A Analysis of Berkeley Measurements

With the development and increasing use of distributed systems, computer communication traffic over local-area networks has changed. Conventional remote terminal access and file transfer are no longer the dominant applications. More and more computing systems are built around diskless workstations and file servers. These workstations are economical, expandable, easy to manage, and more suitable for offices, where the noise and heat produced by a large disk subsystem would not be tolerable. They often provide a virtual-memory environment and need to page to a remote paging device. Whereas older protocols only had to respond to the slow, interactive terminal traffic and provide throughput for the bulk transfer of data, newer ones have to provide much faster response time for file access and quick network access to the backing store.

As a result of modifications aimed at increasing the performance of computer systems, now a single packet often only contains a portion of the object being transported. For instance, at the time the Ethernet was designed, file pages were 512 bytes or 1 Kbyte in size [89,91]; now, 4 Kbytes appears to be the standard. A Sun-3's memory page consists of 8 Kbytes, which are transferred to the paging device in a single transaction during paging or swapping. Additionally, new programming interfaces have increased users' need for bandwidth. Dumping the image of a graphic display may require the transfer of millions of bits. In the case of diskless workstations, these transactions are performed through network protocols, which, at the data-link layer, involve the transmission of a number of *frames* in quick sequence.

The Ethernet, designed at the Xerox Palo Alto Research Center (PARC) at the beginning of the seventies [21] and subsequently modified for higher speed [23], is one of the most successful local-area networks. It combines simplicity and low cost with reasonably large bandwidth. The characteristics of its traffic have been repeatedly studied, starting with basic measurements performed about ten years ago at Xerox PARC by J.F. Shoch and J.A. Hupp [80]. Shoch and Hupp suggested, and it is now commonly accepted [41], that the Ethernet under ``normal'' workloads is lightly loaded and that hosts and host interfaces, rather than the network, are the protocols' performance bottlenecks.

For the purpose of evaluating how network and protocol behavior has been modified by the developments in distributed systems and user interfaces, we took traffic measurements of a 10-Mb/s Ethernet at the University of California at Berkeley (UCB) aimed at determining the performance of an Ethernet local-area network in the presence of file-system access performed by diskless workstations. This paper reports the primary results of the measured data analysis, an essential part of which is the interpretation of network communication protocol behavior.

The next section presents the general statistics of the measured traffic and compares our traffic with the traffic that was described in 1980 by Shoch and Hupp. Sections III-V examine in detail the three protocols that transported the largest amount of data: the Transmission Control Protocol (TCP) [70], which is part of the Defense Advanced Research Projects Agency (DARPA) family of protocols [69, 75]; the Sun Network Disk (ND) protocol [85]; and the Sun Network File System (NFS) protocol [63]. Finally, in the last section we present the conclusions of our study.

II. General Statistics

In this section we shall briefly describe the UCB environment measured and discuss the measurement methodology. We shall analyze the network utilization, the distribution of packet lengths, the network protocols, and the patterns of packet interarrival times.

We took great care to ensure that the measurement system would generate as accurate a picture of the traffic as possible. Rather than using a special-purpose device for the measurements, we instrumented the kernel of one of our fastest general-purpose UNIX systems to read all the packets in the network, timestamp them, and save the packets' protocol information on tape for off-line analysis. The high measurement accuracy was achieved primarily by allocating a large buffer area for the Ethernet board's DMA and by employing a double-buffer technique to store the packet headers before moving them to disk and subsequently to tapes; in this way, we were able to reduce packet loss to a minimum.

The measurement system lost less than one percent of all packets transmitted. Most previous measurement projects [7, 15] suffered from much higher packet loss, in the range of 10 to 15 percent or more. Furthermore, the timing information in our traces is very accurate, enabling us to study the interarrival times closely. Though we used a clock with a resolution of 1 microsecond, the timestamps were subject to variability in the DMA time, which was caused by concurrent activity on the machine's bus; to queuing time in the Ethernet interface, which depended on the packet arrival rate; and to interrupt response time and queuing time in the host. We estimate that, because of these variable factors, our packet timings are accurate to within 1/10 ms.

The Ethernet we chose to study, a single-cable Ethernet in the Computer Science department that UCB system administrators have denominated *XCS* network, connected, at the time of the experiment, about a hundred machines: 41 diskless Sun workstations (both Sun-2s and Sun-3s) arranged in clusters around six file servers running Sun UNIX, which is based on Berkeley UNIX 4.2BSD; 23 XEROX Star workstations; three VAXs, several MicroVAXs; and a few Symbolics and TI Explorer LISP machines. Two of the VAXs and a Sun workstation were used as gateways to other UCB networks. By and large the XCS user community is similar to those of other universities and large research organizations. User activities include program development, text editing, experimentation with computation-intensive programs, and operating-system research.

For this study we collected traces of packet headers during a period of three weeks for a total of 6,500 Mbytes. Because of the sheer amount of processing and storage space involved in analyzing in detail the entire data set, in this paper we shall restrict our attention to a typical weekday for XCS traffic.

The 24-hour trace under study consists of 11,837,073 records–one for each packet. The total number of bytes transmitted is 7,060,163,648 (this figure includes header, checksum, and data bytes). Although the mean network utilization–the proportion of time in which one transmitter was active–is a mere 6.5 percent, during the period of high traffic in the late afternoon, it is very common to have relatively long intervals of mean utilization ranging around and above 30 percent. By contrast, the 2.94-Mb/s Ethernet measured by Shoch and Hupp [80] a network with about 120 machines, carried less than four percent of the total number of bytes we observed. Therefore, the first, most significant difference between our data and previous measurements, none of which notably deviated from Shoch and Hupp's data, is the higher utilization of the communication channel.

A. Network Utilization

Figure A.1 displays the network utilization in the 24-hour period under study. Each vertical line represents the fraction of time during a 1-minute interval in which transmitters were active. Thus, the packet headers, the data bytes, and the packets' checksums, as well as the 8-byte preamble, which is transmitted prior to each packet to synchronize the receiver and the transmitter, are all counted in calculating the utilization.

FIGURE A.1. ETHERNET UTILIZATION (ALL PACKETS)

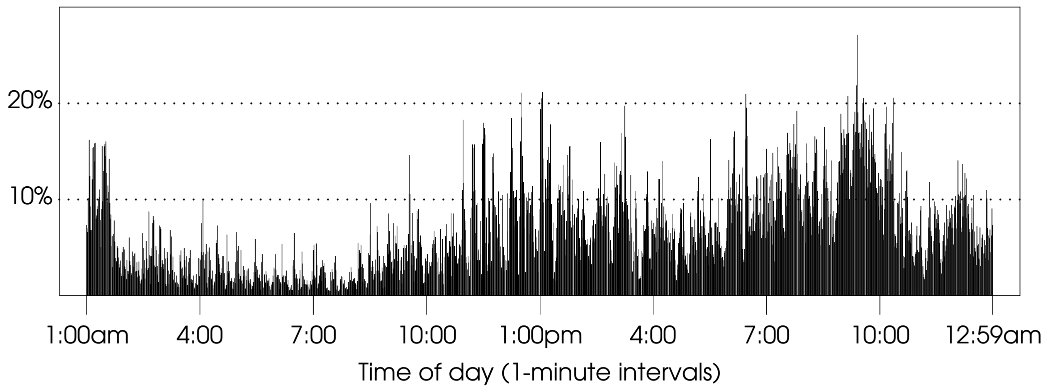
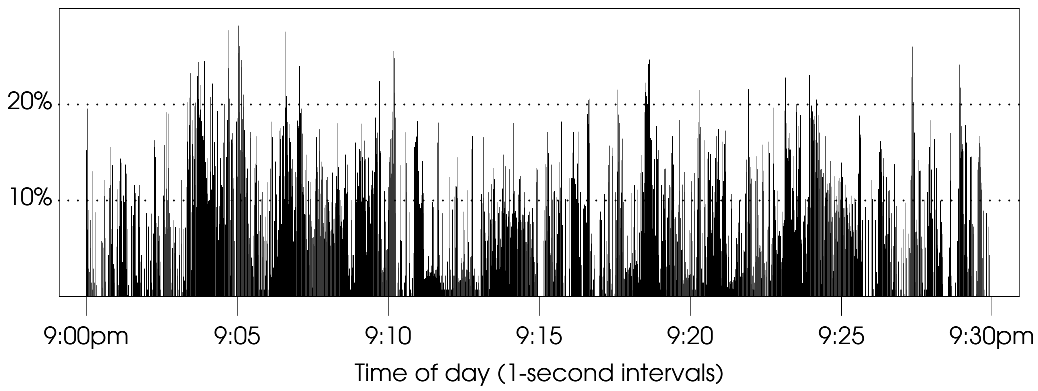


Figure A.1 shows that the utilization, measured over 1-minute intervals, rarely exceeded 20 percent. If, however, we examine the utilization more closely over smaller intervals, we see that in fact the Ethernet behavior was quite bursty, frequently reaching levels over 30 percent. Such traffic would overload Shoch and Hupp's 2.94-Mb/s Ethernet, and would also saturate the IBM 4-Mb/s ring network. Furthermore, we found that hardware changes–faster computers and more intelligent network interfaces–mean that two computers alone may generate high utilization rates. For instance, a Sun-3/180 file server and a Sun-3/50 client workstation, loaded the Ethernet more than 20 percent and reached peak transfer rates as high as 275 Kbytes per second, as shown in Figure A.2.

This behavior has been observed in experimental environments where two machines sent artificially generated data to each other [15,44]. However, its presence in

FIGURE A.2. ETHERNET UTILIZATION (TRAFFIC BETWEEN TWO WORKSTATIONS)

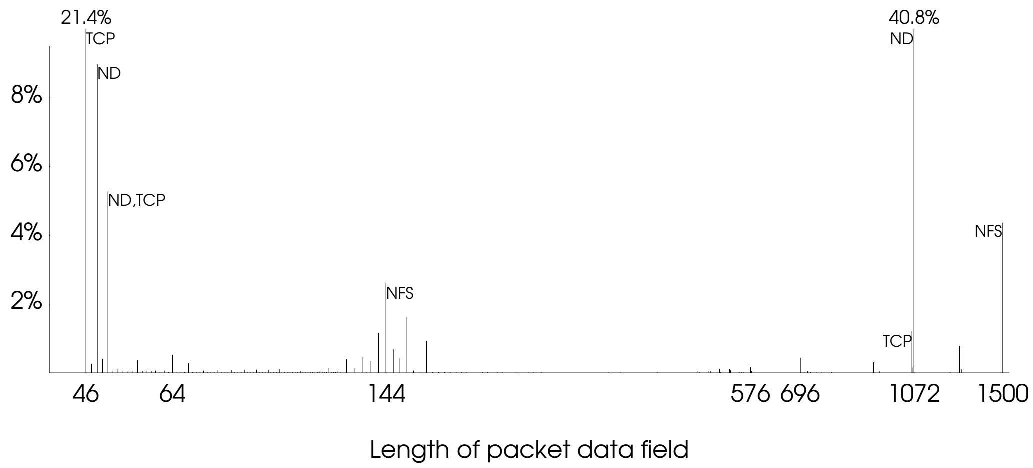


a real-world environment compels us to re-evaluate the notion that under normal traffic an Ethernet is lightly loaded [41, 80].

B. Packet Length

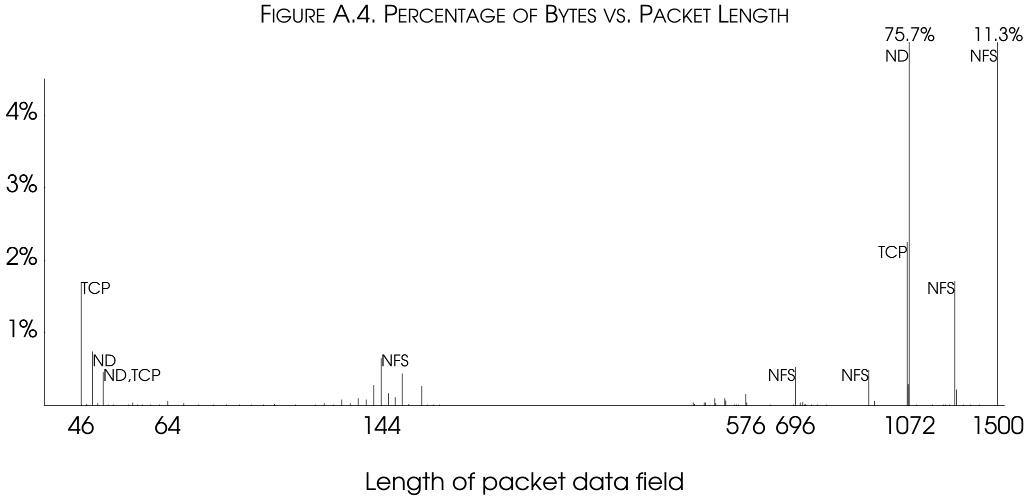
Figure A.3 and Figure A.4 display respectively the number of packets and number of bytes transmitted as a function of the packet data field length.

FIGURE A.3. PERCENTAGE OF PACKETS VS. PACKET LENGTH



In Shoch and Hupp's Ethernet the packet size distribution was bimodal and a small number of large packets accounted for most of the traffic volume. The mean and median packet lengths were 122 and 32 bytes respectively. In our network, based on a modern environment of diskless workstations with virtual-memory operating systems, the mean packet size is 578 bytes and the median is 919 bytes. However, since the maximum packet sizes of the two Ethernets are different–554 for the 2.94-Mb/s experimental Ethernet and 1500 for the 10-Mb/s Ethernet–a more revealing comparison of the these values can be obtained by dividing each mean and median by the corresponding

FIGURE A.4. PERCENTAGE OF BYTES VS. PACKET LENGTH



maximum packet length. In the case of the Xerox PARC Ethernet the mean is 22.0 percent, and the median 5.8 percent, of the maximum packet size; in our Ethernet, the mean is 38.5 percent and the median 61.3 percent.

The smallest packet sizes, those less than 50 bytes, together transport 30.4 percent of the total packets but only 2.9 percent of the total bytes. A notable portion, 30.8 percent, of these packets is used to carry requests for ND protocol transfers. Yet 66.8 percent of them are used by the character and acknowledgment traffic of TCP.

Packets between 124 and 168 bytes in length are used almost exclusively by the NFS protocol, which is based on *request-response* remote procedure calls. They account for 8.3 percent of the total packet count, but for only 2.2 percent of the total byte count. We believe that this segment of packet lengths will acquire more and more weight in the future as applications based on request-response protocols develop [15, 76].

Finally, packets larger than 576 bytes comprise 50.2 percent of the total packet count and 93.1 percent of the byte traffic. TCP transports 1.4 percent of the packets and 2.6 percent of the bytes. ND and NFS transport 40.8 and 6.2 percent of the packets and 75.7 and 14.6 percent of the bytes respectively.

C. Network Protocols

Table A.1 lists the number of packets and bytes transferred and the mean packet size for each of the major protocols in use on XCS. The entries in the *DATA BYTES* column of Table A.1 are computed by summing up the lengths of the data field in the Ethernet packets. The mean packet sizes in the last column are instead computed from the full packet lengths. We observe again that the remote disk access and file system protocols, ND and NFS, have relatively large mean packet sizes and comprise most of the bytes transferred (93.8 percent together), whereas the conventional character and file-transfer traffic of TCP has a relatively small mean packet size.

One of the reasons for which the disk and file access protocols generate such a large portion of the traffic is that the physical memories in our workstations are rather small–ranging from 2 to 4 Mbytes–in comparison to the demands of the software run

TABLE A.1. PROTOCOL STATISTICS

PROTOCOL	PACKETS		DATA BYTES		MEAN
	NUMBER	PERCENTAGE	NUMBER	PERCENTAGE	PACKET SIZE
nd	6,158,602	52.03	5,245,577,140	76.61	869.75
nfs	1,880,402	15.89	1,179,797,526	17.23	645.42
tcp	3,379,564	28.55	356,640,214	5.21	123.52
other	418,505	3.53	65,081,454	0.95	173.51
Total	11,837,073	100.00	6,847,096,334	100.00	596.45

today. In particular, the size of the operating system increases as more protocols are added to the kernel, graphic libraries are very large, and common programs' working sets grow as software becomes more sophisticated. The activity of the ND protocol is particularly dominant. As we shall see in Section IV, most of this activity is *system* activity, which includes paging and swapping, accounting, and the like. This observation leads us to question the common belief that the bulk of system operation is directly related to the user-generated workload; rather, it appears to be caused by the system's own managerial tasks not directly linked to user activity. We shall deal with these issues in more detail in Sections IV and V.

All three protocols, ND, NFS, and TCP, as well as a large portion of the packets listed as *other*, use the IP protocol [69] as the underlying network layer. The IP protocol alone is responsible for 97.8 percent of the total packet count and for 99.6 percent of the total byte count. Since IP traffic accounts for virtually all the traffic, in the next two subsections we analyze the traffic communication patterns of IP packets, which is very convenient because an IP packet header provides both the network and host numbers for the source and destination addresses.

Broadcast packets form only 1.3 percent of the total traffic. It appears that the great majority of the broadcast packets are generated by LISP machines that do not recognize subnetwork addresses [43, 58] in the process of converting IP addresses into Ethernet addresses [66]. If this problem were corrected, the percentage of broadcast packets would significantly decrease. From the data and these observations, we conclude that the broadcast mechanism exerts a negligible influence in an Ethernet. This fact can be used to support two antithetical views: either that there is room for expanding the usage of the broadcasting mechanism [17], or that the mechanism is unnecessary since it can be replaced by serial point-to-point communication with little loss of functionality [71].

D. Intranetwork Traffic and Internetwork Traffic

The traffic is distributed very unevenly among machines. Two machines alone, a file server and a client workstation, are responsible for 19.6 percent of the number of packets. The communication between the three most active diskless workstations and their file servers comprise almost 41 percent of the total byte count. Since an unbalanced traffic distribution is often overlooked by system administrators, who do not have tools to measure traffic, Ethernets may operate suboptimally.

The user of the most active workstation was developing a language-based editor in a LISP environment and worked for many hours during the day. On other days in our traces, the traffic distribution was more even mainly because individual users were active for fewer hours and ran applications that did not load the network as much. However, the Ethernet load on weekdays (as opposed to weekends) during the three weeks on traces displayed roughly the same characteristics, and, on the average, in every hour-long interval we observed about the same number of active workstations as we did in the corresponding interval on the selected day. In this sense the chosen day is typical. Our choice to show a more unbalanced example illustrates that an Ethernet could be substantially stresses by heavy users.

Sun workstations dominate the traffic as they exchange among themselves almost 95 percent of the total byte count; VAX-to-VAX traffic contributes only two percent of the total; and Sun-to-VAX traffic accounts for another two percent.

By looking at the addresses in the IP packet headers, we can divide the traffic into three categories: 1) traffic that remains within XCS (intra-network traffic); 2) traffic that goes outside of XCS but remains within the Berkeley campus (inter-network); and 3) traffic directed to or generated by machines outside the Berkeley local-area networks (inter-network).

All of the observed file system traffic (either carried by the ND or NFS protocol) is local. Although it would be possible to place client workstations and their file servers or ND servers on different networks, the large amount of data that these protocols carry, suggests that gateways would be potential bottlenecks. Inter-network traffic consists of character and file transfer traffic transported by the TCP protocol. In the remainder of this sub-section all data are expressed as percentage of the total number of TCP packets and total number of TCP bytes.

Table A.2 divides TCP traffic according to the source and destination networks of IP addresses: we label with *xcs* the traffic that was directed to or coming from hosts in the XCS network; with *ucb* all the traffic, except for the XCS traffic, that remains within the campus networks boundaries; and with *outside* the traffic that originates or is directed outside UCB. For each combination of source and destination addresses, we show two figures: on the top, the percentage of the total packet count and, on the bottom, the percentage of the total byte count. First, we see that only 10.4 percent of the total TCP packet traffic is local; 89.0 percent is inter-network traffic within campus; and the remaining 0.6 percent is inter-network traffic going to or coming from non-UCB networks. Only a negligible portion of the total TCP traffic is generated by hosts outside UCB. Two reasons explain this: first, interactive sessions across wide-area networks between distant machines are rare because of high line delays, and second, as we shall point out in Section III-C, most users do not receive mail on their diskless workstation.

The three XCS gateways together switched more than 400 million TCP bytes, or 100 million more than the total traffic of Shoch and Hupp. The busiest one alone routed 164 Mbytes, more than 50 percent of the Xerox PARC total. This is possible (recall that the total TCP byte count in Table A.1 amounts to more than 348 Mbytes) because there is traffic generated outside XCS and directed to machines outside XCS. We call this traffic *traffic in transit*. A single TCP packet in transit is counted twice: as an incoming packet on some gateway, and as an outgoing one on another. We observed 540,876 packets in transit, which transported more than 95 Mbytes. This traffic is represented in Table A.2; it accounts for 15.6 percent of all TCP packets and 25.6 percent of all TCP bytes.

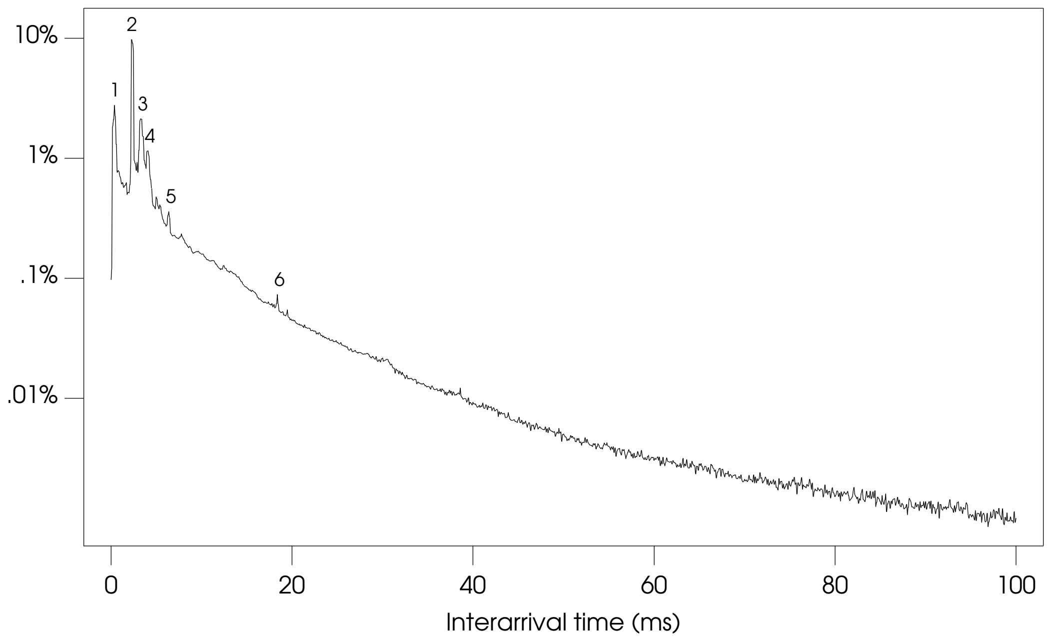
TABLE A.2. SOURCES AND DESTINATIONS OF TCP TRAFFIC (PERCENTAGES)

(packets/bytes)		DESTINATION			
		xcs	ucb	outside	Total
SOURCE	xcs	10.37	40.66	0.37	51.40
		17.33	33.12	0.16	50.61
	ucb	32.77	15.58	0.00	48.35
		23.58	25.62	0.00	49.20
	outside	0.25	0.00	0.00	0.25
		0.19	0.00	0.00	0.19
	Total	43.39	56.24	0.37	100.00
		41.10	58.74	0.16	100.00

E. Interarrival Time

Figure A.5 illustrates the distribution of packet interarrival times over one day. Interarrival time is computed as the difference between the times when the transmissions of two subsequent packets began. Note that because of the high arrival rate, there are a small number of interarrival times greater than 100 ms. By contrast, in measurements taken at MIT [24] there is a significant percentage of arrivals with interarrival times above 200 ms.

FIGURE A.5. PERCENTAGE OF PACKET ARRIVALS (ALL PACKETS)

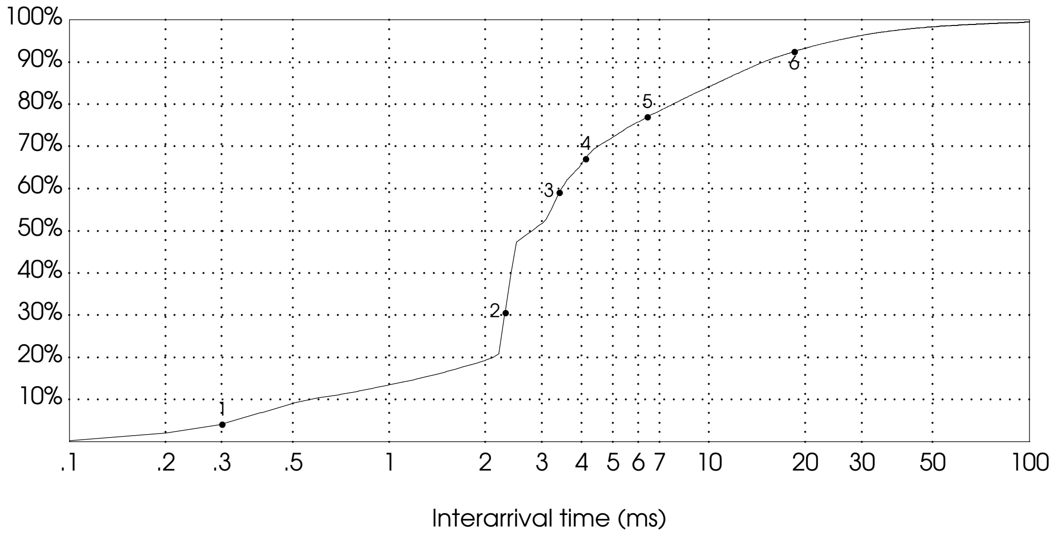


The Poisson model, very often used in analytical and simulation studies, is inappropriate for modeling our data because it assumes that there is no correlation between arrivals. One of our investigations shows that the arrival process is highly bursty and that the independence assumption is not justified [34]. Figure A.5 shows that there is a high probability that one arrival will be followed by a second one within a deterministic time dependent on the protocol, the packet sizes, and the traffic intensity. Three facts explain this high probability: first, the objects transmitted by network protocols are much larger than the maximum packet size on the Ethernet; second, request-response protocols (both ND and NFS can be so classified) generate interarrival times whose distributions have pronounced modes; and third, the sources of packet traffic are bursty [42,88]. In this paper, by studying the various protocol components of the traffic separately (see Sections III-E, IV-B, and V-E), we show that we can characterize interarrival time constants that are not visible if the arrival process is studied globally. However, within the limited scope of this article, we do not try to answer the difficult and challenging question of what the most appropriate model of the arrival process for diskless workstations is.

We have numbered the peaks in Figure A.5; as observed above, they represent patterns of interarrival time. All of these peaks are generated by the ND protocol and will be explained in Section IV-B. They correspond, in order, to peaks 1, 3, 5, 6, 8, and 12 in Figure A.11.

Figure A.6 displays the cumulative distribution of the interarrival times. The numbers refer to the peaks in Figure A.5. Notice that 50 percent of the packets are followed within 3 ms by the next packet, 84 percent are followed within 10 ms by another, and 99 percent within 90 ms. Shoch and Hupp observed for the same percentages time values three times as large.

FIGURE A.6. CUMULATIVE PERCENTAGE OF PACKET ARRIVALS (ALL PACKETS)



There are several factors that contribute to this variation. First, the network utilization is higher, which makes the mean interarrival time lower. The mean packet arrival rate is 137 packets per second and the mean interarrival time is 7.3 ms, whereas the corresponding figures for the Xerox PARC Ethernet were 25 packets per second and 39.5 ms respectively. Second, the bit rate in our Ethernet is 10 Mb/s while it was 2.94 Mb/s in the case of the older Ethernet. Third, the newer protocols, in particular the ND protocol, have been optimized for the shortest response time. Finally, our Suns and MicroVAXs are faster than the Alto computers [90] on the Xerox Ethernet and many of our Ethernet interfaces (Sun Intel and AMD Am7990 Lance) provide much higher performance than the Experimental Ethernet controller in exclusive use at Xerox PARC ten years ago. (The Experimental Ethernet controller for the Alto was half-duplex [81] The time required to switch to receiving mode after packet transmission could have caused packets to be missed by the receiver.)

In spite of the high arrival rate, we observed very few *back-to-back* packets, packets for which the interpacket time is close to the minimum data link layer interframe spacing of 9.6 microseconds for a 10-Mb/s Ethernet.

F. Contention

In the Ethernet local-area network, channel contention is regulated by a *binary exponential backoff* protocol, which randomizes the retransmission of colliding packets. A few works have recently pointed out some instability characteristics of this protocol [2, 31, 79]; for instance, some stations may experience more collisions than others. In general this instability appears at traffic intensities much higher than those we observed in the UCB network. Although we cannot cite actual measurements of collisions (our network interface did not report the number of collisions), we believe that there are two factors that kept the number of collisions lower than predicted by Poisson models. First, as we shall see in the following sections, each station generates packets in bursts that are followed by relatively long silent periods. Second, intervals between packets in a burst are much larger than the Ethernet minimum interframe spacing; as a result, should two sets of stations generate bursts at the same time, the chances are decreased that packets from one burst will collide with packets from the other.

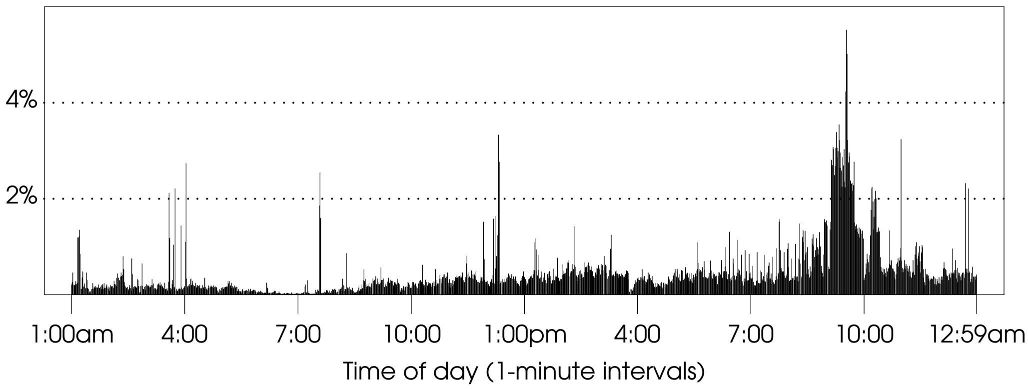
Two factors appear to cause long intervals between packets in a burst: first, the host protocol implementations may not be tuned to achieve maximum performance, and second, in many cases the Ethernet hardware is programmed to use a single buffer for packet transmission, despite the flexibility offered by new interfaces. V. Jacobson has reported that, by rewriting the Ethernet driver to allocate an arbitrary number of buffers for the interface, he achieved back-to-back packets and throughput of 9 Mb/s on an Ethernet between two Sun-3s [39]. (He also reported that, at that speed, the problem of interface buffer overruns becomes very serious.) Of course, for this new implementation the conditions described above that may account for a lower collision rate, do not apply.

Analyzing the individual protocols will give us a more precise picture than the one obtained so far by looking at the entire packet stream. In the following three sections we shall describe the network protocols by focusing on TCP, ND, and NFS. We shall limit the analysis to those features of the protocols that are relevant to understanding communication traffic. Thus, the file access properties of ND and NFS will not be covered in detail.

III. The Transmission Control Protocol

In Figure A.7 we show the network utilization produced by the TCP protocol. The most striking aspect of this protocol is that, despite the high number of packets transmitted, it generates very low network utilization. The activity peak between 9 and 10 pm is caused by file system dumps performed across the network. File system dumps may not be good indicators of TCP throughput since, in the presence of disk contention, data may not be generated rapidly enough to keep TCP buffers always full. In addition, the dump program may be inefficient.

FIGURE A.7. ETHERNET UTILIZATION (TCP PROTOCOL)



A. Data Size

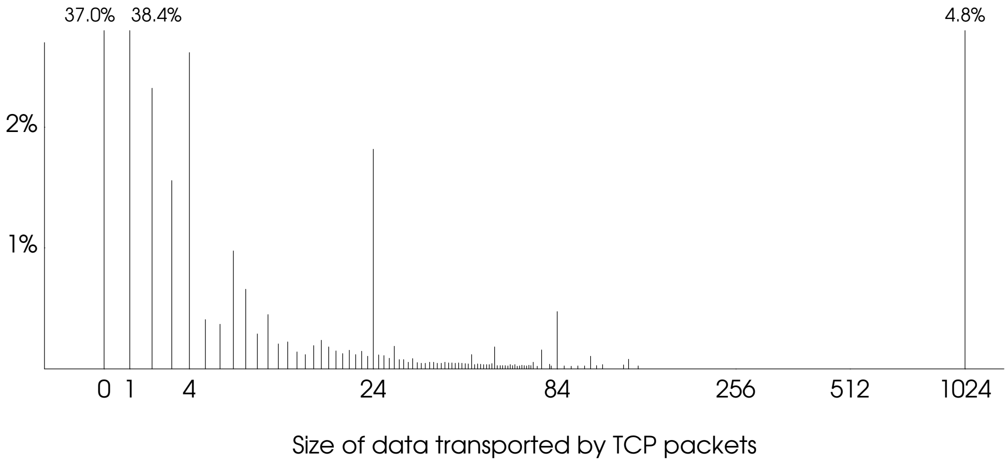
Many TCP packets carry character traffic (characters of UNIX shell command lines) from diskless workstations on the XCS network to VAXs on other networks. Therefore, many packets are short. This is displayed in Figure A.4, which also shows that TCP traffic is essentially bimodal. Figure A.8 shows the size of the objects transported by TCP packets, i.e. the length of the data portion of TCP packets. More than a third of the TCP packets are acknowledgment packets, which transport no data, sent from the receiving host to the transmitting one. Although a portion of the 1-byte objects is produced by editors that put terminals in *raw mode* and do not buffer character transmission, most of them are *keep-alive* messages. Keep-alives, which involve two messages, are implemented in Berkeley UNIX by periodically retransmitting the last byte of a conversation (a 1-byte message) in order to get an acknowledgment (a zero-byte message) from the other side of a connection.

The two peaks at 24 and 84 bytes result almost exclusively from packets transmitted over connections established by *xterm* in the X window system [78], which, at the time of the measurements, was used on XCS's MicroVAXs. It is the traffic of 24-byte data messages that generates the small peak at 64 bytes in the global packet size distribution in Figure A.4.

B. Connections

In order to provide data demultiplexing, TCP uses a set of addresses, called *ports*, within each host to address individual processes. The source machine address, the source port number, the destination machine address, and the destination port number uniquely identify a TCP connection. The number of open connections during the day varied from around 40 to 50, following the crash of one of the largest VAXs of the Computer Science division, to about 130 to 140 at the peak of the TCP activity in the early

FIGURE A.8. PERCENTAGE OF TCP PACKETS VS. DATA SIZES



afternoon. We found that most connections were idle, i.e. only keep-alives were transmitted; the number of active connections over 1-minute intervals never went above 60 and on the average was below 30.

C. Higher-Level Protocols

TCP follows the convention of port numbers that are known in the internet environment and to which network servers bind permanently. These well-known port numbers facilitate the procedure for establishing connections. Each server uses a specific higher-level protocol; therefore, by looking at those ports, we can identify the higher protocol traffic components of TCP. Table A.3 lists the major components of TCP traffic in decreasing order of the sum of bytes transmitted from both servers and clients. All of these protocols can be classified as application-layer protocols in the OSI model. The *DATA* columns show the percentage of total data bytes (using the sizes of the TCP data fields) transported by the TCP packets.

TABLE A.1. TCP PROTOCOL STATISTICS (PERCENTAGES)

PROTOCOL	SERVERS' SIDE			CLIENTS' SIDE		
	PACKETS	BYTES	DATA	PACKETS	BYTES	DATA
login	32.99	21.99	13.61	43.52	20.28	0.42
rcp	3.38	3.73	4.04	4.79	32.89	55.89
printer	0.60	0.32	0.08	0.80	5.57	9.46
ftp	0.40	3.07	5.25	0.42	1.03	1.52
telnet	2.84	1.65	0.72	3.86	1.80	0.05
smtp	0.13	0.08	0.04	0.15	0.14	0.13
Total	40.34	30.84	23.74	53.54	61.71	67.47

The servers' side lists the data that was transmitted by the various network protocol servers (often running on hosts on networks other than XCS), while the clients' side shows the data that user processes sent to servers on XCS or other networks. In some case, as for example for the login server, there is one server per machine. In others, there are only one or a few per local-area network such as the case of the printer server.

Table A.3 includes two remote-login protocols, login and telnet, and two file-transfer protocols, rcp and ftp. This reflects a fundamental trend in, we believe, the majority of local-area environments. Even within the confines of a single organization, the need to transmit information and share data on different hardware and software systems forces system administrators to maintain, and users to employ, a variety of programs.

These protocols encompass 93.9 percent of the total number of TCP packets, 92.6 percent of the total TCP bytes, and 91.2 percent of the total TCP data bytes. The login protocol is responsible for the majority of packets and bytes sent on the network. This is the combined results of both the multiwindow environment of personal workstations, which enables one to open many connections simultaneously, and the user-community custom of using the department VAXs for tasks such as mail, electronic bulletin board, and so on. Notice, however, that the rcp protocol accounts for most of the data bytes transferred–59.9 of the total. This again can be explained by the high number of acknowledgment and keep-alive packets that are sent on idle login connections, whereas rcp is used to copy files.

It is interesting to note that rcp is used primarily to transfer data from the client's side to the server's side (56 versus 4 percent of data bytes). We have verified, by identifying sources and destinations of data transferred on the rcp port, that most of the bytes were moved from the Sun workstations to VAXs. Therefore, rcp traffic is primarily generated by remote file copying programs invoked on XCS workstations to ``push'' files to VAXs, which do not share any file systems with Sun workstations, on networks other than XCS. (The rcp protocol is virtually never used to transfer files between Suns as the remote file system implementation of Sun UNIX allows workstations to mount file systems from various file servers.)

Notice that the low-level activity of the mail protocol [67] (SMTP) is a result of users' custom of sending electronic mail, or replying to messages received, from the machines where they *receive* their mail. Few at Berkeley receive their mail on personal workstations. As a result, most electronic mail traffic is observed as login traffic since users typically connect to the VAXs in order to read or send mail.

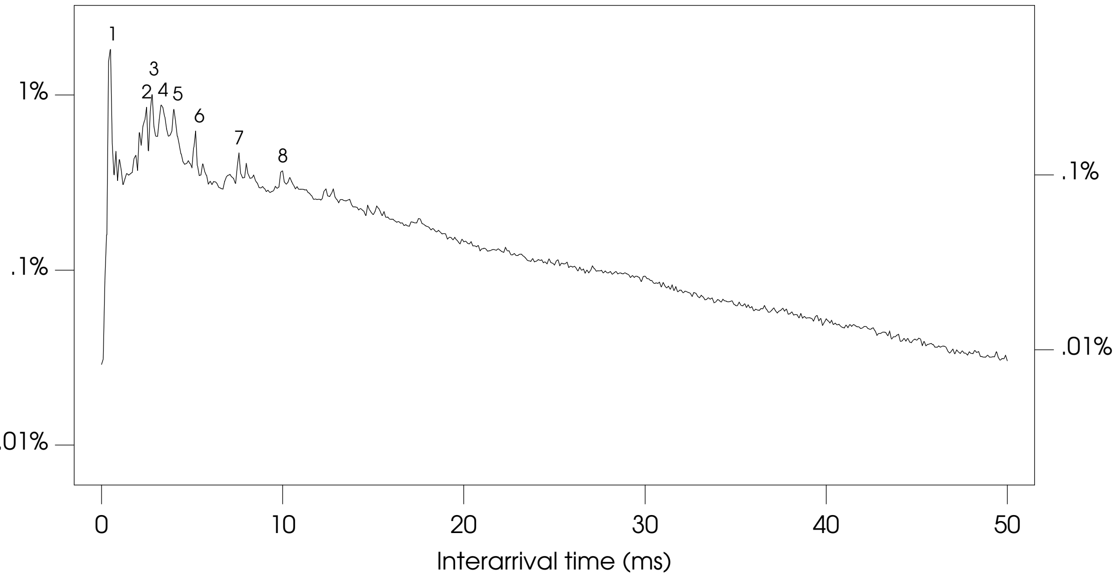
D. Source-Destination Patterns

The analysis of the source-destination traffic patterns shows that most TCP communication occurs between workstations and VAXs. No single machine transmits more than six percent of the total TCP byte count: the imbalance present in the traffic patterns for the total IP traffic is absent in TCP. Unlike the total IP traffic, however, there is a marked imbalance, at least for some machines, in the traffic between two machines. (For instance, one Sun sends 5.8 percent of the bytes to a VAX, but receives only 0.1 percent of the bytes from it.) As observed above, TCP file transfers, which carry the bulk of the data, move, for each pair of machines, mainly in one direction. While NFS displays a similar imbalance, we shall later see why ND, which accounts for 76 percent of all IP bytes, has more balanced traffic between any two machines.

E. Interarrival Times

Figure A.9 shows the interarrival time distribution for the TCP protocol. Interarrival time is computed as the difference between the beginning of the transmissions of two subsequent TCP packets. The left axis of the graphs represents the percentage of TCP packets; the right, the percentage of all packets received. The initial peak is generated by two TCP packets following one another in quick sequence. Since the analysis of source and destination addresses shows that the pairs of packets generating peak 1 are not related, and since most of the pairs follow the transmission of a long packet, it is likely that these packets queue up on different stations, waiting for the long packet transmission to terminate, and subsequently collide.

FIGURE A.9. PERCENTAGE OF TCP PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



The peaks numbered 2, 3, 4, and 5 occur after an interval in which there are fewer interarrival occurrencies. These peaks correspond to TCP packets that are queued as they are generated during the transmission of one of the many 1-Kbyte ND packets (see Figure A.4). Had we observed the arrival of packets at the network interface queues, rather than at the network after the Ethernet transmission contention has taken place, we would have seen a distribution closer to that of an exponential distribution, which, on a semilog plot, is a straight line.

TCP is slow compared to ND and NFS, which carry exclusively local traffic. Since packet acknowledgments come mainly from machines on other networks, gateways add their switching time to the propagation delays of most TCP packets. Thus, 50 percent of the packets are followed by another within 11 ms, ten percent are followed by another within 2 ms, and 90 percent within 52 ms.

IV. The Network Disk Protocol

The ND protocol provides disk block access to a remote disk server. The major disadvantage of this type of access is that it does not allow file system write sharing. In fact, since each client stores disk blocks in a locally maintained buffer cache in order to reduce the number of expensive network accesses, concurrent write operations can create inconsistencies between a client's file data structures and the physical disk image, causing clients to crash. The NFS protocol provides shared remote file access and thus avoids the problems of the ND protocol. However, the performance of NFS, which is implemented with a remote procedure call protocol and is itself a rather complex protocol, is not as good as that of ND.

Sun Microsystems employs both protocols compromising between performance and flexibility. Whereas NFS provides access to general file systems for which read/write sharing is required, ND is used to implement each client's root file system, a distinct one for each client; to access the paging and swapping area of each workstation; and to access shared portions of the file systems that, by containing read-only executable files, do not give rise to inconsistencies. Although the latest release of Sun UNIX removes the ND protocol and uses NFS for the virtual-memory implementation [86], the systems running in the local-area network we measured relied on both protocols.

The ND protocol, which consists only of the two operations read and write, is very simple and is entirely driven from the client side. Read requests and acknowledgments are short packets, and data are transferred 1 Kbyte per packet. As shown in Figures A.3 and A.4, the packet size distribution for the ND protocol is strictly bimodal. For the purpose of the following discussion, we shall distinguish two types of ND packet sequences: a read-request packet followed by a data packet (Req./Data) from the server, and two data packets one after the other (Data/Data). (Notice that write operations are initiated by clients with the transmission of the first data packet.)

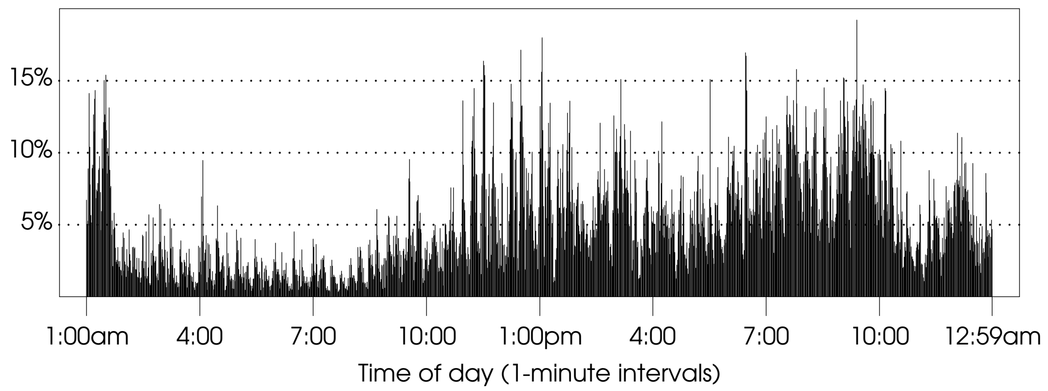
A. Utilization

Figure A.10 displays the network utilization attributable to the ND protocol. The graph looks very similar, both in shape and in magnitude, to that of Figure A.1, which shows the overall Ethernet utilization.

As mentioned before, the ND protocol is used to access different classes of partitions on remote disks: the read/write *private* and the read-only *public* partitions, which are shared by client workstations. The ND protocol is also used by diskless workstations to access their *paging* partitions to which the virtual-memory system of the Sun UNIX kernel directs swapping and paging traffic. By far, the largest amount of traffic is due to this last component.

In the Sun UNIX virtual memory, processes compete for memory pages from a common page pool. Therefore, if one process requests more memory space and the system free list is empty, the *pagedaemon* looks for pages owned by other processes to free (and writes them to the paging device if dirty). This system generates a high amount of paging traffic when workstations' physical memories are not sufficiently large. Paging activity constitutes the largest component of the traffic generated by the ND protocols: more than 50 percent of the total ND traffic.

FIGURE A.10. ETHERNET UTILIZATION (ND PROTOCOL)



It is important to notice that, although there is a monstrous level of paging traffic, because of the way processes compete for each others' pages, Sun UNIX, as well as Berkeley UNIX, generates very little swapping activity–the suspension of processes by saving their entire state information on the backing store. The low level of swapping has often led researchers who have measured Berkeley UNIX or Sun UNIX virtual-memory performance to conclude erroneously that virtual-memory activity would have a negligible effect on network traffic should network file systems replace local disks [49,64].

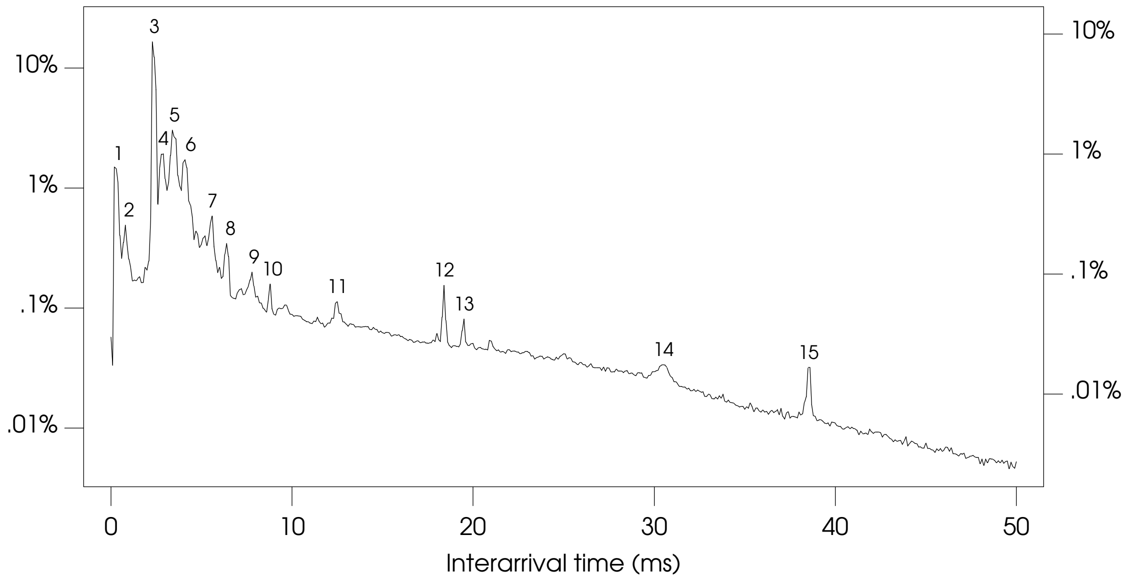
One might find it surprising to discover that the ND protocol accounts for most of the network traffic. The relative importance of the ND protocol components can be explained by 1) small physical memories on most client workstations, which cause excessive paging activity; 2) the tendency of the virtual-memory algorithms of Sun UNIX [4,5], designed when the cost of memory was high compared to the cost of disk space, to make maximal use of the memory even at the cost of generating some extra I/O traffic; 3) small available file buffer caches, which force the kernel to discard pages belonging to commonly used programs; and 4) UNIX applications that use extensively the directory /tmp, which in a diskless Sun environment resides in the root partition.

We noticed that for some of the file servers the number of bytes written to the paging partitions exceeded the number of bytes read from the same partitions. Whereas the high level of paging activity certainly depends on the small physical memories in use on our workstations, the fact that virtual-memory pages are only rarely read back from the paging device may result from system inefficiencies. Since space allocated to each process on the backing store is organized as a contiguous sequence of blocks, disk transfers are particularly efficient because disk arms need not seek between accesses to a process's data. It is not clear, however, that with today's fast file systems [55] the backing store provides significantly shorter access time. Alternatively, virtual memory could page to and from the file system [59]. This would have the additional advantage that a process's text pages need not be written to the backing store the first time they are paged out.

B. Interarrival Time

Figure A.11 plots the packet interarrival-times between ND packets. Figure A.12 shows the cumulative interarrival time distribution. In both figures, the vertical axis on the left represents the percentage of ND packets; the axis on the right, the percentage of all packets received.

FIGURE A.11. PERCENTAGE OF ND PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



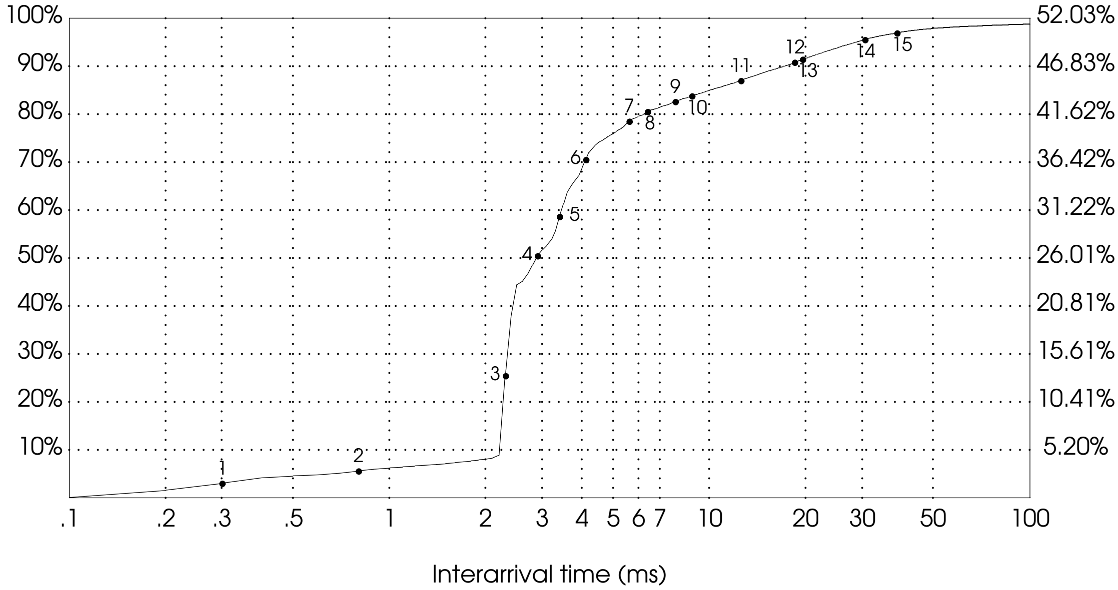
The ND protocol is very fast: 90 percent of all ND packets are followed by the next packet within 18 ms. Three factors contribute to this remarkable speed. First, the protocol is extremely simple, since it uses the IP transport mechanisms to bypass much of the kernel's code for higher-level protocols; second, read requests typically involve 4 or 8 Kbytes of data resulting in bursts of packets sent with small interarrival times (notice that the time between the read request and the first response packet is generally longer since it includes disk access time); and third, write operations are delayed and servers can send acknowledgements as soon as the data have been received.

We have numbered the significant peaks in Figures A.11 and marked them in Figure A.12. These peaks represent patterns in the communication behavior and time constants that are determined by the type of transaction, the disk speed, the CPU speed, and the network interface used. As expected after seeing the influence of the paging activity, the largest component of many of the peaks is the ND paging partition traffic.

The ND peaks numbered 1, 3, 5, 6, 8, and 12 are also clearly identifiable in Figure A.5, the graph of interarrival times of all packets. This means that no packet belonging to other protocols was transmitted between two ND packets in a good portion of arrivals that contributed to these peaks.

Peaks 1 and 2 are due to Req./Data packet sequences. Data/Data message sequences between Sun-3s account for a large portion of peak 3, which occurs at about

FIGURE A.12. CUMULATIVE PERCENTAGE OF ND PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



2.2 ms. Since it takes 0.89 ms to transmit a 1-Kbyte ND packet, protocol overhead for Sun-3s amounts to about 1.3 ms. Some Data/Data message sequences between Sun-3s are delayed and have interarrival time of about 3 ms (peak 4). There are three possible reasons for the delays: first, there could be queuing at their network interfaces, which are used by other protocol modules; second, the network channel could be busy when the two machines generate messages for transmission; and third, the two machines could be loaded.

The Data/Data transactions at peak 5, which occurs at around 3.4 ms, are generated mostly by Sun-2 machines; the associated protocol overhead is in this case 2.6 ms.

Notice that while a write Data/Data message sequence is generated by client workstations, the read Data/Data one is generated by servers. This explains why the communication between a Sun-3 client and its Sun-2 server produced write Data/Data sequences at peak 3 and read Data/Data sequences at peak 6.

One of the file servers had a slower, non-standard disk, which was exclusively used by a single client workstation. The absence of disk contention generates the sharp peaks numbered 11, 12, 13, and 15.

V. The Network File System Protocol

This section analyzes the network performance of the NFS protocol, which provides diskless workstations with remote access to shared file systems over a local-area network. It does not directly address the file access properties of Sun remote file-system implementation; while the patterns of file access are very important in file-server design, they are outside the scope of this study. Understanding the traffic characteristics of the NFS protocol is particularly important because, unlike most of the ND protocol traffic, NFS traffic represents user rather than system activity.

The NFS protocol is much more complicated than the ND protocol: it uses remote procedure calls (RPC) [87], which in turn use the User Datagram Protocol [68] (UDP) as the transport protocol. Furthermore, NFS, in order to work with heterogeneous file servers, converts the information it transmits into a machine-independent format [53].

Among the procedures used by NFS, there are procedures to read and write data, to create files, to convert file names into pointers to files, to return file attributes, and to create, remove, or read directory entries. A description of the NFS protocol procedures can be found in Sun manuals [63]. If files are to be accessed on remote file systems, Sun UNIX transforms the system calls that operate on them into one or more of NFS remote procedure calls.

A. Utilization

Figure A.13 shows the network utilization, over 1-minute intervals, of the NFS protocol. The pattern of NFS communication generally follows that of the total Ethernet traffic although the graph's spikes indicate that it is more bursty. The 1-minute average utilization is well below five percent, with occasional bursts in the range of five to ten percent and one peak at 12 percent after 10 pm.

FIGURE A.13. ETHERNET UTILIZATION (NFS PROTOCOL)

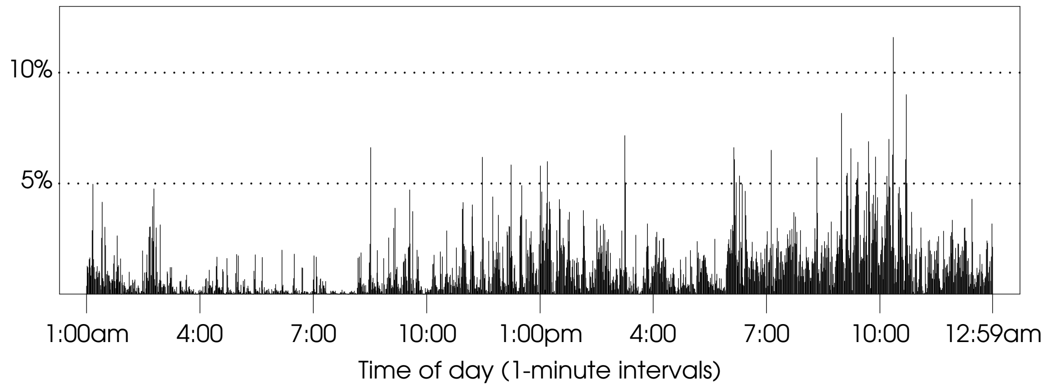
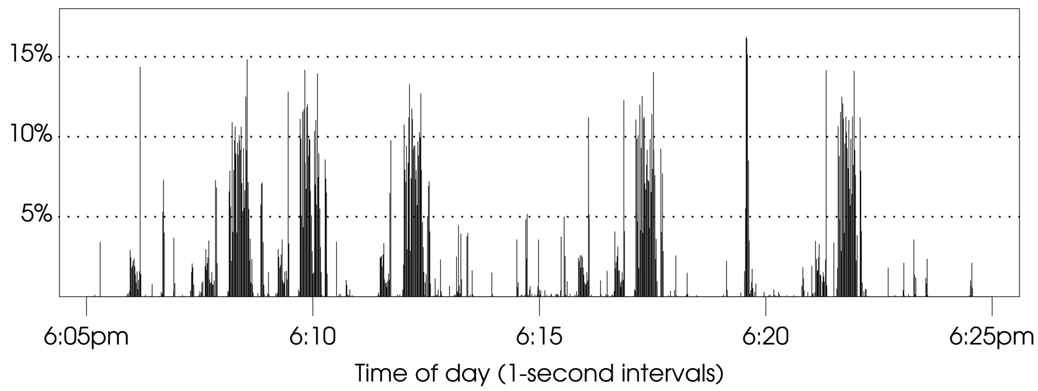


Figure A.14 displays the network utilization, over 1-second intervals, generated by NFS traffic between two Sun-3 workstations: a client and a heavily used server. The network activity is intermittent, following the usual pauses in user behavior. In addition to occasional 1-second peaks at 15 percent, there are times when the network utilization stays at around ten percent for several seconds. This corresponds to a throughput rate of about 120 Kbytes per second. Several other workstations, even those less heavily used, displayed similar data transfer rates over short intervals.

B. Packet Length

The packet size distribution generated by the NFS protocol reflects that NFS is based on remote procedure calls. The distribution of packet lengths is bimodal. The two modes differ, however, from the modes of ND and TCP. One mode corresponds to a large number of short packets whose lengths are distributed around 144 bytes; these are

FIGURE A.14. ETHERNET UTILIZATION (NFS TRAFFIC BETWEEN TWO WORKSTATIONS)



requests and responses transporting NFS and RPC headers. The other mode occurs at 1500 bytes and is produced by the data fragmentation performed by the IP module during file read and write operations. The minor peaks at 696, 920, and 1276 bytes are caused by the last fragments of fragmented messages. These peaks are displayed in Figures A.3 and A.4.

C. NFS Procedure Statistics

The routine that performs file path-name lookup is the most heavily used among the NFS remote procedure calls; it represents 48.5 percent for the total number of calls. Lookup operations dominate since path-name translation is performed one component at a time.

Read and write calls correspond to 33.7 and 4.3 percent of the total. On the average, there are 7.8 read operations for each write operation. A UNIX file system study [95], in which user requests were traced at the system-call interface, reports that there are about 2.7 reads for each write operation. One would expect our read/write ratio to be higher, not smaller, than the study's, since, by looking at the network activity rather than at the system-call activity, we do not capture file read operations that hit the local file caches; moreover, under NFS all write operations go to the server [77]. However, our caches are not very effective, mainly because of their small sizes, and thus, the number of reads we observed is close to the number of reads issued by the file system layer. Our higher read/write ratio is explained by two factors. First, especially in the paging in of programs, user read requests are large and each request corresponds to more than one NFS packet (recall that page-outs are handled through the ND protocol). Thus, we have a higher number of reads. Second, the UNIX accounting information written to a system's file each time a process terminates is handled in diskless Sun workstations by the ND protocol; therefore, we also have a lower number of writes.

An NFS client caches file data. Repeated use of these data requires a verification from the file server that the data are up to date. This is done with the getattr NFS procedure call. The developers of the NFS protocol discovered that the getattr call accounted for 90 percent of the total number of calls made to the server [77]. The situation was remedied by adding a so-called attribute cache at the client side, which keeps file attributes for 30 seconds after they arrive from the server. The number of getattr procedure calls we observed came to 4.6 percent of the total.

One of the reasons for the mentioned ineffectiveness of the NFS caches is the large size of frequently used programs and graphics libraries. For instance, the Sun window library occupies 640 Kbytes, the C compiler library, typically stored in an ND public partition, alone occupies 350 Kbytes and the compiler commonly requires 1 Mbyte of physical memory. The size of the UNIX kernel on a diskless workstation is approximately 500 Kbytes; Sun UNIX normally allocates ten percent of the total available memory for caches and buffer pools. Therefore, the user of a 4-Mbyte machine is left with only about 3 Mbytes of memory for user processes, a good portion of which is taken by Sun's graphics programs. (These sizes were measured on a Sun-3 workstation.) Under these conditions, we should expect both poor cache hit-ratio and heavy paging traffic.

Three of the NFS routines, read, write, and readdir (this last one used to read the content of directory files) may transmit in one call more data than fits in a single Ethernet packet. In this case–we have seen that UDP provides the network transport layer for NFS–the IP layer fragments the messages. The average number of packets for the read responses, write requests, and readdir responses is 2.9, 3.1, and 1.6 respectively.

The small number of fragments for the readdir calls indicates that most directories that users list contain only few files. One reason for this is that many users avoid listing large directories since the response time of the UNIX *ls* command on directories with many entries is particularly slow on diskless workstations.

D. Protocol Timers

We have observed that an excessive number of NFS requests were retransmitted. Although some of these retransmissions were triggered by factors not related to the NFS protocol, such as lost packets, many retransmissions were caused by RPC timers expiring before the entire message was received because of delayed packets and delayed server responses.

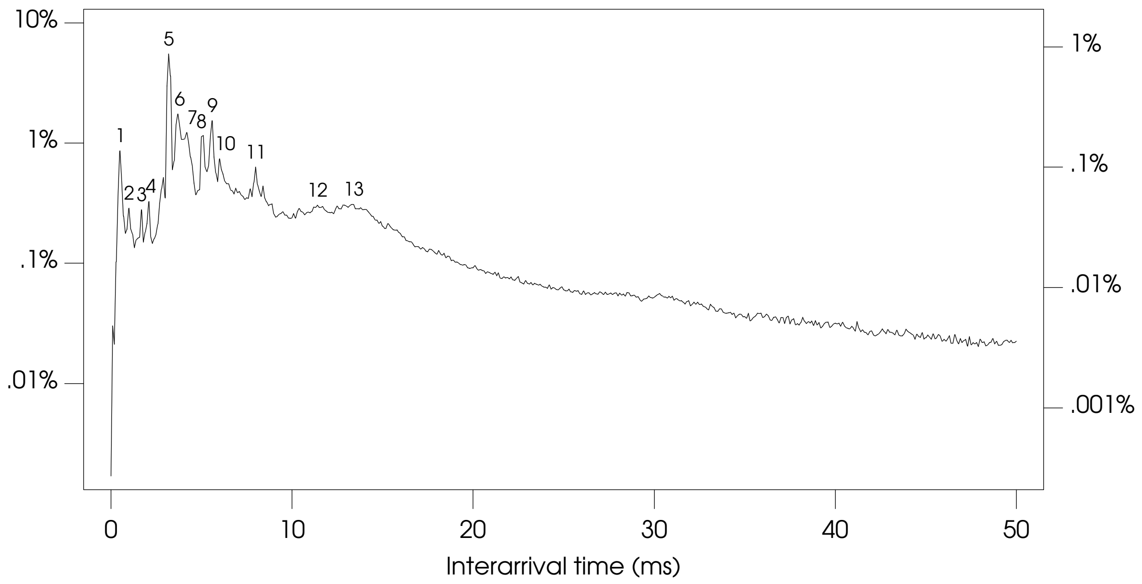
For example, in case of a client 8-Kbyte read request, the server fetches the data from the file system and issues a single RPC response. The RPC module on the server side adds an RPC header and passes the request to the UDP module. The UDP module adds a UDP header and passes the data to the IP module. IP takes care of the transmission over the Ethernet channel by fragmenting the message into five 1500-byte packets and one last 920-byte fragment packet. The receiving IP module on the client side reassembles the message, passing it up to the UDP and RPC modules only after the last packet has been received and the message fully reassembled. Therefore, in the case of high network utilization when queueing times become substantial, the RPC layer times out while the response it generated is being received by the lower layers.

More than six percent of the write calls and more than one percent of the read calls were retransmitted. Most of these retransmissions occurred 0.7 seconds after the original transmission. The proportion of write retransmissions is higher than the proportion of read retransmissions because reads are buffered (i.e. pages are cached at the server, which implements a read-ahead policy) while writes always go to secondary storage. Based on an analysis of the correlation between network utilization and number of NFS operations, we conjecture that both network and disk contentions are responsible for these retransmissions.

E. Interarrival Time

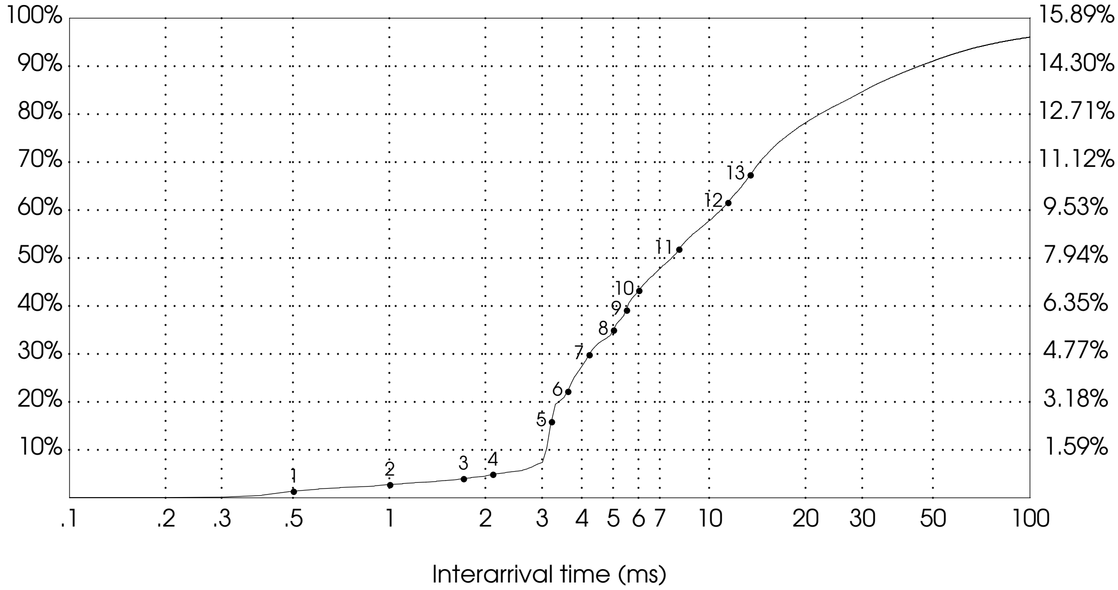
Figure A.15 displays the NFS packet interarrival-time distribution, and Figure A.16 the cumulative interarrival-time distribution. In the two figures, the interarrival time is the difference between the beginnings of the transmissions of two subsequent NFS packets. In both figures, the vertical axis on the left shows the percentage of NFS packets and the one on the right the percentage of all packets received. We have numbered the peaks in Figure A.15, and reported their position in Figure A.16. In the following discussion three different message types are used: *Request*, for an RPC client request; *Response*, for an RPC server response or for the first packet of a server response; and *Fragment*, for subsequent packets of a multi-packet RPC response. NFS Fragment packets play the same role as Data packets in the ND protocol: they are large packets that result from protocol-specific fragmentation of long messages.

FIGURE A.15. PERCENTAGE OF NFS PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



Most of the peaks with lower interarrival times are primarily produced by Sun-3 machines. The first significant occurrence of Fragment/Fragment message sequences occurs at peak 5 at around 3.1 ms. For the ND protocol and the same pair of machines the Data/Data message exchanges generated interarrival times as low as 2.2 ms. The time difference between these two measures is due in part to the longer messages of NFS (50 percent longer) and in part to higher protocol processing time. Using calculations analogous to those of Section IV, we estimate that the protocol overhead amounts for Sun-3 CPUs to about 1.9 ms for this type of message sequence, as opposed to 1.3 ms for the ND protocol.

FIGURE A.16. CUMULATIVE PERCENTAGE OF NFS PACKET ARRIVALS (AT RIGHT AS PERCENTAGE OF ALL ARRIVALS)



Peak 7 allows us to compute the equivalent protocol overhead for Sun-2 machines. Here, the predominant message sequence is a Response/Fragment one, which indicates that Sun-2s request fewer data per transaction from file servers. The sequence is otherwise equivalent to a Fragment/Fragment sequence. The peak occurs at about 4.1 ms; the overhead amounts to about 2.9 ms. In the case of Sun-2 machines, the computed ND protocol overhead for Data/Data transactions was 2.6 ms.

It is important to notice that, although packet protocol overhead is higher in the case of NFS, the increased packet size tends to reduce the per-byte overhead. Since each NFS packet transports 50 percent more data than ND packets (in the case of Data/Data and Fragment/Fragment message sequences), the per-byte protocol overhead is about the same as that of ND: 1.3 microseconds per byte for a Sun-3 and 2.2 microseconds per byte for a Sun-2. Therefore, the lower NFS protocol throughput must be caused by other factors.

Figure A.15 shows a slowly decreasing curve, which indicates that NFS operations tend to be more spread over time than the ND ones. There are several reasons for this behavior. First, since many user-level file accesses involve a small number of bytes, the NFS ratio of physical disk accesses to number of operations is higher than that of ND; second, while ND's writes are delayed, NFS uses synchronous writes; third, the more complex protocol message interactions in NFS communication create greater time variability; and fourth, NFS remote procedure calls are synchronous, i.e. the client process is blocked until the response is received.

An additional reason for lower performance, one not evident from the traffic measurements, is that NFS client caches are less efficient than ND ones. Their data must be validated before each access. The attribute cache discussed above only alleviates the problem.

VI. Conclusions

In this study we have analyzed the behavior of the Ethernet local-area network in an environment of diskless workstations running the UNIX operating system. We have closely compared the traffic characteristics of a medium-size 10-Mb/s Ethernet with those of an older 2.94-Mb/s Ethernet whose traffic was studied at Xerox PARC at the beginning of the 80's. Use of the Shoch and Hupp study for comparison is meaningful because it introduced basic traffic characteristics for Ethernets that subsequent studies have not significantly questioned. As a result, system designers currently develop network software and hardware under the implicit assumption that the network load is light.

The most striking conclusion of our study is that diskless workstations may indeed generate enough traffic to load substantially a local-area network of the current generation. A system built around diskless workstations is highly interactive, and short-term network utilization is more likely to affect users' behavior than long-term average utilization. Therefore, we consider response time more important than throughput in studying the performance of today's systems. The mean network utilization is low (6.5 percent averaged over 24 hours) but bursts generate short-term peak utilization above one third of the Ethernet raw bandwidth. Studies have shown that, at this level, network latencies will begin to increase [48]. For instance, according to Gonsalves [30], who measured Ethernet delays at varying, artificially generated network loads for different packet lengths, the queuing time at traffic intensity 0.3 is larger than 5 ms when all packets transmitted are 64 bytes long, and about 19 ms when the packet length is 1500 bytes. We have seen in Section V how the increased latency affects the NFS protocol.

Our measurements show that a workstation's traffic falls into three broad categories: character traffic from the workstation to other machines; paging traffic generated by the workstation's virtual memory to a remote paging device; and file access traffic to remote file servers. A workstation's behavior will depend on the characteristics of each of these three types of traffic. These components were easily identifiable because a different protocol was employed for each component. Character traffic generates many small packets but no substantial network utilization. File access to a remote file server generates bursts of traffic lasting several seconds, which may demand bandwidths in the order of 120 Kbytes per second, or about ten percent of the Ethernet raw bandwidth. Paging traffic, which accounts for maximum network utilization levels of 20 to 25 percent over 1-second intervals between a single client workstation and a file server, has been greatly increased by small physical memories (by today's standard a 4-Mbyte memory is small indeed) and, possibly, by sub-optimal performance of the virtual-memory algorithms. It could be argued that larger memory sizes will decrease the level of paging traffic; however, since there is a tendency to write new, larger applications that take full advantage of memory size increases, it is more likely that paging will remain a noticeable component in future workstations. It does seem, however, that paging traffic could be reduced by improved buffering schemes; one such scheme, which calls for a global caching area for both virtual memory and file system, has been implemented in Mach [94] and in release 4.0 of Sun UNIX. In addition, bridges could be used to cluster diskless workstations and their file servers on separate segments of an Ethernet. This arrangement has the advantage of confining the virtual-memory traffic of the client workstations within the boundary of each individual network segment. Unless these approaches are used extensively in the future, paging traffic may remain intolerably high, and the diskless workstation technology may be doomed to limited development in current local-area networks.

The analyses of the interarrival time for each of the three categories of traffic have revealed that there are very few occurrences of back-to-back packet arrivals despite the high packet arrival rate. Although we did not obtain performance measurements on the number of collisions, we believe that the observed packet arrival times–with spacings among subsequent packet arrivals that correspond almost to a full packet transmission–contribute to keep the number of collisions and protocol latencies smaller than predicted by Poisson models. As network interfaces and protocols are developed that can transmit data, for the same type of user requests, at the rate of the Ethernet data link layer, we would expect that a higher percentage of bandwidth will be wasted in resolving collisions and that protocols' response time will increase. Thus, on the Ethernet, as protocols become more efficient, they must also become more sophisticated to deal with increased latency.

Even though it is difficult to quantify the network bandwidth necessary to support diskless workstations–this is largely a subjective parameter that depends on the performance demanded by users–it seems clear that response time is the single, most important performance characteristic for evaluating these bandwidth requirements. We have seen that bursts of file system traffic may transport data on the order of one to two million bytes. Because of coupling between CPU speed and new application interfaces, future software applications will need to move even greater amounts of data. For this reason, the development of effective caches, as demonstrated by current research efforts and by our measurements, is a very important objective. Yet larger caches will not eliminate users' need for larger amounts of bandwidth.

The Ethernet local-area network is a mature product. It provides an extremely effective and low-cost communication medium for interconnecting computers within a local area. Designed when users' communication needs were quite different from today's, the Ethernet has been adapted to provide effective bandwidth to distributed systems based on diskless workstations. Although pushed by improper memory configurations on machines with high-performance network interfaces, the Ethernet still functions satisfactorily. However, based on the measurements described in this study, we believe that precise decisions on the number and type of workstations that can be placed on a single-cable Ethernet must be made soon, if not immediately.