors in the data.

A negligible percentage of connections was removed because they had impossibly high transfer rates or exceptionally unlikely sizes (e.g. 2 GB). These represent erroneous data which was either introduced by our matching algorithm or was a result of packet modification due to overflows during data collection. Of the approximately 2.4 million SYN/FIN packets recorded, approximately 54% were included in the final data set. Many of the packets would have been removed in any case, e.g. because they were resent packets, or due to variations in topological routing over the Ethernet versus the FDDI backbone. However, a significant portion was removed because their corresponding SYN/FIN packet was dropped or because their sequence numbers were incorrectly overwritten either at the kernel level or in tcpdump. As a result, no projections of total bandwidth on the backbone are made in this study.

Results

In this section we present the analysis of connection-level data. Figure 1 shows the distribution of bytes, seconds, and connections among the protocols measured. The traffic for each protocol is presented as a percentage of the total traffic for each metric. The leftmost bars in a cluster represent the percentage of the total bytes measured which were transferred by each protocol; the values of these bars totals 100%. Similarly, the middle bars represent the percentage of the total seconds consumed by a specific protocol. The final bars represent the percentage of the total connections made by that protocol. For example, Gopher accounts for approximately 1% of the bytes transferred, slightly less than 1% of the real-time seconds spent in connections, and approximately 4% of the total connections made. The highest consumer of bytes was NNTP, which transferred approximately 5 gigabytes over the three day period measured. Telnet consumed the most seconds, 66 million. It is expected that telnet traffic is particularly large because of traffic from modems. Finally, SMTP made 271 thousand connections, the highest number observed.

Each clustered measurement of the percentage of bytes, seconds, and connections used by a protocol can be considered to be the profile of that protocol. The profiles of the protocols differ greatly in the relative distribution of these three metrics. For example, Telnet consumes a large number of seconds/connection compared to Gopher and HTTP (the World Wide Web protocol).



We also examined the distribution of the traffic according to topology. In Figures 2a and 2b, the first bar shows the percentage of each protocol's traffic which involved internal hosts only. The second bar shows the percentage which involved both an internal and an external host. These two bars sum to 100% for each protocol, representing the total traffic for that protocol. (Traffic with both an external source and destination did cross the backbone occasionally, but since it constituted approximately 0% of the total traffic, it was not considered in this analysis.) The distributions of byte and connection traffic are similar across protocols. The information retrieval services, as well as many other information gathering services such as FTP, have a high percentage of traffic to outside hosts. Services such as rlogin, telnet, and domain, however, have a strong tendency to use only internal hosts. One reason for this distinction may be that services such as rlogin require accounts on remote machines, while IR-related services and FTP require only guest privileges.

Figure 2a: TCP Topological Byte Distribution

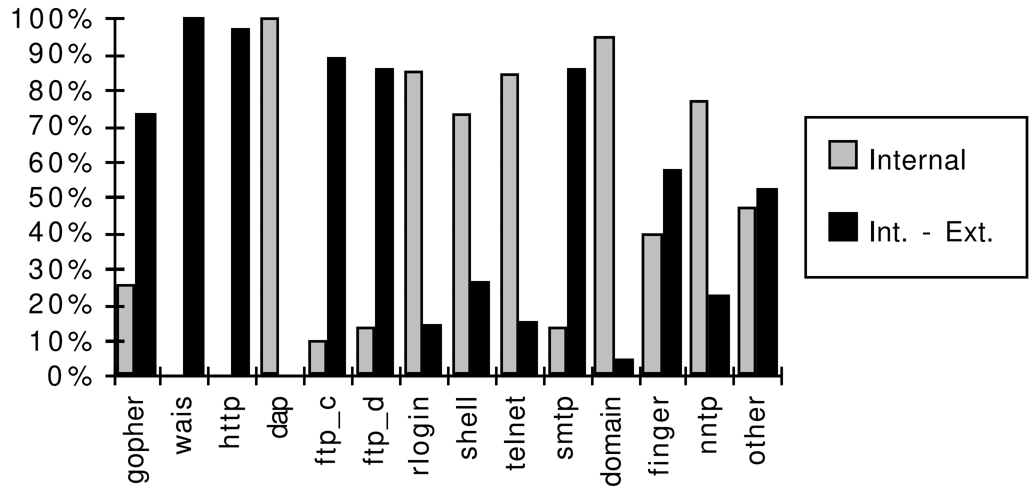


Figure 2b: TCP Topological Connection Distribution

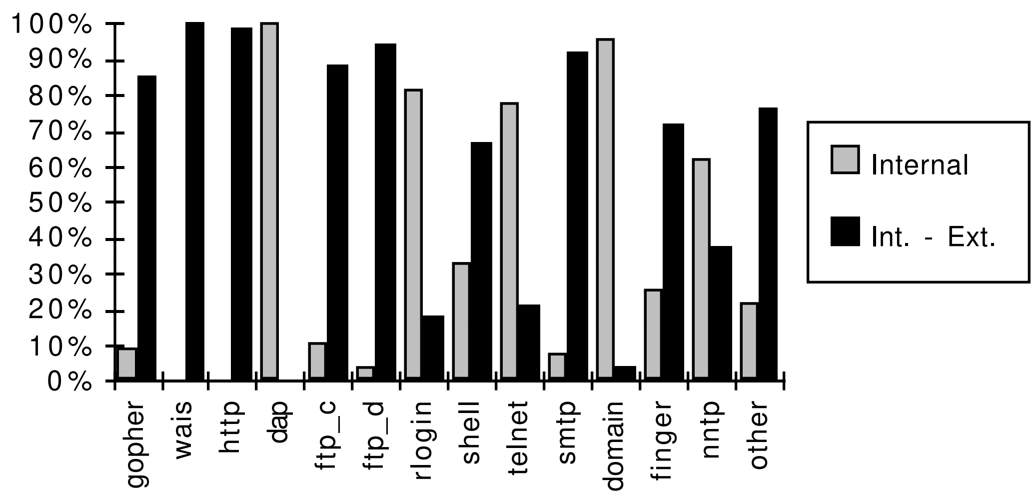


Figure 3a: TCP Request-Response Bytes

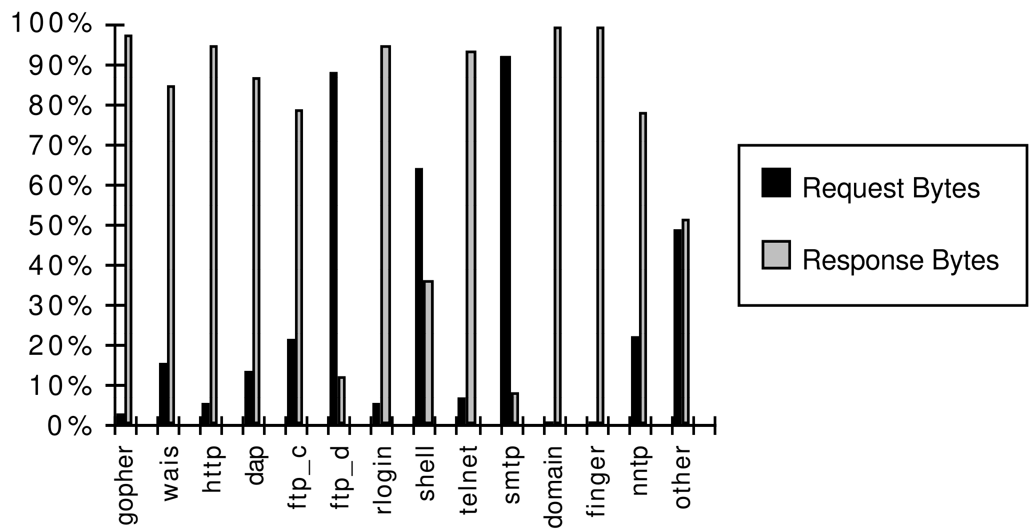
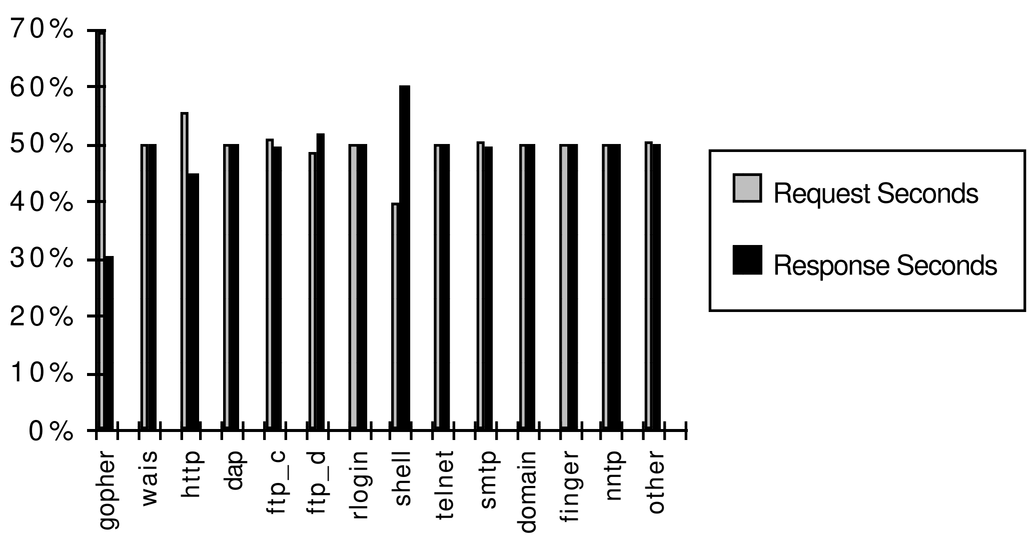


Figure 3b: TCP Request-Response Times

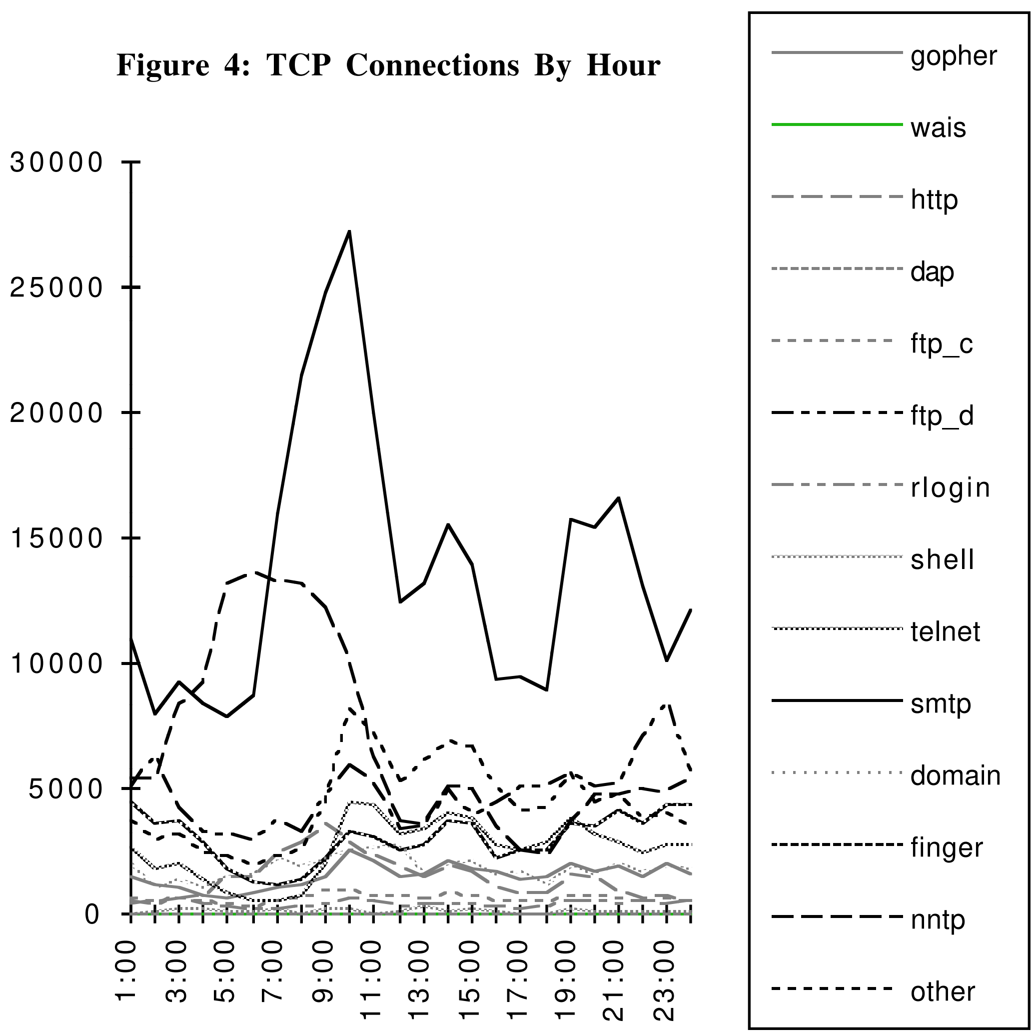


We were interested in the request-response nature of TCP protocols. Figure 3a presents the results of our analysis of the percentage of bytes transferred each direction in a conversation. The first bar for each protocol shows the percentage of the total bytes which were transferred over request connections. The second bar for each protocol shows the percentage of the total bytes which were transferred over response connections. These two bars sum to 100% for each protocol, representing the total bytes for that protocol. Most protocols have a relatively low percentage of bytes transferred over request connections and a relatively high percentage of bytes transferred over response connections. One notable exception is FTP data, which initiates the conversation at the FTP site, at instructions from FTP control. Other exceptions include SMTP and shell. The surprisingly large number of total bytes transferred by remote shell has been noted by [Paxs93a]. One possible explanation for the size and directional distribution of the traffic could be the use of remote shell to transfer large files for processing on remote hosts.

The time distribution analysis of request-response data was also useful in evaluating the effectiveness of our algorithm in eliminating spurious connections. If both connections and conversations were matched properly, both the request and response connections of a conversation should have approximately the same duration in most of the protocols. On the other hand, if packets were matched to form non-existent connections, the odds of a corresponding response connection between the same hosts were very low. Further, the odds of the response connection consuming exactly the same time as the request connection were even lower. Therefore we analyzed the time distribution for request and response for each protocol. The results of this analysis are presented in Figure 3b. The first bar for each protocol shows the percentage of the total seconds which were consumed by request connections. The second bar for each protocol shows the percentage of the total seconds which were consumed by response connections. These two bars sum to 100% for each protocol, representing the total seconds consumed by that protocol. It can be seen that most protocols have nearly identical time distributions, indicating that all or nearly all connections remaining in the data set were valid. Gopher, WAIS, and shell are notable exceptions. Analysis of the data reveals that this difference is consistent across connections, and does not indicate invalid data. The behavior of Gopher is explained by the fact that the request side always opens the connection and waits until the response side closes the connection before closing [Bern93b], and we expect that the explanation of the variation for WAIS is similar. We are not sure of the reason for the variation in shell connections. Note also that the difference in time distribution is correlated to topological distribution; many protocols which transfer almost all traffic between internal hosts, such as rlogin and telnet, have virtually identical request/response time distribution.

Figure 4 presents the distribution of TCP traffic over time. Each line on the graph shows the actual number of connections terminated with a FIN packet by each protocol during each hour on the X axis. The behavior of the protocols seems to be strongly correlated with work patterns. For example, SMTP has a large spike peaking at 10 in the morning. There is a small decline in almost all protocols' activity at lunch and dinner times. NNTP is interesting in that it has a large amount of traffic very early in the morning; much of this is apparently due to transferring between news servers. IR-related protocols have no noteworthy characteristics, perhaps indicating that they have not yet been incorporated in daily routines. However, the relatively small number of connections for these protocols made it difficult to profile their behavior over time.

Figure 4: TCP Connections By Hour



From the examination of TCP traffic, we have learned that IR traffic is characterized by short, small connections with high percentages of traffic to outside hosts and traffic in response connections.

Data Set #2: Packet-Level

The second data set collected packets which could be used to characterize packet size. tcpdump outputs all types of packets which cross the network, including packets from other IP protocols such as UDP. A complete packet trace was taken for three hours, from 9AM-12PM on Friday, November 19th. The resulting trace consists of nearly 2 million packets.

Network Traffic Metrics and Categories

For each connection, our post-processing script records the total number of bytes, header included, transferred in each packet. Each packet is assigned to the following categories:

protocol

```
protocol :: TCP | UCP
TCP :: gopher | wais | http | dap | ftp_c | ftp_d | rlogin |
       shell | telnet | smtp | domain | finger | nntp | other
```

In addition to the TCP protocols identified above, packets from other IP protocols such as UDP and ICMP were measured [Info81a]. Since the number of packets from all but TCP and UDP packets was negligible, these were dismissed from further analysis.

time of day

```
time of day :: 1 | 2 | ... | 24
```

As in the connection-level analysis, time-of-day was determined by the time the packet was seen by the kernel.

Results

Figure 5 presents the distribution of IP traffic among the TCP and UDP protocols. The first bars in each cluster represent the percentage of total bytes transferred by each protocol. The second bars in each cluster represent the percentage of total packets transferred by each protocol. Each of these sets of bars sums to 100%. The graph shows that TCP transferred 54% of the total bytes, in 75% of the total packets. UDP transferred

46% of the total bytes, in 25% of the total packets. This implies that TCP packets are smaller on average than UDP packets.

Figure 5: Distribution of IP Traffic

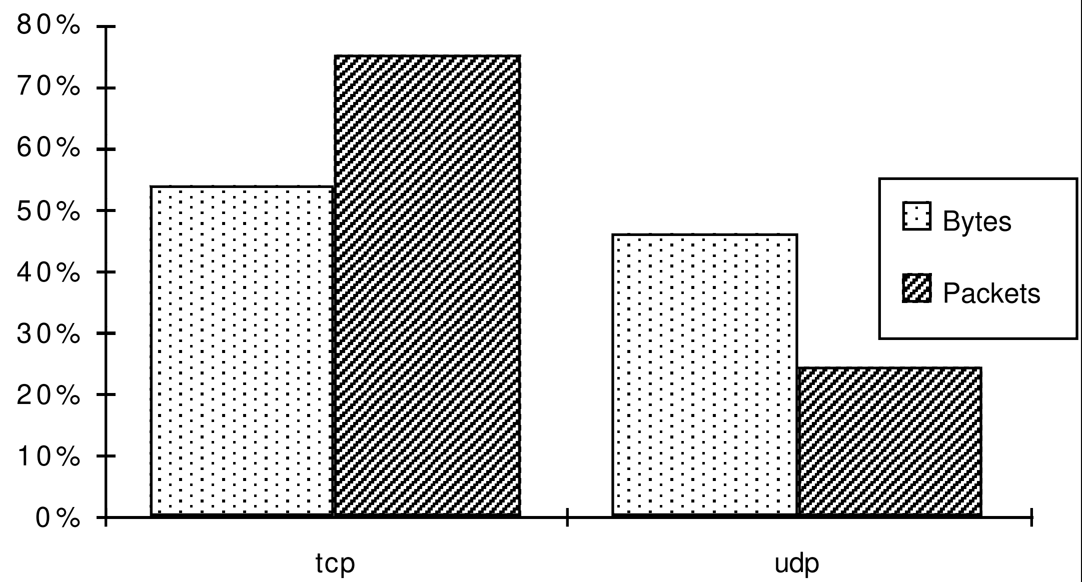
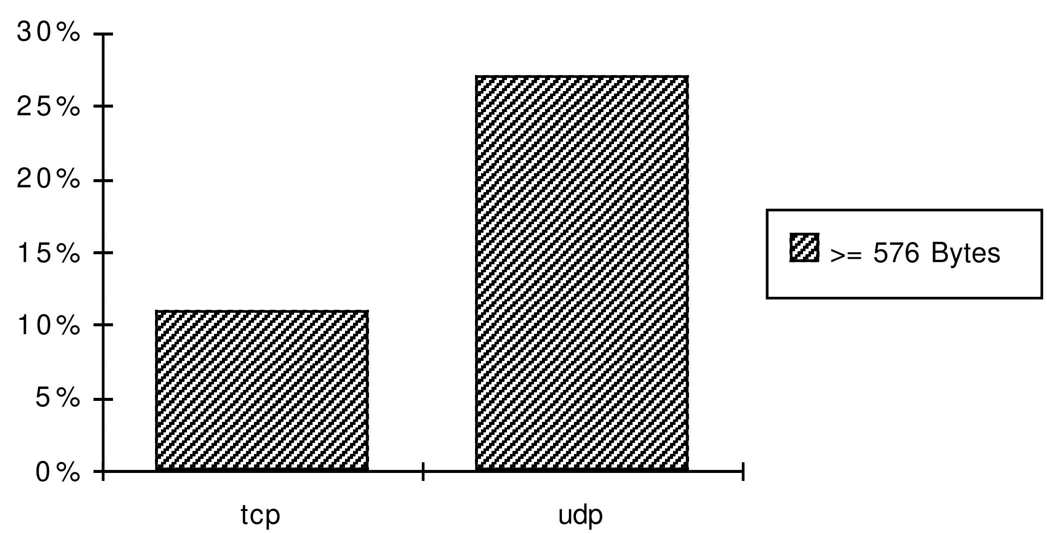


Figure 6: IP Packet Length Distributions



This property is more clearly illustrated in Figure 6. This figure shows the percentage of packets greater than or equal to 576 bytes. This value was chosen because it is a standard fragmentation boundary in the IP protocol [Hedr87]. While 27% of the UDP packets were greater than or equal to 576 bytes, only 12% of the TCP packets met the same criteria.

The distribution of TCP packet size by protocol is detailed in Figure 7. This graph shows that many of the TCP protocols transfer a large percentage of traffic greater than 576 bytes. In particular, more than half of the packets from World Wide Web (HTTP) and FTP data are greater than or equal to 576 bytes. However, traffic from a small number of protocols such as telnet, which transfers less than 576 bytes in nearly all of its packets, bring down the overall average packet size for TCP traffic. These findings are consistent with the observations of [CDJM91], who noted that bulk transfer protocols such as FTP tend to use large packets, while interactive protocols such as telnet tend to use small packets.

Figure 7: TCP Packet Length Distributions

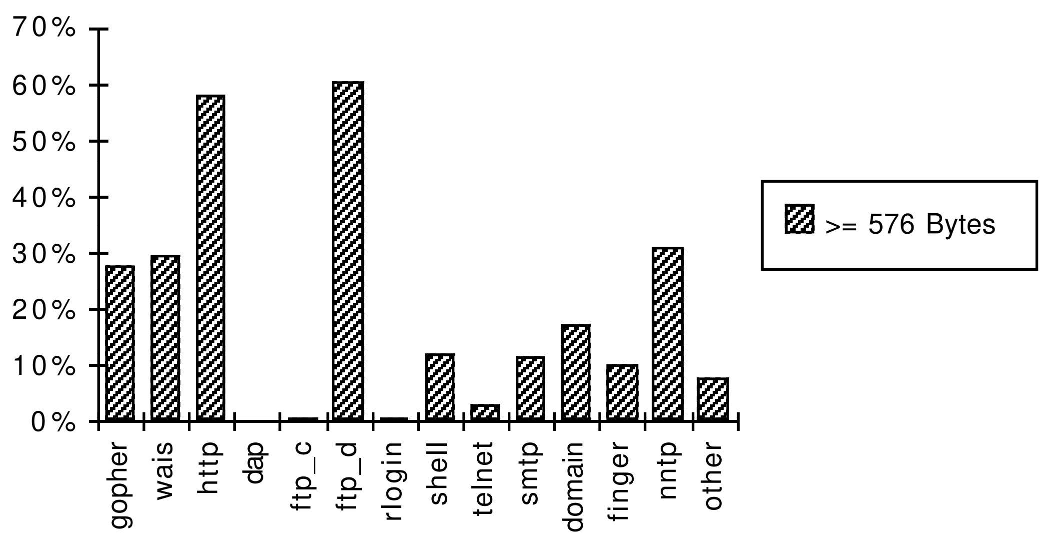
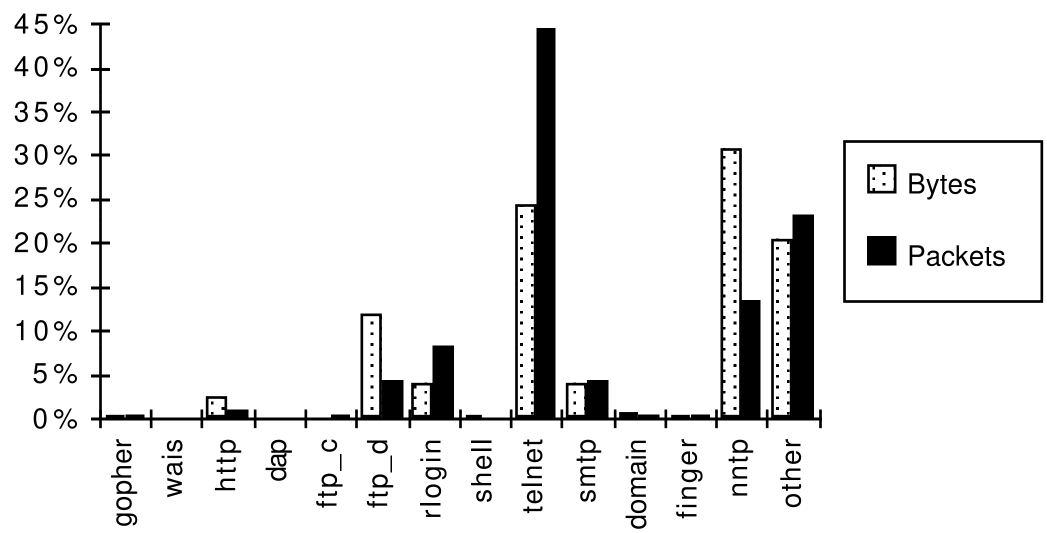


Figure 8 shows more specifically the distribution of bytes and packets measured for TCP protocols with the packet-level data. The traffic for each protocol is presented as a percentage of the total traffic for each metric. The first bars in a cluster represent the percentage of the total bytes measured which were transferred by each protocol; the values of these bars total 100%. Similarly, the second bars represent the percentage of the total packets transferred by that protocol. For example, telnet accounts for approximately 25% of the bytes transferred, but approximately 44% of the packets transferred. A key point about this graph is that the profiles of the protocols in terms of bytes is very similar to that shown in Figure 1. Since the packet data was randomly selected, and was not subject to the same errors as the connection-level data, this helps to verify the correctness of our matching algorithm for the first data set. Slight variations do occur, but this is probably attributable to time-of-day; the connection-level data took place during all hours of the day, while the packet-level data was only collected during morning work hours. Since protocols such as NNTP transfer a significant percentage of their bytes at night, this biases the comparison.

Figure 8: Distribution of TCP Traffic



Overall, the packet level data shows that IR traffic is less than 5% of the total bytes transferred, and an even smaller number of the total packets transferred. IR traffic is characterized by relatively large packets.

Summary

Our study shows that emerging protocols for interactive search and retrieval of information between distributed servers and clients currently account for a small percentage of traffic on the network. In the connection-level data, Gopher and World Wide Web together account for less than 3% of the bytes transferred and approximately 7% of the total connections made. In the packet-level data, they account for less than 5% of the bytes transferred and less than 3% of the packets. IR traffic is characterized by short, small connections and large packets, with most traffic taking place in the response side of a conversation and involving external hosts. These protocols differ from some more long-standing protocols such as telnet, which is characterized by long, large connections and small packets, with most traffic taking place in the response side of a conversation but rarely involving external hosts.

This study is significant in that it has established a baseline for examining the growth of these protocols as they become more popular. It also makes a contribution by profiling an environment with a large amount of traffic. Our results have several implications. First, as distributed retrieval protocols become more popular, and involve more multi-media applications, bulk transfers moving large packets between external and internal hosts will become more popular. These types of transfers will need to be optimized as they become more prevalent. Second, our results illustrate that the protocol definition uniquely determines the profile of the protocol in terms of its distribution of bytes, seconds, connections, and packets. Protocols such as NNTP, which consume an exceptionally large percentage of the total bandwidth, could potentially be redesigned to function in a more distributed manner, delivering specific articles to local servers only when requested. Alternately, because the current IR protocols use short, small connections, connection overhead is a relatively dominant factor; if these protocols became sufficiently popular, mechanisms could be developed to support light-weight connections.

Acknowledgements

We would like to thank Cliff Frost, John Kunze, John Forrest, George Marinos, and Margaret Baker for providing the network access and equipment which made this monitoring possible. We would also like to thank Vern Paxson, Steve McCanne, and Jeff Mogul for their many helpful suggestions during monitoring and analysis. Finally, we would like to thank John Ousterhout and Kim Keeton for their thoughtful comments on the design, implementation, and presentation of this study.

References

[Ankl93] F. Anklesaria et al. The Internet Gopher Protocol. RFC 1436, University of Minnesota, March 1993.

[Bern93a] T. Berners-Lee. Hypertext Transfer Protocol. CERN, November 1993.

[Bern93b] T. Berners-Lee. What W3 Needs from WAIS and x.500. CERN, February1992.

[CDJM91] R. Caceres, P.B. Danzig, S. Jamin, and D. Mitzel. Characteristics of Wide-Area TCP/IP Conversations. *Proceedings of ACM SIGCOMM*, 1991.

[CJRS89] D.D. Clark, V. Jacobson, R. Romkey, and H. Salwen. An Analysis of TCP Processing Overhead. *IEEE Communications Magazine*, 27(6):23-29, June 1989.

[DJCM92] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Estrin. An Empirical Workload Model for Driving Wide-area TCP/IP Network Simulations. *Internetworking: Research and Experience*, 3(1):1-26, 1992.

[Drak93] E. Drakopoulos. Performance and Traffic Analysis of a Network-based Distributed System. *Computer Communications*, 16(3):155-167, March 1993.

[FaSo92] S.O. Falaki and S-A Sorensen. Traffic Measurements on a Local Area Computer Network. *Computer Communications,* 15(3):192-197, April 1992.

[FoLe91] J.H. Fowler and W.E. Leleand. Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management. *IEEE Journal on Selected Areas in Communications*, 9(7):1139-1149, September 1991.

[Harr77] K. Harrenstien. Name/Finger. RFC 742, SRI-KL, December 1977.

[Hedr87] C.L. Hedrick. *Introduction to the Internet Protocols*, Rutgers University, Computer Science Facilities Group, New Brunswick, NJ, 1987.

[Info81a] Information Sciences Institute. Internet Protocol: DARPA Internet Program Protocol Specification. RFC 791, 1981.

[Info81b] Information Sciences Institute. Transmission Control Protocol: DARPA Internet Program Protocol Specification. RFC 793, 1981.

[JLMc89] V. Jacobson, C. Leres, and S. McCanne. tcpdump. Available via anonymous ftp to `ftp.ee.lbl.gov`, June 1989.

[KaLa86] B. Kantor and P. Lapsley. Network News Transfer Protocol. RFC 977, UC San Diego and UC Berkeley, February 1986.

[Kunz93] J. Kunze. *Personal communication*, 1993.

[Mock87] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, Information Sciences Institute, November 1987.

[Paxs93a] V. Paxson. Growth Trends in Wide-Area TCP Connections. *In submission to IEEE Network.*

[Paxs93b] V. Paxson. Empirically-Derived Analytic Models of Wide-Area TCP Connections. *In submission to IEEE/ACM Transactions on Networking.*

[Post82] J. Postel. Simple Mail Transfer Protocol. RFC 821, Information Sciences Institute, August 1982.

[PoRe83] J. Postel and J. Reynolds. Telnet Protocol Specification. RFC 854, Information Sciences Institute, May 1983.

[PoRe85] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959, Information Sciences Institute, October 1985.

[RePo92] J. Reynolds and J. Postel. Assigned Numbers. Information Sciences Institute, RFC 1340, July 1992.