A Characterization of the Variability of Packet Arrival Processes in Workstation Networks

by

Riccardo Gusella

This thesis was typeset in Avantgarde and Courier on a DECstation 5000 using troff. All figures, except for Figure 4.3 which was prepared with Mathematica, were produced using grap.

Acknowledgments

I have been most fortunate in having Domenico Ferrari as my research advisor. His constant scientific and moral support have been the source of inspiration in all my work. Domenico's guidance during difficult moments, when the solution to the most challenging problems seemed out of reach, steered my work into direction. From him I have learned that in "Computer Science" one should stress the second word.

I am deeply indebted to Sam Morgan, who supervised my work when I was at Bell Labs, Murray Hill. Sam has been exceptional; his intellectual rigor has been a source of inspiration. It was Sam who suggested the research topic that led to the development of this thesis. He served on my thesis committee, and provided me with the most detailed assessment of the weaknesses in my thesis draft. His constant support and never-ending encouragement have helped me overcome the most perplexing of obstacles.

My thesis work would not have been possible without the direct involvement of a terrific group of people at Sun Microsystems. Marty Itzkowitz and Mark Liu went out of their way to help me collect the measurement data, and provided invaluable technical support. I am also deeply grateful to Chris Rigatuso and Marilyn Shoemaker for all their assistance.

I am indebted to David Anderson for serving as Quals Committee Chairman and for his comments on the thesis manuscript. I am grateful to David Aldous for serving on my Quals Committee, and to David Brillinger, who read the thesis draft, for his invaluable advice.

I would like to thank William Kahan and Ward Whitt.

I owe much to many friends who, directly or indirectly, have influenced my work. I will not attempt a complete list, for I would surely forget someone. Instead, I collectively thank all the members of the Tenet group, and all my friends in the Computing Science Research group at Bell Labs. But I would like to acknowledge especially Ramon Caceres, Peter Danzig, Kerry Fendick, Kathy Meier-Hellstern, Keshav, Bart Miller, Joe Pasquale, Dave Presotto, Doug Terry, Shin-Yuan Tzou, Dinesh Verma, Stefáno Zatti, Songnian Zhou.

My gratitude goes to Kathryn Crabtree, who assisted me in navigating the Computer Science Division bureaucratic apparatus.

I would like to express my gratitude to Phyllis for having helped me to cope with a thesis that at times looked so frustrating. To her I dedicate this work.

My research has been supported by a number of research grants. I kindly acknowledge the support of AT&T Bell Laboratories; Hitachi Ltd.; Olivetti S.p.A.; the University of California, Berkeley, under a Micro grant; and the International Computer Science Institute.

Abstract

We examine the problem of characterizing the variability of measured packet-arrival processes produced by individual workstations in a local-area network. Commonly referred to as burstiness, variability can be informally described as producing sequences of abrupt transitions from low to high arrival rates and vice versa. Since variability can be related to the relationship among successive arrivals, we adopt a quantitative definition, based on indices of dispersion, from the theory of point processes.

We illustrate the measurement methodology and discuss the error analysis. We then analyze the first- and second-order statistical properties of interarrival-time and packet-count series, which reveal the structure of the underlying point processes. We estimate indices of dispersion for intervals and counts, which express the autocorrelation structure of a point process, and warn about the effect of nonstationary data. Using an artificial example based on the Markov-modulated Poisson process, we show how to incorporate into a mathematical model the second-order stochastic parameters that represent dispersion. Fitting is done so that the index of dispersion for counts of the MMPP model matches closely that of the data, a procedure that produces what we call a "model of variability".

Finally, we derive a model of variability whose structure follows the structure of the data: the interarrival times of short and long packets are disjoint; the lengths of sequences of short and long packets form a discrete-time Markov chain; and a generalized two-state semi-Markov process, in which interarrival times in each of the states are autocorrelated, is shown to reproduce with good approximation the correlation structure of the data for time scales up to 500 ms. The approximation requires only estimates of the first- and second-order moments of the interarrival times. To complete the model, which is two-dimensional, we also provide simple characterizations for the lengths of short and long packets. Because of the recursive nature of the model's equations, the model is suited for simulation studies.

Table of Contents

1 Modeling Variability

Variability in the rate of arrival processes has been a characteristic of data communication between or among computers from the earliest days of networking. Back in the time when remote computers were accessed through terminals, communication lines carried character traffic, and computers were not linked to each other through a network, the rate of arrival of characters did not fluctuate a great deal. The primary cause of variability was human: users paused for different lengths of time to think between typing intervals.

The introduction of networking technology and its refinement has, however, directly led to an increase in the variability of arrival processes. First, the appearance of real computer networks, file transfer protocols, and computer mail resulted in more variability, as users could now instruct a system to send a message to a remote location, or copy data from one location to another. Measurement studies of wide-area network traffic revealed that the traffic was bursty [47]. Next, the advent of higher-speed local-area networks and the simultaneous development of the personal computer workstation [57,90] opened fascinating new avenues for data communication. Remote file systems, distributed systems, more sophisticated protocols, including the Remote Procedure Call protocol (RPC), and distributed applications became the norm. All of these advances contributed to increasing variability since faster protocol allowed for transmission of more data in a shorter time, while many distributed-system operations generated successive transmission requests within a wide range of times. Studies showed that local-area traffic was also bursty [80], but this was not a matter of concern, because local-network traffic was extremely light and there were no noticeable queuing delays.

Variability increased even more dramatically when diskless workstations made their debut. With diskless workstations, virtual-memory traffic had to be moved as quickly as possible across the network. A few keystrokes could produce either a quick burst of up to several hundred Kbytes of data across the network when a new program was invoked, or only a minimal data representing, for instance, characters transmitted to a remote computer.

While the networking research community began to experience congestion and to think about ways to cope with it, telecommunication researchers, who had long been studying telephone voice channels, had already well understood the effects of bursty arrival processes to queues. Consider, for instance, the following two cases. In the first case, we have a single-queue system whose input is a Poisson process with arrival rate λ; in the second case, the input process is batch Poisson, i.e. arrivals occur, according to a

Poisson process, in batches of mean size B. If the arrival rate of the batch-Poisson process is chosen to be λ/B, the mean arrival rate is the same in each of the two cases. The average times spent in the systems, according to Little's law, will be proportional to the number of customers in the systems. The mean number of customers in the system for a Poisson process is $N_P = \frac{\rho}{1-\rho}$. The mean number of customers in the system in the batch-Poisson case can be calculated easily [46, problem 4.3]:

$$N_B = \frac{1}{2}\left[1 + E(B) + \frac{VAR(B)}{E(B)}\right] N_P .$$

The quantity between parentheses is always greater than 2 (it equals 2 when all batches are of size 1, i.e. when the batch-Poisson process reduces to a Poisson process), and hence the average waiting time of customers arriving according to a Poisson process is shorter than the waiting time of customers arriving according to a (more bursty) batch-Poisson process with the same mean arrival rate.

Despite its profound effects on computer network traffic, queuing, and congestion, variability has been studied mostly in terms of analytical or simulation models. In this dissertation, we will consider measured packet-arrival processes, study their variability based on indices of dispersion, and propose models that will characterize this variability.

II. Central Problem of Dissertation

This thesis examines the problem of characterizing the variability of packet-arrival processes produced by individual workstations in a local-area network. The study is based on measurements and emphasizes understanding and modeling real data. These data are packet-arrival processes extracted from the arrival times of aggregate traffic produced by engineering workstations on an Ethernet local-area network. The term "processes" refers to realization of stochastic processes along a time axis. Technically, they are marked point processes as each point has several attributes, such as a packet length, a protocol type, a source and a destination network addresses, and so on. Often we will derive the properties of the point processes by operating on time series of interarrival times—the series of intervals between successive packet arrivals.

Of all the features of complex phenomena such as packet arrival processes—a result of the complexity of the systems that produce them—we will focus on variability. Often referred to as burstiness, variability can be informally described as producing a string of abrupt transitions from low to high arrival rates and vice versa. We found no satisfactory quantitative definition of variability in the computer science literature, but since variability can be related to the relationship among successive arrivals, we adopted a definition from the theory of point processes based on indices of dispersion.

We have chosen to study packet arrival processes generated by engineering workstations because these workstation constitute an important example of distributed system whose network usage patterns are not completely understood. We conducted the measurements on a local-area network, and not on a wide-area network, in order to keep the measurement effort manageable, but also because arrival-process variability is more evident in a local-area environment where there are fewer queuing interactions and less traffic smoothing produced by the superposition of traffic generated by

machines communicating concurrently. There are other types of interesting systems that produce network traffic worth modeling; transaction traffic generated by distributed database systems comes to mind; however, given the intense research effort [38] in the design of the new generation of workstations that will handle continuous media applications in addition to current user interfaces, we felt that models (even if only models of variability) of current workstation traffic could have a direct impact on the workstation design.

Though a major portion of the dissertation is devoted to the analysis of the first- and second-order statistical properties of interarrival-time and packet-count series, which reveal the structure of the underlying point processes produced by individual workstations, we will also define models of traffic. These models are not intended to reproduce the traffic exactly as generated by the workstations. Instead, the models will focus on the notion of variability of packet arrival processes. Using indices of dispersion, which express the autocorrelation structure of a process, we can quantify data variability; we can then incorporate into mathematical models the second-order stochastic parameters that represent dispersion, and derive indices of dispersion for the models. Fitting is done so that the models' variability closely matches that of the data. This approach produces what we have defined as ''models of variability''. The gist of the argument is that the autocorrelation structure of a point process depends on the lengths of sequences of observations smaller than the mean, on the lengths of sequences larger than the mean, and the way the two types of sequences alternate. Hence, the form of the solution that we seek to the problem of characterizing the variability of packet arrival processes is one in which models will reproduce the correlation structure the the measured point processes, not necessarily the exact marginal distribution of their interarrival times.

III. Thesis Roadmap

Chapter 2 deals with the measurement methodology, the data collection, the preparation of the data for analysis, and the initial data analysis. The data was collected on a large Ethernet, which we will denominate ''Engineering network'', at the Sun Microsystems headquarters in April 1989. The network consisted of about 130 machines, divided among diskless workstations and machines with local disks. The workstations were served by six major file servers. In the thesis we study arrival processes that occurred approximately between 8:15 am and 6:30 pm on a single day. Based on observing the measured network over a period of several days, we are confident that the chosen day is typical. The data was packaged into traces, which, stored on tapes, consist of packet headers and timing information.

The data collection was preceded by several months of study for the preparation and design of the measurement system. The Performance Evaluation group of Sun let us use the fastest machine available at the time. We knew from our previous measurement project (described in Appendix A) that data analysis would greatly benefit from the availability of an accurate clock with which to timestamp packet arrivals. As the resolution of the machine clock was only 10 ms, a serious limitation, we employed an external timing device, described in Section 2.III, whose resolution was 0.5 microseconds.

Data analysis required reading the traces innumerable times. This would not have been practical if we had kept the data on tapes, as tape-drive access times are too

slow. Since we could not monopolize many hundred Mbytes of disk space for our own use, we were forced to compress the trace data; this is described in Section 2.VI. While compressing the data, we also transformed the timing information, which was recorded after each packet had been fully received, to the times when the packet transmissions started. This is possible because most of the packet-delay components are deterministic. The random components of the delay, most importantly the interrupt time latency, contribute to the measurement errors.

During the timing conversion, we noticed that for a sizable portion of the packet arrivals, the beginning of the transmission time should have occurred before the transmission of the previous packet had completed. This, on a single-channel, baseband network, is impossible. Hence, the situation must have been created by measurement errors. In Section 2.VII, we analyze several cases of errors in this class and, in all cases, we show that errors were generated by pairs of packets that were transmitted with the minimum interpacket time admissible on the Ethernet, 16 microseconds. Our analysis attributes the error source to the tracing machine's internal bus, which was shared by the CPU, the network interface, and the timing device.

At the end of Chapter 2, in Section VIII, we describe some of the characteristics of the aggregate traffic of the Sun Engineering network. We discuss the network utilization, the source-destination traffic patterns, the interarrival-time probability density, and the distribution of packet lengths. This data analysis should be compared to the analysis of older measurement that we took at Berkeley in an environment of diskless machines. (The major results of this study are reported in Appendix A.)

Chapter 3 deals with the stochastic analysis of the traffic produced by individual workstations. The point processes produced by the workstations are packet arrivals at the Ethernet data-link layer. As such, they are different from the processes representing arrivals of messages produced by users. Consider for instance a typical 8-Kbyte user read operation: the single user request is translated by the data-link layer into a sequence of six fragments. In addition, data-link-layer packet arrivals are affected by network contention: were source and destination machines the only communicating machines the arrival patterns would be different than those extracted from the aggregate traffic. However, based on the analyses in Section 3.IX, we are confident that the errors introduced by network contention are small.

One of the most important questions is whether arrival processes (or, equivalently, interarrival time series) are stationary. Our packet-arrival processes appear, even to summary investigation, clearly nonstationary. In Section 3.V we discuss whether standard statistical techniques for removing nonstationarity can be applied to interarrival time series. We conclude that those technique, all of which amount to filtering the data so as to remove the lowest frequencies, would disrupt the data appearance and would prevent us from recognizing arrival process features and associating them to system parameters.

We recognize that the question of whether a time series is stationary is intimately related to the time scale under consideration. There are at least two time scales of interest in arrival processes that are generated by protocols under user-driven workloads: a coarse one that corresponds, for instance, to user's busy and idle intervals, and a fine one that captures protocol time constants. The key observation is that user-generated events occur infrequently when compared to protocol-generated ones. Hence, a

coarse time scale that represents macroscopic changes in the packet arrival patterns is more appropriate for characterizing user activity. If instead the focus is on protocol behavior, a finer time scale, ranging from a fraction of a millisecond to several hundred milliseconds, is more desirable. In Section 3.V we observe 1) that we do not have enough data to attempt a user characterization, and 2) that a fine time scale is more important to the queuing dynamics of packet arrival processes in interactive systems in which response time is the key performance index. Thus, we decide to restrict the analysis to fine time scales, and we remove the major nonstationary components by considering only interarrival times shorter than a fixed threshold, which we select in Section 3.VI. (Unfortunately, this truncation operation does not fully eliminate nonstationarity.)

In Chapter 3, we also discuss network utilization and histograms of interarrival times, which are related to marginal probability densities. Using the properties of our marked processes (for instance packet lengths, and source/destination addresses), we partition interarrival times into components. The partition shows that short packets produce long interarrival times and, vice versa, that long packets produce short interarrival times. We then study the autocorrelation structure of our processes, and, in the last section, examine whether processes produced by independent machines are statistically related because of network contention and concurrent access to file servers. The result is that unrelated processes are also uncorrelated.

We also carried out the spectral analysis of interarrival times, but did not include it in the dissertation. Peaks in the power spectra of intervals are expressed in terms of lags. Charnock in [13] addresses the problem of converting the lag references to times. Aside from the complication introduced by this procedure, the frequency-domain analysis revealed only, and was dominated by, the effects introduced by packet fragmentation; no other relevant phenomenon was discernible.

In Chapter 4 we define the indices of dispersion for intervals and for counts, and examine their properties. They are series of coefficients related to the autocorrelation coefficients of intervals and counts. These indices are the pivotal elements of our variability characterization. We seek models that approximate well the indices of dispersion, or, equivalently, the series of autocorrelation coefficients.

In Section 4.IV we give an example of a model (though a somewhat artificial one) in which parameters are fitted using the index of dispersion for counts. The counts produced by the models may not be similar to those of the data (they are related to them because the mean and the variance are fitted), but, by matching the correlation coefficients, the model generates sequences in which groups of counts larger than the mean and groups of counts smaller than the mean have a structure that follows the structure of similar groups in the data. The unproven assumption in this dissertation is that if data arrivals and the arrivals produced by the model are submitted to the same server of a queuing systems, they will produce comparable queuing effects.

Chapter 5 is devoted to the definition of the models of variability. These are models of interarrival times based on what we call a "generalized two-state semi-Markov model": a semi-Markov model in which intervals in each state are correlated, though intervals belonging two different states are independent. Our modeling objective is to reproduce the variability rather than to insure that the queuing behavior of the models approximates to some degree the queuing behavior of the data. We believe that there

is a close relationship between the two, but the extent to which this is true will have to be explored with further research. (This question is discussed in the Future-Work section in Chapter 6.)

The distribution of packet lengths studied in Chapter 2 indicated a sharp repartition of packet shorter than 200 bytes (short packets) on one side and those longer than 500 bytes (long packet) on the other side; there are virtually no occurrences of lengths between 200 and 500 bytes. To build the models, we first show that the lengths of short packets and the lengths of long packets constitute a two-state Markov chain: one state for short packets, the other for long packets. Second, we associate to the states of the Markov chain the probability density functions (component functions) of short and long interarrival times. Third, we derive the index of dispersion for intervals for the generalized semi-Markov model in terms of the autocovariances of the interarrival densities of the two states. A nice, simplifying feature of a semi-Markov model, which is also valid for the generalized model, is that its autocorrelation coefficients depend only on the first and second-order properties of the component density functions.

We discover that the model does not fit nonstationary data and that, to obtain a good match, we have to limit its range of applicability to lags up to 30 and to interarrival times up to about 500 ms. However, these are the intervals over which the variability of arrival processes, produced and consumed by fast, interactive queuing systems, is relevant.

This concludes the thesis outline; in the reminder of this chapter we will examine some of the published works that are related to several aspects of the dissertation.

IV. Literature Review

In this section we will briefly review the major works that are related to the dissertation. We will consider two classes of studies: network measurements and analysis studies, and packet arrival process modeling with emphasis on arrival-variability modeling.

A. Network Measurements and Their Analysis

One of the first measurement studies of local-area network traffic was performed in 1979 at Xerox PARC by Shoch and Hupp [80]. Shoch and Hupp suggested that the Ethernet under its normal workload, produced by users of Alto [90] computers accessing file servers, is lightly loaded and the host interfaces, and not the network, are the protocols' performance bottlenecks.

More recently, Boggs et al. [9] have performed measurements of artificial traffic to explore the limits of Ethernet behavior.

D.R. Cheriton and C.L Williamson [15] measured the traffic properties of an Ethernet at Stanford University that connects about 50 diskless Sun workstations running the V operating system [16]. The authors focused on the performance of the VMTP request-response protocol [14], a general-purpose protocol used for all network communication in V. They did not quantify the network utilization (which was rather low) since they were mainly concerned with the interactive characteristics of their system, such as the message transaction rate, the duration of transactions, and VMTP's retransmission behavior. The measured V systems did not have virtual memory; as a result, most of the traffic was

generated by file accesses and V's distributed-system services such as the naming service. They express the opinion that the current generation of protocols is not suitable for future high performance workstations.

E.D. Lazowska et al. [49] measured a network of diskless Sun-2s and their file servers in the context of a modeling study in 1986. Surprisingly, they found that their workstations, equipped with 2 Mbytes of physical memory, display a 4:1 file access to paging ratio. Such a finding could be explained in part by the fact that UNIX applications were smaller at that time.

In a study on network file-system caching in the Sprite operating system, M. Nelson et al. [59] concluded that even without client caching, Sun-2 machines require network bandwidths on the order of 80 Kb/s and Sun-3s on the order of 250 Kb/s; in the Berkeley study [33] we observed 1 Mb/s between Sun-3s with NFS client-side caching. The difference can be explained by the fact that they ran benchmarks consisting of compilations, sorting tasks, and text formatting jobs rather than simulating the full workload generated by a user running processes in a window environment.

P.J. Leach et al. have designed an integrated distributed system where diskless workstations and file servers communicate via a 12 Mb/s baseband token-passing ring network [50]. Preliminary performance measurements indicate that the average network utilization is low: less than 1/12 of the total network bandwidth was used. However, paging alone consumes ten percent of the network bandwidth when running artificial workload tests. Since in the described implementations the maximum packet size was 550 bytes–their network interfaces allow packets as large as 2048 bytes–higher utilization will be achieved by future, more efficient implementations.

A number of measurement studies point out the presence of high burstiness in packet data, but detailed models that incorporate this variability explicitly are not given [33,54]. Measurement studies have been recently reviewed and classified by Pawlita [65]; we refer the reader to Pawlita's comprehensive taxonomy for additional references.

Newer measurement studies [51,93] have used high-resolution clocks to obtain timing information with which it is possible to estimate statistical distributions with small bias.

Wide area traffic was measured recently by Caceres [12] and Heimlich [36]. This studies are interesting though only minimal statistical analyses are presented.

B. Arrival Process Modeling

Here we review some contributions in the area of the analysis of variability in arrival processes. The key observation is that a number of models have been developed to deal with data variability; however, most models have not been applied to computer communication traffic data or to real data at all.

In the 60's a number of statistical studies emphasized the notion that many interesting point processes cannot be described accurately as Poisson processes [6,52]. These efforts culminated in the publication of a path-breaking book by Cox and Lewis [19], whose characterization of the second-order properties of point processes is still used today in the recently revived interest in the study of data variability spurred by growing use of communication networks. For instance, Sriram and Whitt [84] propose to use the index of dispersion for intervals (IDI) for characterizing the superposition of voice and

data arrival processes, and in a recent contribution Fendick and Whitt [26] propose to use a scaled version of the variance-time curve (the index of dispersion for work) as a measurement to describe the variability of the traffic offered to a queue.

Ramaswami and Latouche [74] propose a unified stochastic model for the arrival of voice packets to a packet switch. They develop a point-process model that takes into account the statistical fluctuations of packet generation of individual calls. For this, they also resort to the use of index of dispension curves.

Heffes and Lucantoni [35] use a Markov Modulated Poisson Process (MMPP) to model the dependencies present in voice and data sources. They obtain the moments of the voice and data delay distributions as well as the queue length distribution. Numerical results for the tails of the voice packet delay distribution show the dramatic effect of traffic variability and correlations on performance.

Jain and Routhier [42] propose clustering techniques to capture dependencies in packet streams and apply these techniques to packet data collected on a ring network at MIT [24]. Trains are sequences of packets that are considered as a unit. They are identified by a single parameter: the *intercar gap*. If the interpacket time is larger that the intercar gap, then the next packet will start a new train, otherwise the next packet is part of the current train. The authors explored several variations on the theme, but seem to ignore any physical interpretation: their mean intertrain time is 23.8 seconds! They define their trains arbitrarily and give no suggestion as to how one should chose the maximum intercar gap.

Jain and Routhier's approach has been carried on to a more theoretical basis by Fontana and Guerrero, who propose a Markov model extended to include packet trains [29].

Song and Landweber propose to modify the protocol layers on host machines in order to optimize the transmission and receipt of trains of packets [83]. The proposed packet-train mechanism takes advantage of the application of similar protocol processing functions to all packets associated with a single bulk transfer. They studied the performance of their scheme with simulation and tried different train sizes, but offer no insight on how to choose the sizes of trains, nor they seem to rely on any particular train model.

Fendick et al. [25] in a theoretical study associate the burstiness of data in communication networks to various correlations in the data.

The variability characteristics of a process generated as the superposition of many sources has been studied by Albin [1], by Sriram and Whitt [84], and Heffes and Lucantoni [35].

Neuts [60, page 336] is interested in developing mathematically tractable descriptors of the physical behavior of point processes. Among such descriptors are the moments of the counting process, the peakedness functional, the power spectral density of a square wave (random telegraph wave) associated with the point process, and the caudal characteristic curve, which was introduced in [61]. The (exponential) peakedness function of a stationary point process is the ratio of the variance and the mean of the number of busy servers at an arbitrary time in an infinite server queue with exponential service times and the given point process as input.

Cruz [22] develops a calculus for obtaining bounds on delay based on input data streams that satisfy "burstiness constraints". A data stream is said to satisfy a burstiness constraint if the amount of data from the stream contained in any interval of time is less than a value that depends on the length of the intervals.

In the context of operating-system and networking research aimed at providing support for continuous media, Anderson et al. [3] define a model for traffic sources based on Cruz's thesis work. Their abstraction has the following parameters: a maximum message size of S_{max} bytes, a maximum message rate of R_{max} messages/second, and a maximum burst size of B_{max} messages. The long-term data rate is $S_{max}R_{max}$ bytes/second. In any interval of length t, the number of messages arriving at a network interface may not exceed $B_{max} + tR_{max}$. The burst parameter allows short-term violations of this rate constraint. An analogous, simple model is proposed by Ferrari and Verma [27].

Leland's study of LAN traffic [51] should be emphasized as it is similar in methodology and basic results to our measurements and analyses in Chapters 2 and 3. Leland develops the aggregate data analysis more than we do, and, although he mentions possible models for various data characteristics, he does not develop a traffic model. He associates explicitly the notion of burstiness to indices of dispersion.

2 Traffic Measurements

The Ethernet local-area network is a broadcast medium: every station must be able to monitor all communication in order to select frames destined to its own address. This makes it easy to obtain traffic measurements of single-wire Ethernets without disruption of the measured system: a single (powerful) machine can be instructed to listen to all messages without modifying or altering their arrival times. This technique was first used by Shoch and Hupp [80] and later reused numerous times [7, 15, 88]. The same passive listening technique can be easily adapted to ring networks [24], and, with a little more work, to star networks such as a single-node Datakit [54].

This is also the method we used in a previous study of the traffic of an Ethernet in the Computer Science Division of the University of California, Berkeley, in which we focused on the properties of the aggregate traffic produced by diskless workstations that communicate with their file servers [33]. Beside the specificity of the topic investigated, that study was notable for the high accuracy of its timing information and for the very small packet loss. A detailed account of the Berkeley study is reproduced in Appendix A.

A more recent work at Bellcore pushes the limits of clock accuracy and packet loss farther [93]. Its authors needed a measurement system that could detect broadcast storms: the sudden increase in network traffic caused by many hosts that answer a broadcast message.

For a number of reasons, which we will discuss in Section III, but primarily to look beyond diskless workstations and the small machines of the Berkeley environment, we decided to conduct a new study in an industrial environment with more powerful machines at Sun Microsystems, where we took traces of packet protocol information. The structure of the new measurement system is similar to the one we devised for the Berkeley study: a dedicated UNIX machine with a customized Ethernet interface driver collected packet headers and packet arrival information that were stored on disk and later moved to tapes for offline analysis. For the Sun study, we used the fastest machine available at Sun and a hardware clock with a resolution of 1 microsecond. Mainly because of bus contention, the accuracy of the timing information in the traces is lower than the resolution of the clock: the maximum errors are estimated to be smaller than 100 microseconds. This accuracy is more than sufficient for the estimation of statistical quantities and the modeling effort we intend to carry out in this study.

This chapter begins with a brief description of the main results of our Berkeley measurements, which we frequently refer to in our analyses of the data that is the topic of this study. We then describe the new measurement software, concentrating on the

hardware high-resolution clock and our modification to the Ethernet driver. We then explain the formats of the packet traces; since we had to compress the traces to be able to store on-line most of the recorded information for ease of processing, we also describe the compressed record formats. We next analyze in depth the timing errors caused by the variations in the access time to the bus shared by the CPU and the Ethernet I/O interface. Finally, we describe the aggregate traffic on the Sun Ethernet in terms of network load, packet length distribution, and packet interarrival times.

II. Berkeley Measurements

We had previously taken measurements of packet traffic on an Ethernet in the Computer Science Division of the University of California, Berkeley, in which we studied the aggregate traffic produced by more than 100 machines. Although the network environment, composed of Sun, Xerox, and VAX computers, was heterogeneous, most network traffic was generated by diskless Sun machines. For this reason, we focused on those workstations and their software. We analyzed and discussed network load, packet length distribution, source-destination traffic patterns, and interarrival time distributions, and focused on three protocol families: the Transmission Control Protocol (TCP), the Sun Network Disk protocol (ND), and the Network File System protocol (NFS) [33].

The most striking conclusion of the Berkeley study was that diskless workstations may indeed generate enough traffic to load substantially a local-area network of the current generation. The mean network utilization was low (averaging 6.5 percent over 24 hours), but bursts generated short-term peak utilization that exceeded 33 percent of the Ethernet raw bandwidth.

Our measurements showed that a workstation's traffic can be subdivided in 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 of roughly 120 Kbytes per second, or about 10 percent of the Ethernet raw bandwidth. Paging traffic, which accounted 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 standards a 4-Mbyte memory, common in the Berkeley network, is small indeed) and, possibly, by sub-optimal performance of the virtual-memory algorithms. Increasing memory sizes may decrease the level of paging traffic; however, since there is a tendency for applications to grow as more and more memory becomes available, it is likely that paging traffic will remain a noticeable component in future diskless workstation traffic. Paging traffic could be reduced, however, by improved buffering schemes; one such scheme, which calls for a global caching area for both virtual memory and the file system, has been implemented in Mach [94] and in release 4.0 of Sun UNIX.

We also observed that the interarrival time distribution of the aggregate traffic differs substantially from that of a Poisson process. We suggested that this was probably

attributable to correlations between arrivals, but did not analyze the possible correlations.

III. Sun Measurements

The Berkeley study provided us with certain insights into the nature of network traffic and experience in traffic measurements. For studying the problem of the variability of network traffic, we felt that new measurements on a different network—we chose a network at the Sun Microsystems headquarters in Mountain View, California—needed to be done. The Sun network presented some important hardware and software advances. First, the machines in the Berkeley network (mostly Sun 3/50s with 4 Mbytes of memory) were being superseded by more powerful models with larger physical memories, leading to a smaller proportion of paging traffic and a lower level of network traffic altogether. Second, the recently released SunOS 4.0 had changed the virtual memory algorithms, managing to provide private root file system partitions to client workstations without employing the ND protocol, a low-level protocol tailored to work with the Ethernet local-area network, which in the Berkeley measurements accounted for 52 percent of the traffic. Third, following the work of V. Jacobson [40], Sun had improved the performance of NFS, increasing its throughput. We anticipated that, in a SunOS 4.0 environment, the NFS protocol would be more pervasive, accounting for a larger proportion of the total traffic than the 16 percent we measured in the Berkeley network. In addition, we felt that it would be worth analyzing the effects of some features of the Sun network that were not present in the Berkeley network. For instance, at Sun Microsystems many machines have local disks, which makes those workstations independent of the network for running UNIX and its virtual memory. Moreover, the Sun computing environment is more integrated than the Berkeley environment: machines (with disks and diskless) mount file systems from a variety of file servers and from other disk-based workstations. Finally, although the clocks used in the Berkeley and Sun measurements have similar resolution (1 and 0.5 microseconds), the accuracy of the Sun clock is, as discussed below, better by at least one order of magnitude. Thus, a second study in this new environment would provide more general results and conclusions more readily adaptable to other types of network environments.

The new measurement system was similar in structure to the old one, consisting of a powerful stand-alone machine, a Sun-4/280, whose kernel had been modified to trace all packets and that had been equipped with a hardware counter that was physically assembled on a small board plugged into the machine's socket normally reserved for an optional encryption chip. The device used was an Am9513 counter chip, driven by a 2 MHz crystal, and was programmed to increment one of its five 16-bit registers every 0.5 microseconds. (We did not use the remaining four registers.) Register values thus ranged from 0 to 65535 cyclically and were used in combination with the kernel time to produce a high-resolution clock. The data path to the counter chip was 8-bit wide, making the reading of a register value to occur in two steps, and was on a bus shared by the CPU, the memory, and the Ethernet interface.

We added a routine, `microtime`, to the instrumented kernel to produce a time-stamp when called according to the following C programming language type definition:

```
typedef struct {
        struct timeval ts_tv;
        unsigned short ts_ticks;
} Timestamp;
```

The data structure `timeval`, used to represent timing information in Berkeley UNIX 4.2BSD and Sun UNIX, provides separate variables for the seconds and for the microseconds:

```
struct timeval {
        long tv_sec;
        long tv_usec;
};
```

The UNIX kernel variable `time`, which stores a value that approximates the wall-clock time, is of type `timeval` and is incremented by SunOS on a Sun-4 every 10 milliseconds. The routine `microtime`, called with a pointer of type `Timestamp` as a single argument, simply recorded the UNIX time along with the current value of the counter register from the Am9513 chip in the object addressed by the pointer argument.

The routine that read the counter chip actually had to cross the bus several times in order to obtain a reading. First, the counter value was saved in a chip's auxiliary register so that slower reading cycles would not affect its value. This required writing three 8-bit words in sequence, each in a separate bus cycle, to the chip's control and data registers. Second, the chip had to be told about which auxiliary register to make available to its data I/O port. This again required writing three separate words to the control and data registers. Finally, as mentioned above, two reading cycles were necessary to obtain a 16-bit counter register value through the 8-bit data path, bringing the total number of bus accesses to eight for each timestamp.

Derivation of packet interarrival times from the timestamps first required some processing. This was done by calculating separately the differences between two consecutive packets' UNIX times and between their counter-register values. (The differences in the counter values are taken modulo 2^{16}, by which, if x_1 and x_2 are two such values, we mean that $D = x_1 - x_2$ if $x_2 \geq x_1$ or $D = 2^{16} + x_1 - x_2$ if $x_2 < x_1$.) Then, we can use these differences to evaluate R, the number of times the counter has wrapped around. Table 2.1 shows how this is performed. The interarrival time in microseconds is then $D/2 + 2^{15}R$.

This relatively complicated timing system was chosen, over a simpler one in which the counter chip would have maintained by itself a full microsecond-accurate clock by chaining three of its 16-bit counters together, in order to reduce the time needed to read the clock and thus increase the accuracy of the measurements. Using three 16-bit counters would have required reading the counter device six times through the slow 8-bit data path to the socket into which the counter circuit was plugged; in contrast, under our system, the overhead was confined to offline processing.

We measured the time it took to obtain a timestamp using a short program (which we ran in kernel space during the testing phase of our modified kernel) containing the following code segment:

```
. . .
ptr = &hrc[0];
do {
     microtime(ptr);
} while (++ptr < &hrc[4096])
. . .
```

In this code, `hrc` and `ptr` are defined respectively as an array of size 4096 and a pointer, both of type `Timestamp`. In the 'do' loop, between calls to `microtime`, the Sun-4 C compiler generated object code with seven instructions, which the machine executed in 13 clock cycles or 0.78 microseconds. After running the code segment, we computed the differences between subsequent values contained in the elements of the array. In 97.3 percent of the cases the difference was 13 microseconds; in 2.5 percent, 14 microseconds. There were eight values of differences greater than 14 microseconds, the largest being 80 and 108 microseconds. Except for the first interarrival time of the series, which was of 21 microseconds because the code was not yet in the instruction cache, these larger intervals were caused by bus contention during accesses to the counter register. Recall that the Ethernet device with its busy DMA controller was on the same bus. Using more than one counter in the Am9513 chip would have increased the probability of bus contention delay, since more register accesses would have been necessary. Based on the measurements above, we estimate that a minimum of 12 to 13 microseconds are needed to obtain a timestamp with `microtime`.

TABLE 2.1. DIFFERENCES BETWEEN UNIX TIMES (IN MS) AND COUNTER VALUE READINGS

DIFFERENCE BETWEEN UNIX TIMES	RANGE OF DIFFERENCES BETWEEN COUNTER VALUES		NUMBER OF TIMES COUNTER HAS WRAPPED AROUND
	FROM	TO	
0	0	19999	0
10	1	39999	0
20	20001	59999	0
30	40001	14463	0, or 1 if between 0 and 14463
40	60001	34463	0, or 1 if between 0 and 34463
50	14465	54463	1
60	34465	8927	1, or 2 if between 0 and 8927
70	54465	28927	1, or 2 if between 0 and 28927
80	8929	48927	2
90	28929	3391	2, or 3 if between 0 and 3391

IV. The Ethernet Driver

We modified the Ethernet driver to extract protocol information from each Ethernet packet and to store it with a timestamp in a system's file, from where it could be dumped onto tapes. Although we cannot publish the code of our modified driver, since it is owned by Sun Microsystems, we will describe here some details of its implementation.

The Sun-4 used for the measurements had two network interfaces, one on the CPU board and one available as a card on the VME bus. We disabled the VME interface and used the on-board Ethernet controller as it accesses a bus with larger bandwidth. The on-board Ethernet controller, based on the Intel 82586 chip [37], was programmed to receive, without CPU supervision, all Ethernet frames that were transmitted. The 82586 coprocessor moves each frame it receives to one of a large number of buffers in kernel

memory using DMA. It uses for temporary storage a small 16-byte first-in-first-out queue to prevent a temporarily busy bus from disrupting data reception. This organization is superior to one in which the controller buffers a frame entirely in its internal memory before moving it to the computer memory since it reduces latency in data reception. Upon being informed by an interrupt of the successful transfer, the CPU extracted the Ethernet packet header and the portion of the data containing protocol information, and time-stamped the record with the time information described above.

The driver employed a double buffering technique: it kept two 256-Kbyte buffers in kernel memory in which it stored the packet information it was collecting. When the first buffer was full, it switched to the second one, concurrently awakening a sleeping kernel process, which ran only when there was no network interrupt pending or being served, for the asynchronous transfer of the data in the first buffer to a trace file on disk. This procedure alternated between the two buffers.

When a trace file grew to 100 Mbytes—a size we had chosen because it fit in one standard tape at 6250 bpi, and was the size of the three disk partitions we had dedicated to the measurements—the kernel driver automatically switched to another file. Dumping the data onto tape presented no problem since it took only a few minutes, while filling up one file took from about 50 to 80 minutes, depending on the time of day.

An important goal of our measurements was to reduce packet loss to a minimum. There are two major causes for packet losses: overruns of the small FIFO in the controller caused by a busy bus, and lack of buffers in main memory to receive the data. We programmed the controller to initiate data transfer to the bus when the data in the FIFO queue had reached 12 bytes and allocated 250 buffers in kernel memory for the frames. The Intel Ethernet controller monitored the bus transfers and kept counts of the frames discarded due to bus overruns and a shortage of buffer space. The frames lost were typically a few units over a day, when several million frames were received. Apparently, the speed of the CPU, the bandwidth of the on-board bus to which the Ethernet controller was connected, and the memory allocated to receive frames were sufficient.

V. Packet Traces

The packet traces are composed of variable-size records, each of which is preceded by a standard header. The format of the header is shown below:

```
struct emt_record {
        unsigned long dst;
        unsigned long src;
        unsigned short type;
        unsigned short length;
        Timestamp mtime;
        unsigned short bcount;
        long sequence;
};
```

The first two fields of this structure are used to store the least significant four bytes of the six-byte Ethernet source and destination addresses. In general, these four bytes may not identify uniquely each machine. However, since the most significant three bytes of an Ethernet address identify the interface maker, they are likely to be redundant in a

network of machines of the same type. We have verified, by comparing directly each machine's address against each other, that the lower four bytes of the addresses do identify our workstations uniquely.

The `type` field is also taken from the Ethernet frame and identifies the higher-level protocol in the frame's data portion, the length of which is stored in the `length` field of the record above. The fifth field, `mtime`, is used to store the timestamp; the `bcount` field, which indicates how many bytes of the original frame follow the trace header, is a pointer to the next packet's header; and the sequence number, which we included to discover how many records would be lost in case of recording errors in the secondary-level magnetic storage media, is incremented by one with every received frame.

The decision of how many bytes to save for each packet was made by scanning the type field of the Ethernet header, making assumptions about higher level protocols, and trading off space for time. For instance, if an IP packet was of type UDP, we assumed it was transporting NFS data and saved enough bytes to record the largest possible NFS header. Careful, complete protocol demultiplexing would have resulted in longer execution time during trace collection, not justifying the space saved.

VI. Data Compression

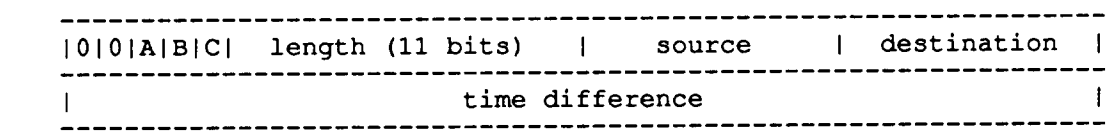
Typically, the amount of information in the traces can be as large as 100 Mbytes per hour of network traffic. The analyses and results shown in this study required reading and processing the packet traces countless times. Without keeping the information on secondary random-access storage our work would have not been possible. Given the severe space limitations of our computing environment, it was necessary to compress the traces in order to store several hours of network activity on line. The formats that we chose for the compressed data reflect the requirements of most, though not all, of our analyses: in most cases we needed only information about the source and destination addresses, the packet length, and the arrival times.

The format of the records of the compressed data is variable: there are four different formats, which are identified by the first two bits of each record. The source and destination addresses required eight bits each. Even though a single Ethernet can have more than 256 interfaces (each one with its own unique address), commonly Ethernets are class-B IP subnets, for which the last byte of the IP address is reserved for the hosts (or, more precisely, for the host interfaces). On the Sun network we monitored, all machines have (distinct) IP addresses. Packet length information (a number between 46 and 1500), required 11 bits. Finally, we encoded the timing information in the form of an interarrival time: the interval of time between a packet and the packet preceding it.

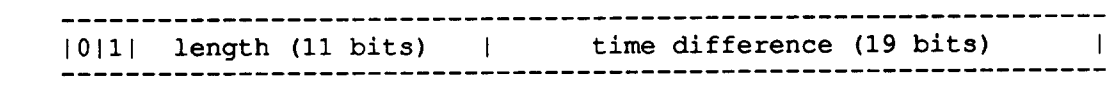
While converting the packet arrival time information, we also changed the original times, which we recorded in the traces as the times when packets were received, to the times when the transmission on the Ethernet first started. This was done by subtracting from each recorded packet arrival time the packet transmission time, which is proportional to the packet length.

The first type of format, identified by two zeros in the most significant bits (the *format type bits*), encodes the interarrival time in a total of 35 bits—32 bits in the second four bytes of the record, and three bits, indicated by A, B, and C, just after the two format

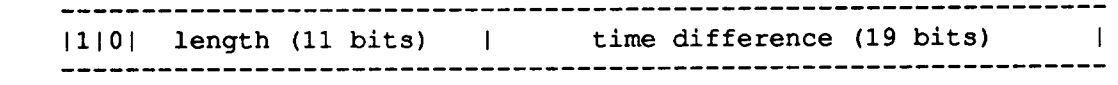
type bits. This allows us to represent time differences of about nine and a half hours, more than enough since workstations are never silent for this long a time. (In this and the following pictorial representation of formats, we show 32 bits on each line.)



If a packet followed the exact same path of the packet that immediately preceded it, i.e., moved from the same source machine to the same destination, there is no need to store the source and destination fields in the record. Instead, we use a second type of format, identified by "01" in the format type bits, which instruct trace-analysis programs to produce a packet with the same source and destination addresses as the previous packet's:

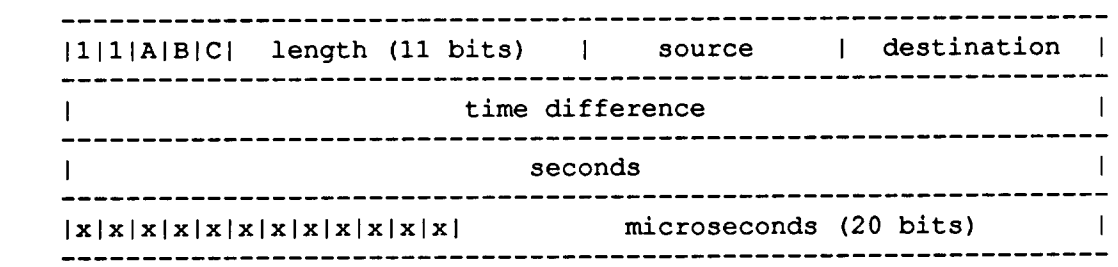


If a packet moved between the same pair of machines as the preceding one, but in the opposite direction, we employ a third, "10", format:



In these two formats, we reserved for the time difference 19 bits, capable of representing interarrival times of up to about half a second. This should be sufficient for most cases, though, especially in traces of traffic of single workstations, interarrival times may be larger. When this occurs, we revert to the first format for the record, even if the address fields of the two packets are the same.

Using interarrival times helps to save considerable space, but has the disadvantage of requiring sequential reads of all the previous records in order to obtain the arrival time of a specific packet. To obviate this problem, we chose a fourth format based on the first format but expanded to include the arrival time, expressed in the standard Berkeley UNIX 4.2BSD convention in which the time is represented in seconds and microseconds since midnight, January 1, 1970. (The x's indicate unused bits.)



A record with this format was inserted at the beginning of each compressed trace file (we kept on line compressed versions of the original traces divided into files representing approximately one hour each). To determine the arrival time of a given packet, we only

need to refer back to the first record of the file in which the packet's record is located, and not the the beginning of the trace. In addition, we used this format to begin each trace containing a subset of the global traffic that we extracted from the aggregate traces.

It is important to notice that we use this type of data encoding for both the full traces that we have collected on the Sun network and the traffic of individual workstations (or any subset of the aggregate traffic) that we extract from the general traces for additional, more specialized processing (such as estimating the interarrival time distribution of a particular machine). Thus, we only needed to design a single set of programs to read and manipulate all the various possible traces.

With these data-compression techniques, we used only about one third of the storage space that would have been needed if we had represented each packet with time, source, destination, and length fields as in the original traces. With respect to the original traces, we need only one sixteenth of the storage space, though, of course, we lose information since in the compressed traces we do not represent protocol data.

VII. Timing Errors

Timestamping may not always accurately record the arrival time of a packet. Although the counter-based clock has high resolution, latencies in interrupt response time and lags in accessing the bus can lead to delays in the recording of timestamps. For instance, if a packet (the receipt of which triggers the posting of an interrupt to the CPU) is followed by the smallest Ethernet packet (64 bytes), and the two packets are separated by the minimum interpacket time permitted (9.6 microseconds), the CPU has only 67.2 microseconds to process the first packet before the next interrupt occurs (51.2 microseconds for the 46 bytes of data and the 18 bytes of header and checksum, 9.6 microseconds for the interframe time, and 6.4 microseconds for the Ethernet preamble, which always precedes the first bit of a frame and is used to synchronize the clock of the receiver with the clock of the transmitter). Since 13 microseconds are necessary to record a timestamp, should the CPU take more than 54.2 microseconds to process the first packet, the second packet would be timestamped later than it should be. (This is so because UNIX serves only one network interrupt at a time.) Notice also that in this situation, since the timestamp of the second arrival time is delayed, the interarrival time between the second packet and the third packet may be shorter than the real interarrival time. Because the driver only needs to perform few operations per received frame, this hypothetical situation should occur infrequently.

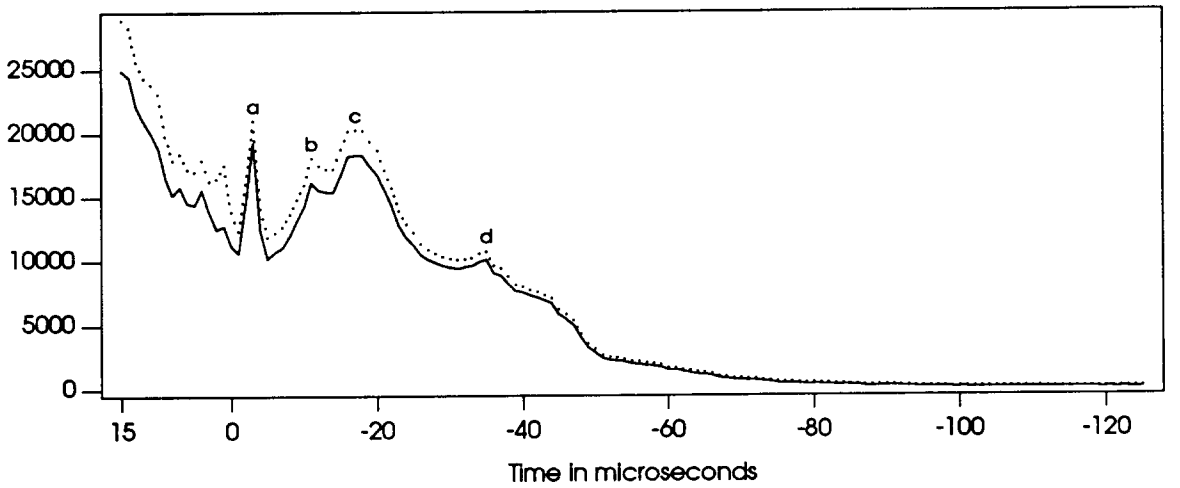
During testing, we found the timestamps of some packets to be such that the transmission of each of these packets should have begun before the arrival of the previous packet—an impossible situation. If we define the *interpacket time* to be the time between the first bit of a packet and the last bit of the previous packet, this situation is characterized by negative interpacket times. Clearly, negative interpacket times must have been artificially created by measurement errors. In this section we will address two issues: bounding the size of the errors and determining the mechanisms that delay the timestamps.

To determine the size of the measurement errors, one needs to compare real values with measured values. Although no actual values are available, we can reasonably hypothesize what the actual interpacket times were for a class of packets pairs that generated errors. Specifically, these are packet pairs with unrelated source and destination addresses. Assuming that communications between unrelated pairs of machines are statistically independent—an assumption that we will discuss in detail in the next chapter—it is intuitive that, given any marginal probability distribution of the interarrival times of packets between two stations, the probability that a ready-to-transmit station will find the network busy is directly (though perhaps not linearly) related to the length of the packet in transmission. The majority of negative interpacket time errors were caused by pairs of packets in which the first packet had the maximum size of 1500 data bytes. Thus, it is reasonable to assume that each second packet was queued up, waiting for the large-sized packet in front of it to reach its destination. Since an Ethernet interface ready to transmit samples the medium continuously and attempts transmission only after the wire has been idle for at least 9.6 microseconds, a behavior that has been termed 1-persistent [48], a packet that is queued at one interface should generate, in the absence of collisions, an interpacket time of 16 microseconds (9.6 microseconds of the minimum interframe spacing plus 6.4 microseconds necessary to transmit the frame's preamble). Thus, interpacket times of less than 16 microseconds for two packets in our traces with unrelated source and destination addresses should in fact be exactly 16 microseconds. (Some authors define two packets with minimum interpacket time as *back-to-back* packets.)

In Figure 2.1 we show, represented by a solid line, the number of interpacket times less than 16 microseconds including negative times, for these pairs of unrelated packets belonging to a 10-hour trace, which we analyze in the remainder of this chapter and use in the rest of the thesis. About 9.8 percent of all packets in our traces—a sizable number—were affected by this type of errors. We also show, as a dotted curve, the total number of errors for all packets, including those generated by pairs of packets between the same two of machines. This curve follows the shape of the unrelated-packet curve, suggesting that packets between the same two machines may also be caused by measurement errors involving back-to-back packets. In fact, in SunOS 4.0, back-to-back packets can occur between two machines: as reported in [40], improvements in the Ethernet throughput were achieved mainly by reducing the interpacket gap during packet fragmentation.

We observe that the large majority of errors in Figure 2.1 is between 15 and –85 microseconds (some 99.2 percent of the timing errors are within this interval). In the discussion that follows, we will also show that there some back-to-back packet pairs for which the measured interpacket times are larger than 16 microseconds. These too are caused by measurement errors; however, they are many fewer than the errors in Figure 2.1 and do not extend towards positive times as much as those extend towards negative times. We can therefore conclude that in our traces with probability 0.99 the absolute value of the measurement errors is smaller than 100 microseconds. In addition, in building the compressed traces, which we have used for all timing analyses, we have adjusted the measured arrival times of all the packet involved in Figure 2.1 so that their interpacket times become 16 microseconds. In doing so, in all probability we have reduced the

FIGURE 2.1. ERRORS IN THE TIMING MEASUREMENTS



value of the average error. Based on these estimates, we can state that the accuracy of these measurements is fully adequate for all analyses we perform in the following chapters.

The conclusion above is predicated upon the assumption that first packets are long; as Figure 2.2 shows, 72 percent in fact are longer than 500 bytes. In Figure 2.2 we plot the packet length distribution of the first packets in the pairs that generated the interpacket time errors; in Figure 2.3, the distribution of the length of the second packets of those pairs. (These data are from the 10-hour trace.) Notice that the median of the distribution of the first packets is 1500 bytes. This, as observed above, represents a case of length biasing: since it is more likely that the minimum interpacket time occurs after a long packet, a timing error of this type is more probable after a long packet.

FIGURE 2.2. PERCENTAGE OF PACKETS VS. PACKET LENGTH
FIRST PACKETS IN PAIRS

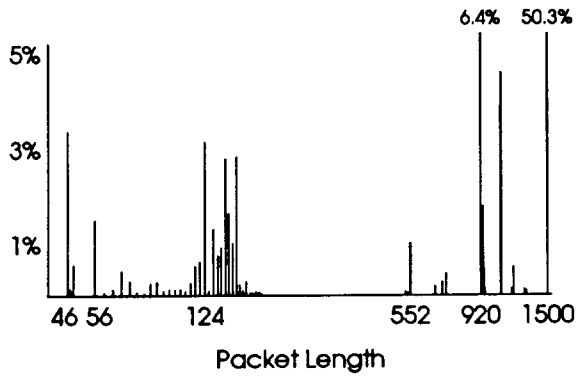
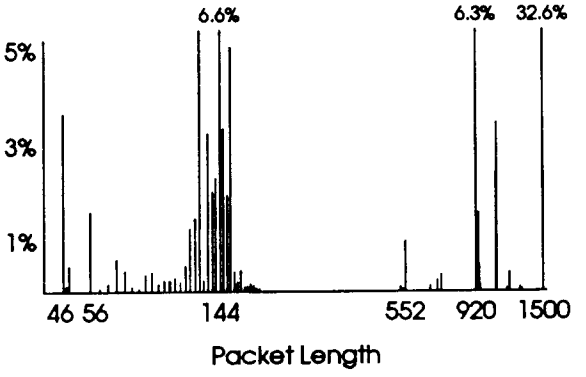


FIGURE 2.3. PERCENTAGE OF PACKETS VS. PACKET LENGTH
SECOND PACKETS IN PAIRS



These two figures should be compared to Figure 2.15, which shows the packet length distribution of all the packets in the 10-hour trace. Unlike Figure 2.2, Figure 2.3 is

quite similar to Figure 2.15, suggesting that the lengths of the second packets may be independent of the lengths of the preceding interpacket time. In order to explore the hypothesis of independence, we computed the distribution of the lengths of consecutive interpacket-time errors. Errors that occur in sequence one after another generate dependencies in the packet lengths. To see this, consider, for instance, two errors in a row. Three consecutive packets in the aggregate traces are involved, the second and the third of which will both be considered among those in Figure 2.3. Since the lengths of packets exchanged between two machines are autocorrelated, as we will see in Chapter 5, and since there is a high probability that subsequent packets in the aggregate trace are between the same stations, there may be dependencies between two subsequent timing errors and the packet lengths of the associated packets.

We found that 84.9 percent of the errors are single, that is, the interpacket times that precede them and those that follow them are not in error. In 13.2 percent of the cases there are two errors in a row, in 1.6 percent three errors, and in 0.2 percent four errors. Since only a fraction of the 15.1 percent of two or more consecutive errors are caused by packets exchanged between the same pairs of machines (the probability we mentioned in the previous paragraph is less than 1), the global effect of consecutive errors on Figure 2.3 is negligible. Thus, we conclude that the packets of Figure 2.3 represent a random sampling of all the packets in the traces.

Next, we would like to identify and understand the mechanisms by which these short interpacket times were generated. The presence of peaks in Figure 2.1, some of which we have labeled, indicates that errors do not occur randomly, but are produced by some recurring conditions. We noticed that packet pairs in which the second packet was 46 data bytes generated errors of value –3 microseconds (i.e., peak *a*) very frequently, suggesting that perhaps the length of the second packet was a primary factor responsible for the peaks.

To determine whether or not errors were related to the length of the second packet in a packet pair, we plotted the number of errors occurring for various second packet lengths. Figures 2.4 and 2.5 show the results for two of the most common, shortest second packet lengths, 46 and 56 data bytes. (These two data lengths were chosen because they occur frequently, and hence allow the error estimation to be statistically meaningful.) These graphs prove that in Figure 2.1 the peak at –3 microseconds, labeled *a*, is caused by the errors associated with 46-byte second packets and that the glitch at –11 microseconds, labeled *b*, is instead caused by 56-byte packets.

We will explain the shapes of Figures 2.4 and 2.5 with the help of the timing diagram of Figure 2.6. In this diagram, which illustrates the transmission of a 46-byte frame (and a 56-byte one) separated from the previous frame by the minimum interframe time, we show a possible scenario for the situation that leads to the errors in Figures 2.4 and 2.5. The unit of the time axis is one microsecond, and the tick marks are spaced two microseconds apart. The time origin is arbitrarily set at the time when the last bit of the first frame (of the two frames that will generate the interpacket time error) arrives on the network. Each event is represented by a labeled arrow and a time value. Events above the time axis are Ethernet events; those below the axis are events—either measurements, or quantities derived from measurements, or machine events—that occur in the measurement system.

FIGURE 2.4. ERRORS IN THE TIMING MEASUREMENTS
SECOND PACKETS OF 46 Data Bytes

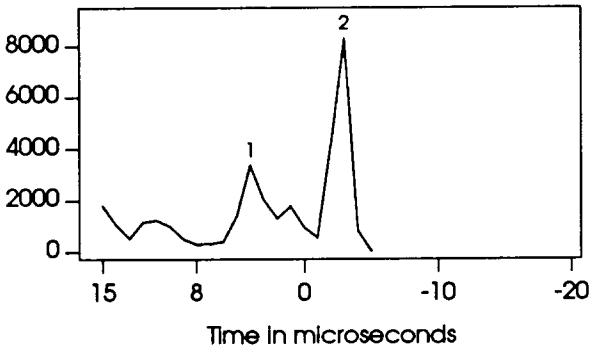


FIGURE 2.5. ERRORS IN THE TIMING MEASUREMENTS
SECOND PACKETS OF 56 DATA BYTES

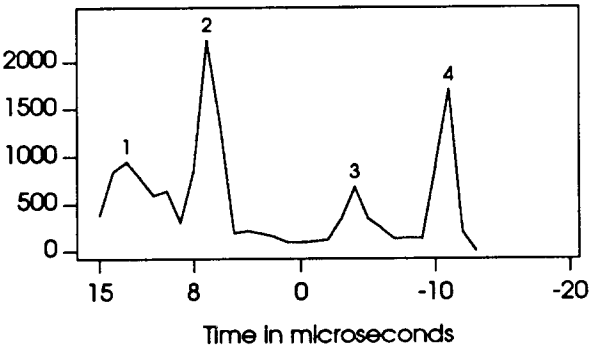
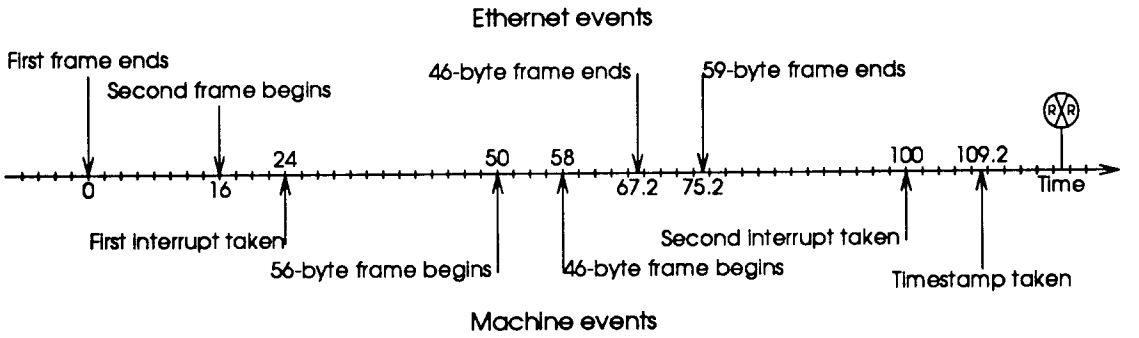


FIGURE 2.6. TIMING DIAGRAM OF INTERPACKET TIME ERRORS
SECOND PACKETS OF 46 AND 56 BYTES



The first frame is fully received at 0 microseconds, and, assuming that the CPU is idle—a reasonable assumption since most of the first packets are 1500 bytes implying that the previous interrupt occurred at a time smaller than −1230 microseconds in our reference system—the corresponding interrupt is signaled shortly thereafter. We indicated that the interrupt occurs at 24 microseconds because of the interrupt latency time—the several dozens of instructions that UNIX executes in order to switch context to the interrupt-handling routine and then to the device-driver entry point—and because the Ethernet controller chip has to update various data structures and pointers in the computer's memory, which is done through DMA, before posting the interrupt. (Most likely this is an optimistic estimate; however, whether or not the interrupt actually occurs a little later is not important for our purpose of deriving qualitative conclusions from these ballpark estimates.) The first bit of the second Ethernet frame is transmitted at 16 microseconds, after the minimum interframe spacing of 9.6 microseconds and the transmission of the 6.4-microsecond Ethernet preamble. Therefore, the timestamp routine, called soon after the interrupt handler gives control to the Ethernet driver, is executing while the second frame is being received by the Ethernet interface device.

To timestamp the first packet requires, as observed above, crossing the bus, which is now busy moving the data of the second frame from the interface device to the buffers

in main memory. Although the counter value is copied to an auxiliary register early in the timestamping process, three bus accesses are required before this can happen, increasing the value of the timestamp. Moreover, all memory accesses for the Ethernet driver code also require bus cycles, causing additional delay before the timestamp can be taken. Although we do not have measurements of bus utilization during our Ethernet packet tracing and cannot provide direct data to assess the value of these timestamp delays, we can cite measurements on a Sun-3/50 [92] that showed that the Ethernet-interface DMA engine loaded the bus up to 70 percent during the transfer of Ethernet frames into memory. The result of this study supports our interpretation of the delays.

If the frame's data length is 46 bytes the last bit of the frame is received at 67.2 microseconds; if the length is 56 bytes, the last bit is received at 75.2 microseconds. We will assume that the Ethernet will become idle after receiving the second frame; again, this is a reasonable assumption, as we have seen that the large majority of errors are single. Under this assumption, the timestamps of the second packets are not delayed.

Now assume that servicing the interrupt of the first packet is completed by the time the 46-byte frame is received. Then, all things being equal, the interrupts associated with a 46-byte frame and with a 56-byte one will be separated by an interval of time equal to the interval between the two frames: 8 microseconds. Similarly, between the two associated timestamps there will be 8 microseconds. Finally, it is plain that the starting times of the two frames, reconstructed from the timestamps, will coincide. But if this were true, since the 0's on each x-axis in Figures 2.4 and 2.5 coincide with the beginning of each frame in Figure 2.6, Figures 2.4 and 2.5 would have to show peaks at the same times, which is not the case. Thus, we conclude that the servicing of the first frame's interrupt extends beyond the time when the 56-byte frame is fully received and, hence, that the interrupts generated by both 46-byte and 56-byte frames must occur at the same time.

We have illustrated this situation graphically in Figure 2.6, which also shows the reconstructed times for the beginning of the transmission of the two frames at 50 and 58 microseconds. Consider peak 1, at 4 microseconds before the beginning of the second frame of 46-bytes in Figure 2.4. In Figure 2.6, this point would be at 54 microseconds, that is 4 microseconds after the beginning of the 56-byte frame. Thus, we recognize that peak 1 in Figure 2.4 is associated with peak 3 in Figure 2.5. The association is caused by the same amount of delay in the timestamp of the first frame. Through a similar line of reasoning, we can show that peak 2 in Figure 2.4 and peak 4 in Figure 2.5 are also associated. This gives us evidence that the timestamping of the first frame is delayed by discrete, fixed intervals depending on how many times the three bus accesses are delayed during bus arbitration.

Peaks 1 and 2 in Figure 2.5 occur when the timestamps of the first frames are delayed by smaller amounts. In these circumstances, the interpacket times of 46-byte frames are larger than the 16-microsecond minimum and, though not shown in Figure 2.4, still must be considered errors. We have verified that these errors are present but, at least in this particular case, they are fewer. However, in general, the error intervals of figures such as Figure 2.4 and 2.5 need to be extended to include valid-interpacket-time errors.

Figures 2.7 and 2.8, analogous to Figure 2.4 and 2.5, refer to errors generated by packet pairs in which the second packets are 124 and 144 bytes, respectively. We can see that the peak labeled 2 in the 124-byte figure produces a small bump labeled *d* at –35 microseconds in Figure 2.1.

FIGURE 2.7. ERRORS IN THE TIMING MEASUREMENTS SECOND PACKETS OF 124 DATA BYTES

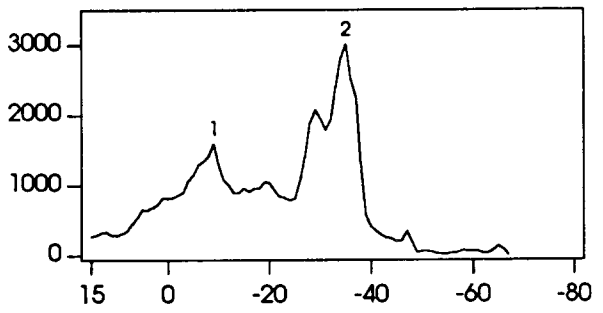
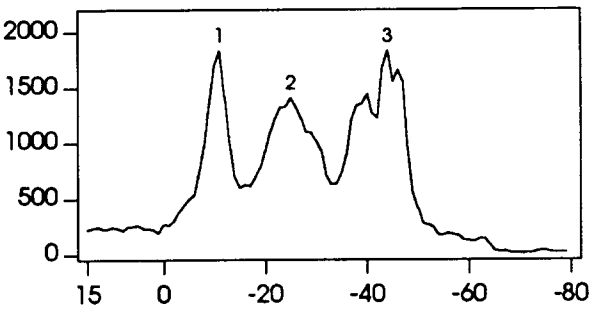


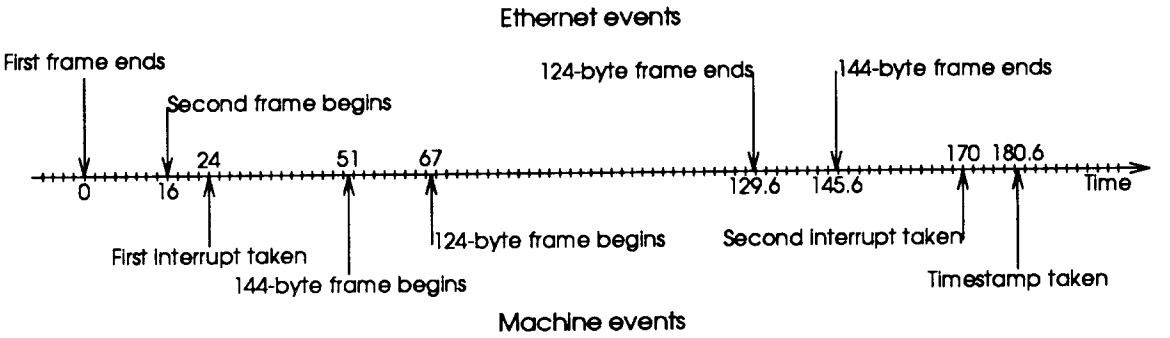
FIGURE 2.8. ERRORS IN THE TIMING MEASUREMENTS SECOND PACKETS OF 144 DATA BYTES



Here, it is more difficult to interpret the peaks. In Figure 2.9, in which we show the timing diagram of the transmission of a 124-byte and a 144-byte packet, we have again indicated that the timestamps for the two packets occur at the same time. These are NFS packets of which we saved 124 bytes in the traces (a number of bytes that is guaranteed to contain the longest NFS header). It is important to observe that since the SPARC is a RISC architecture and does not have memory-to-memory instructions, the data from the network buffer must first be moved to a CPU register and then from the register to the trace buffer, crossing the bus twice in short sequence. Thus, for these packets the interrupt servicing time is much longer than for the short 46 or 56-byte TCP packets. Because of these facts, the beginnings of transmission of the two packets are 16 microseconds apart, which can be recognized as the difference between peak 1 in Figure 2.7 and peak 2 in Figure 2.8. However, this does not appear to be the case if we compare peak 2 in Figure 2.7 at –35 microseconds after the beginning of the 124-byte packet, and peak 3 in Figure 2.8 at –44 microseconds after the beginning of the 144-byte packet. These two peaks are only 9 microseconds apart. The double-pointed peaks in both figures suggest that in some instances the interrupts for the 124-byte packets and for the 144-byte packets occur after equally long delays, but in other cases after delays of different lengths.

Figures 2.10 and 2.11 show the errors associated with pairs of packets in which the second packets are respectively 920 and 1500 bytes. The two graphs are virtually identical, indicating that the time when interrupts are received depends on packet lengths. They are also very similar to graphs (not shown) for packet lengths of 1064 and 1072 bytes. The mechanisms generating the errors are the same as those we have discussed previously. Here, again, the timestamp associated with the first arrival is delayed, while the second timestamp, taken when the Ethernet and consequently the CPU bus are idle, is not. Whereas in the case of short packets one could argue that the delays are bounded by the packet transmission times, since after the completion of a packet's

FIGURE 2.9. TIMING DIAGRAM OF INTERPACKET TIME ERRORS
SECOND PACKETS OF 124 AND 144 BYTES



transmission the bus becomes idle, allowing the timestamp process to proceed without additional delay, in the case of very long packets arbitrarily long delays could occur. But this is not the case, since the great majority of the delays in both figures are smaller than 40 microseconds, indicating that the maximum amount of delay is bounded. Our model of delays attributable to bus contention capture this phenomenon: bus contention is resolved in a finite time since bus arbitration does not allow starvation. Thus, the timestamp of the first packet when the second packet is long occurs within a short time. The long tails may be due to other factors such as the effects of collisions, which increase interpacket times by some 50 microseconds during which the channel is jammed. The initial decreasing slope of the curves indicates that the second timestamp may also be delayed. The explanation is simple: since the second packet is long, there is a high probability that a third packet becomes queued during its transmission.

FIGURE 2.10. ERRORS IN THE TIMING MEASUREMENTS
SECOND PACKETS OF 920 Data Bytes

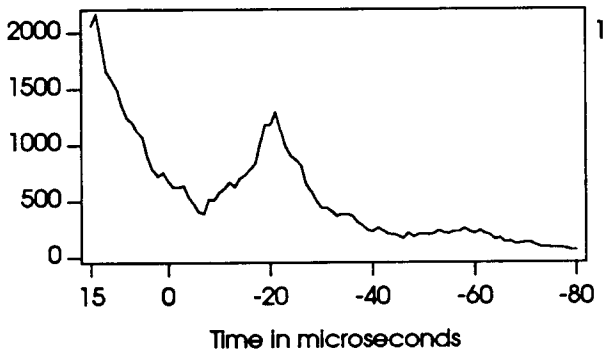
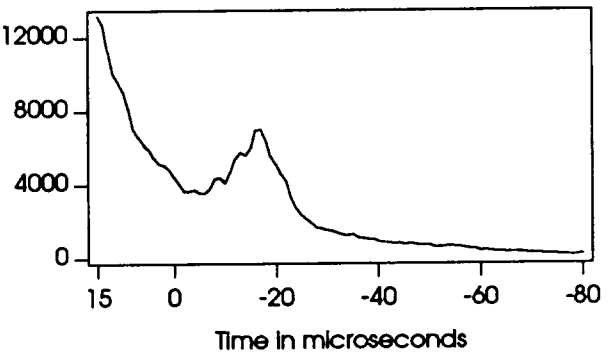


FIGURE 2.11. ERRORS IN THE TIMING MEASUREMENTS
SECOND PACKETS OF 1500 DATA BYTES



Many questions are still not fully resolved. For instance, without additional measurements, it is difficult to assess the precision of the numbers we have used in our argumentation about the timing diagrams of Figures 2.6 and 2.9. Nevertheless, we think we have presented evidence of the types of mechanisms that are behind the interpacket time errors. In summary, we found that the errors are caused by readings of counter values delayed on the single bus used in the CPU board to access both the counter chip and

the Ethernet interface device. The magnitude of the errors depends on how many of the counter accesses necessary to obtain a timestamp are delayed. We estimated that, with probability 0.99, the errors of our clock–counter combination are, in absolute value, smaller than 100 microseconds.

VIII. The Sun Engineering Network

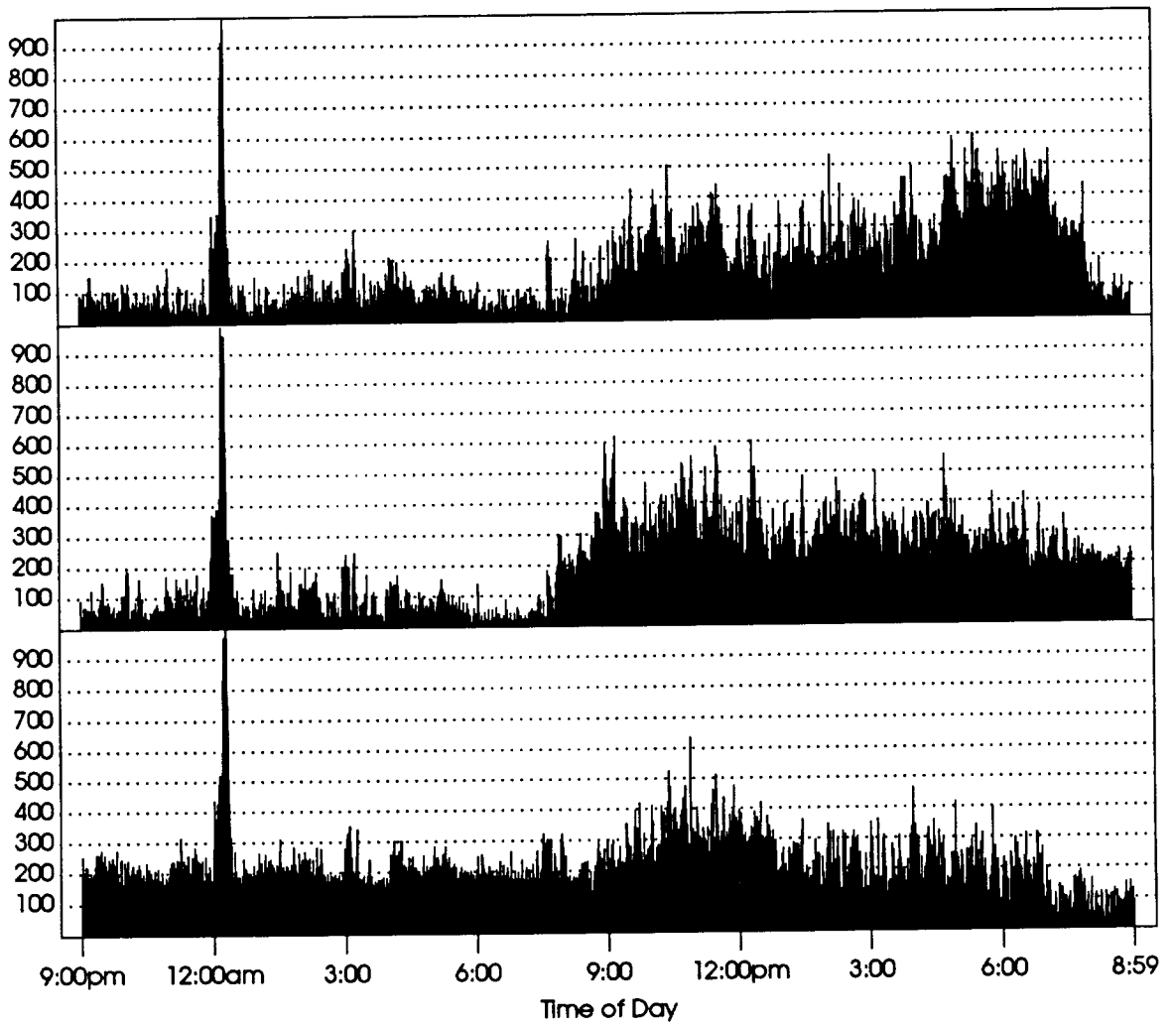
In order to get a picture of the workload of the Sun Ethernet, we monitored the load over 24-hours for several days during the testing phase of the measurement system. Figure 2.12 shows the packet rate over 72 consecutive hours. Notice that the activity, unlike the activity in a university environment shown in Figure A.1, is concentrated during office hours. During the night hours the average packet rate falls to roughly one packet per second per workstation. This background activity consists primarily of diskless workstations periodically flushing to disk their dirty virtual-memory pages and modified pages in the file cache, and of programs, left running by users who are no longer in their offices, that monitor the load on other workstations or file servers by probing them periodically. In addition, UNIX has a mechanism to run "housekeeping" programs, such as the commands to clean up directories that contain temporary files, at midnight every day. Since the clocks of all machines are approximately synchronized, this generates the peak load seen shortly after midnight. The higher load during the night in the last 24 hours in Figure 2.12 was caused by a set of benchmark programs that were left running until morning.

Because the objective of our research was to analyze how the user workload determines the traffic behavior and to build models that capture the variability of the arrival processes, we decided to trace only working hours and not the night periods. On the basis of a preliminary study of which the data in Figure 2.12 is part, we selected a typical day for our analysis. In the remainder of this chapter and in the rest of the thesis, the data presented and discussed will be based on traces that we collected on April 10, 1989 during a period starting at 8:15 am and lasting 10 hours and 20 minutes. The full trace amounted to about 850 Mbytes of information. We kept on line and loaded segments of the original data when necessary for more specialized analysis. No packet loss was experienced in this particular trace, which is the same data set whose timing errors were discussed in the previous section; our conclusion there was that the errors in the recorded arrival times were smaller than 100 microseconds. In the remainder of this chapter we will discuss some general statistics of the aggregate traffic in the Sun network; these results should be compared to the results in Appendix A, which describes the aggregate traffic of the Berkeley network.

The Sun Engineering network in which the data were collected consisted of about 130 machines and six major file servers. (We will differentiate between client workstations, which are single-user machines, and file servers, which have no users of their own, but provide file services to client workstations.) Table 2.2 provides some basic information on some of the workstations of the Sun network. (To ensure security of the Sun network and to protect the privacy of individual users, we have not used the actual machine names; the names used throughout the thesis are fabricated ones.)

Figure 2.13 displays the network load during the 10 hours of our data set. Each vertical line shows the percent mean network utilization over a 30-second interval, that is, the fraction of the 30-second period during which machines successfully transmitted Ethernet

FIGURE 2.12. SUN ENGINEERING NETWORK PACKET RATE (PACKETS/SEC)
THREE CONSECUTIVE WEEKDAYS (FROM TOP TO BOTTOM)



frames. (In computing the utilization, we have included the time spent in transmitting the checksums and Ethernet-frame preambles, as well as packet headers and data bytes.) The highest peaks reach 40 percent, indicating that for several seconds the Ethernet load was nearly half of the network's total capacity.

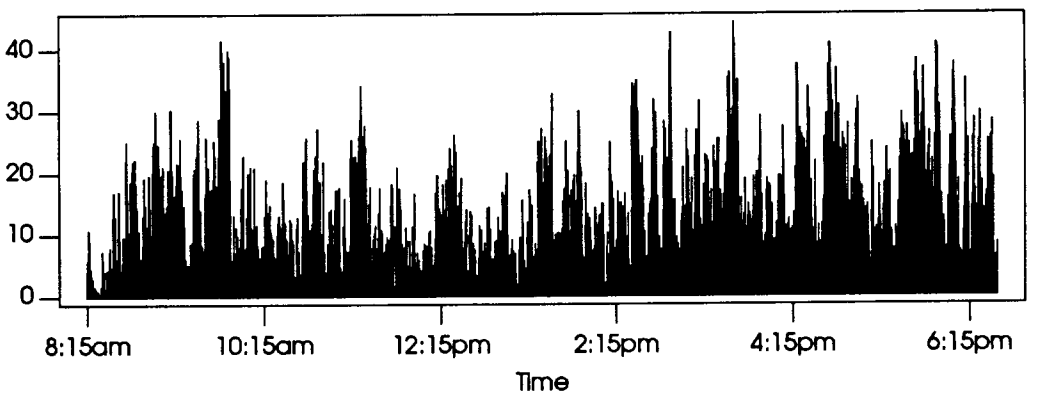
At these levels one would expect a high number of collisions. We have not measured collisions and are not concerned here with the question of estimating the collision rate except for the way in which it affects workstations' network access and the correlations it induces in the traffic among simultaneously communicating workstations. We will deal with these issues in Chapter 3, when studying the sources of correlation in the traffic.

The average load over the 10 hours in Figure 2.13 is 14.6 percent. Much of it involved the six major file servers. As previously noted, at Sun Microsystems it is customary for workstation users to mount several file systems from many different file servers onto their diskless-machine file space. Furthermore, many clients have local disks that in some

TABLE 2.2. WORKSTATIONS WHOSE TRAFFIC IS USED IN THE EXAMPLES

WORKSTATION	TYPE	MEMORY SIZE	OS VERSION	LOCAL DISKS
station 1	Sun-3/160 server	20 Mbytes	Sun-UNIX 3.5	2 disks
station 2	Sun-3/50 client	4 Mbytes	SunOS 4.1	diskless
station 3	Sun-3/50 client	4 Mbytes	SunOS 4.1	diskless
station 4	Sun-4/110 client	8 Mbytes	SunOS 4.0	2 disks
station 5	Sun-3/280 server	8 Mbytes	SunOS 4.0	4 disks
station 6	Sun-4/260 client	16 Mbytes	SunOS 4.0	1 disk
station 7	Sun-4/280 server	32 Mbytes	SunOS 4.0	1 disk
station 8	Sun-4/280 client	32 Mbytes	SunOS 4.0	3 disks
station 9	Sun-4/280 server	32 Mbytes	SunOS 4.0	2 disks
station 10	Sun-3/75 client	8 Mbytes	SunOS 4.1	diskless
station 11	Sun-4/260 client	32 Mbytes	SunOS 4.0	1 disk
station 12	Sun-3/50 client	4 Mbytes	SunOS 4.1	diskless
station 13	Sun-3/50 client	4 Mbytes	SunOS 4.1	diskless
station 14	Sun-3/50 client	8 Mbytes	SunOS 4.0	diskless
station 15	Sun-4/110 client	8 Mbytes	SunOS 4.0	1 disk
station 16	Sun-4/260 client	16 Mbytes	SunOS 4.0	1 disk
station 17	Sun-3/50 client	4 Mbytes	SunOS 4.0	diskless
station 18	Sun-3/110 client	16 Mbytes	SunOS 4.0	1 disk
station 19	Sun-3/280 client	16 Mbytes	SunOS 4.0	2 disks
station 20	Sun-4/110 client	8 Mbytes	SunOS 4.0	1 disk
station 21	Sun-3/280 server	16 Mbytes	SunOS 4.1	2 disks
station 22	Sun-3/50 client	4 Mbytes	SunOS 4.0	diskless
station 23	Sun-4/280 server	32 Mbytes	SunOS 4.0	2 disks
station 24	Sun-4/110 client	8 Mbytes	SunOS 4.0	diskless
station 25	Sun-3/50 client	4 Mbytes	SunOS 4.0	diskless
station 26	Sun-4/260 client	16 Mbytes	SunOS 4.0	1 disk
station 27	Sun-3/260 client	16 Mbytes	SunOS 4.0	1 disk
station 28	Sun-3/280 server	16 Mbytes	SunOS 4.0	2 disks

FIGURE 2.13. NETWORK UTILIZATION (PERCENTAGES)



cases are used to export files to other machines, and thus, at Sun, even personal workstations may function as file servers for other workstations. As a result, the patterns of file access of individual users are quite complicated.

On the average, no single client workstation sent or received much more than 2 percent of the total number of packets over the entire 10-hour trace. File servers generated a higher proportion of the traffic because they interact with many clients simultaneously. This is shown in Table 2.3, which lists the most active workstations. The second

column and third column show what percent of the total number of packets and what percent of the total number of bytes respectively had a source address belonging to the machine listed in the left column. Similarly, the last two columns represent the percentages of packets and data bytes received by the workstation. Thus, each of these four columns would add up to 100 percent when complete. Note that the workstations listed (28 out of a total of 130) account for about 80 percent of all packets.

TABLE 2.3. LIST OF SOURCE–DESTINATION TRAFFIC (PERCENTAGES)

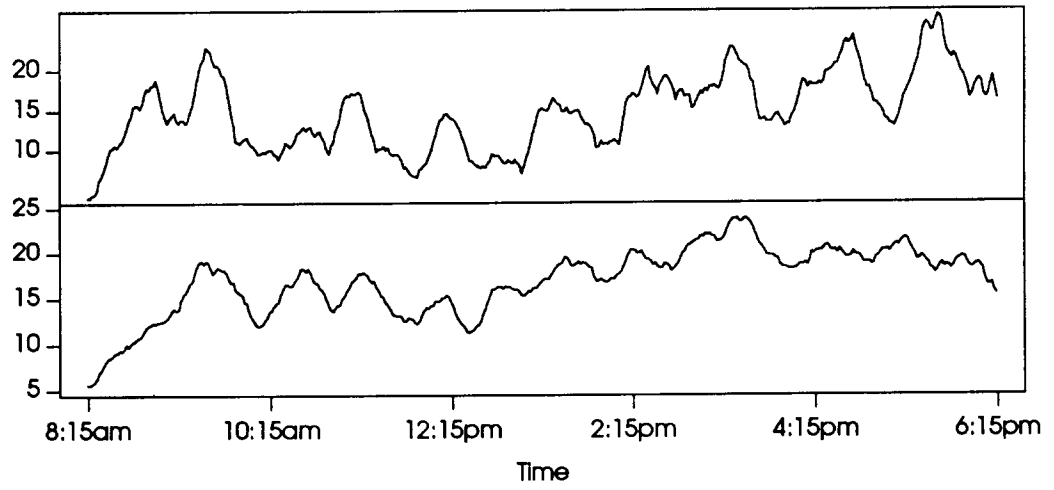
WORKSTATION	SOURCE		DESTINATION	
	PACKETS	BYTES	PACKETS	BYTES
station 21 (s)	18.6	19.3	15.4	12.6
station 7 (s)	8.3	9.2	6.2	4.1
station 23 (s)	7.6	10.8	5.6	6.6
station 28 (s)	6.8	6.7	5.9	3.7
station 5 (s)	4.8	6.7	3.1	2.9
station 9 (s)	3.1	3.1	2.4	0.9
station 13	2.4	2.5	2.8	3.6
station 27	2.0	3.0	1.1	0.7
station 11	2.0	0.4	2.3	1.2
station 25	1.9	2.4	2.8	4.5
station 14	1.7	2.1	2.6	4.2
station 3	1.6	1.8	2.4	3.5
station 15	1.6	2.0	1.6	2.0
station 8	1.6	1.8	1.4	1.5
station 20	1.4	1.1	1.2	0.6
station 22	1.3	1.8	1.3	1.9
station 1 (s)	1.3	0.1	2.1	2.7
station 26	1.3	1.5	1.8	2.7
station 2	1.3	1.6	1.5	2.2
station 19	1.2	1.3	1.1	1.0
station 4	1.2	0.6	0.9	0.6
station 24	1.1	0.7	1.2	0.9
station 10	1.1	0.7	1.1	0.7
station 16	1.0	0.9	2.8	5.0
station 18	1.0	0.7	1.0	0.6
station 6	1.0	0.5	1.1	0.8
station 17	0.8	1.0	1.2	1.8
station 12	0.7	0.6	0.9	0.9
Total	79.7	84.9	74.8	74.4

As we can see, the traffic in this network was rather balanced: no single workstation dominated over the others. In contrast, in the Berkeley study we found that a single diskless workstation on occasion generated 20 to 25 percent of the network load over short intervals of time and accounted for more than 16 percent of the total data bytes over a day. The balanced load in the Sun network suggests that there is a better match between hardware—machine power, local disks, large memory configuration—and user workload than in the Berkeley environment. In a university environment, students often are assigned limited resources, which they tend to overwork even if they have to pay the price of poor performance.

There is clearly a 24-hour periodicity in Figure 2.12 and a shorter periodicity in Figure 2.13. In the next chapter, we will examine in depth the question of whether or not the traffic produced by individual stations is stationary. Here, we are interested in a related question: whether the trends in the aggregate traffic are connected to the number of active workstations. If so, the nonstationary trends in Figure 2.13 can be predicted by the

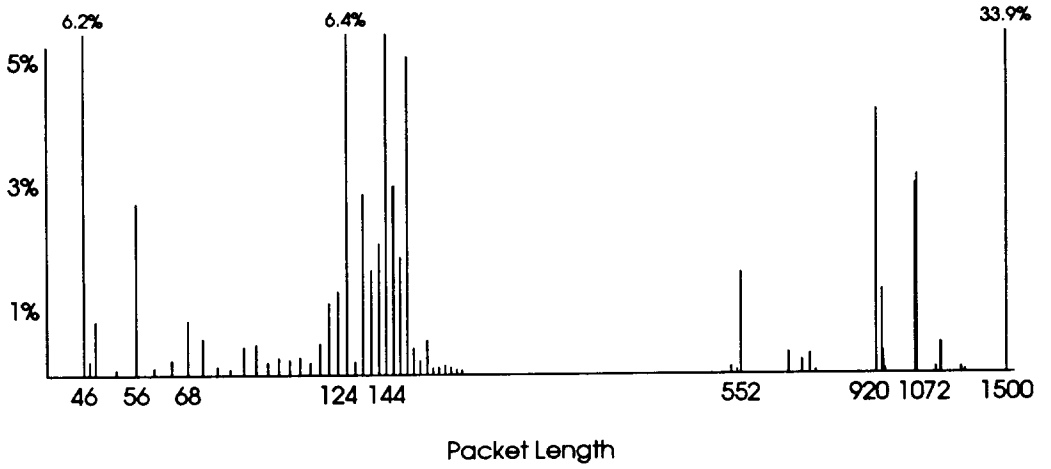
number of active workstations for some definition of "active". At any given time, one can define a workstation as active if it has transmitted at least a specified number of bytes or packets over a given interval. The "best" results were obtained when we considered active those workstations that sent more than 4 Kbytes in 1-second intervals. Even in this case, however, the association between the number of active workstation and network load was weak, as qualitatively illustrated in Figure 2.14, which shows a smoothed version of the load in Figure 2.13 and the smoothed number of workstations that transmitted more than 4 Kbytes in 1-second intervals. (The smoothing was performed in both cases by averaging together 40 consecutive values.) This is not surprising when we recall that different workstations produce different loads, and that even a single workstation generates varying loads during the day.

FIGURE 2.14. COMPARISON OF PERCENT NETWORK LOAD (TOP) AND NUMBER OF ACTIVE WORKSTATIONS (BOTTOM)



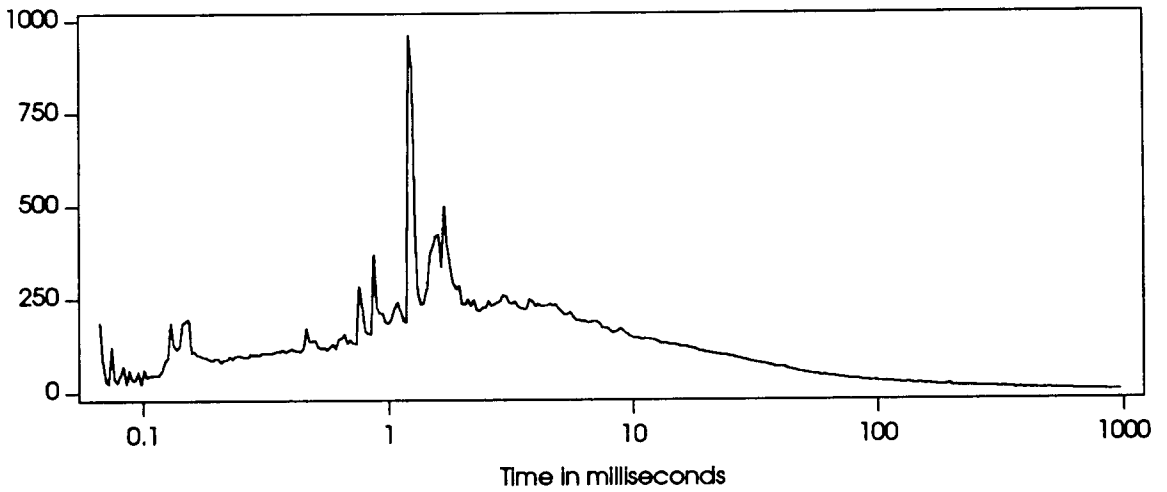
It is interesting to compare the network's packet length distribution with that of the Berkeley study. Figure 2.15 shows the percentage of packets transmitted in the Sun network as a function of the packet length. Its counterpart for the Berkeley network is Figure A.3. Figure 2.15 is very interesting. As expected, there are many fewer small packets than in the Berkeley network, in which many users accessed machines on other networks through remote login programs, producing small TCP packets for keyboard characters and for the associated protocol acknowledgements. (TCP produces in this figure the lines at 46 and 1064 bytes.) Also predictable is the much higher peak at 1500 data bytes, caused by the substitution of ND with NFS as the transport protocol for paging traffic. (Notice that ND is still being used by the server *station 1*; this protocol produces the lines at 48 bytes and 1072 bytes.) We observed that, except for the peaks mentioned above produced by TCP and ND, NFS generates all the other peaks, an indication of the homogeneity of the Sun environment. The higher weight of the mid-sized packets, those between 100 and 200 data bytes, which are NFS remote procedure call protocol packets, is more interesting. Together, the packets in this interval constitute 34.7 percent of the total. We had predicted in [33] that this segment of the packet-length distribution would have increased as more and more applications use RPC protocols.

FIGURE 2.15. PERCENTAGE OF PACKETS VS. PACKET LENGTH



Figures 2.16 and 2.17 show histograms of the interarrival times and of the interpacket times. In both graphs, the abscissa is logarithmic and on the ordinate we plot the square root of the number of events (interarrival and interpacket times respectively) that fall in a particular interval.

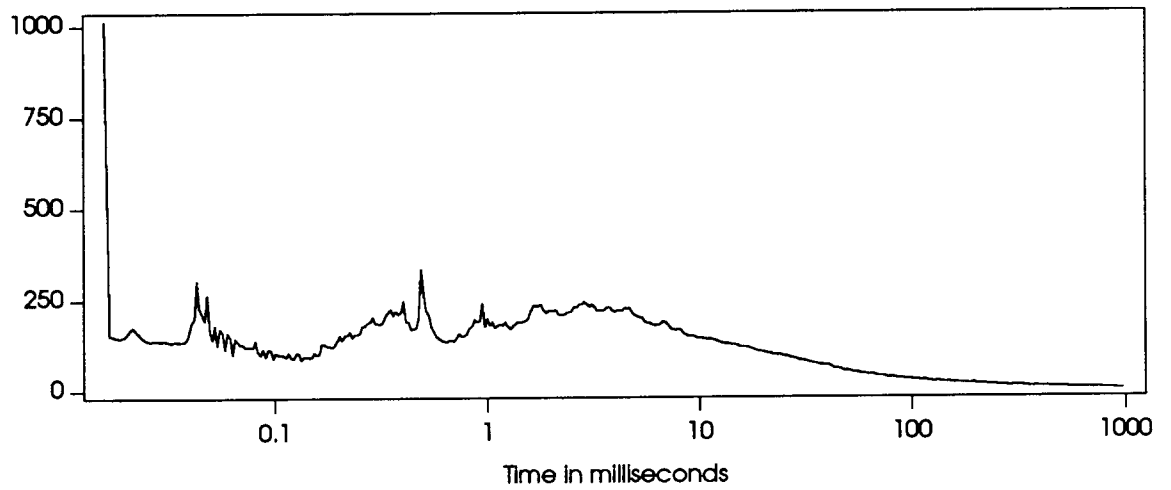
FIGURE 2.16. HISTOGRAM OF INTERARRIVAL TIMES – AGGREGATE TRAFFIC



The highest peak in Figure 2.16 is generated by consecutive 1500 byte packets. As we have seen in Section VII, most of these packet pairs have unrelated source and destination addresses, and are separated by the minimum interpacket times. Thus, most of them appear in the peak near the origin in Figure 2.17.

The histograms in Figures 2.16 and 2.17 are proportional to estimates of the marginal density probability functions of the interarrival and interpacket times. These estimated functions are difficult to model directly as they appear to be the superposition of discrete and continuous components. Although we understand the significance and the origin of

FIGURE 2.17. HISTOGRAM OF INTERPACKET TIMES – AGGREGATE TRAFFIC



the peaks, we can think of no simple analytical representation for them. The value of these histograms is the negative result that they make explicit: times between arrivals are not exponentially distributed,

There are significant correlations between subsequent interarrival times, but we will postpone the study of their second order properties to Chapter 3, in which we look at statistical descriptions of the arrival processes generated by individual workstations.

IX. Summary

In this chapter we have described and illustrated our measurement methodology and techniques. Measurements were taken through passive listening of all packets on an Ethernet local-area network. We used a dedicated machine with a modified network driver that collected packet protocol information and dumped it to large files, which were later transferred to magnetic tapes. The machine had a 2-MHz hardware counter that was used as a high-resolution clock. However, as our detailed timing error-analysis section has shown, the clock values in the traces are not as accurate as the original clock's 0.5 microsecond resolution. These timing errors, the great majority of which are less than 100 microseconds, are still small for the type of analyses we carry out in this study.

We have discussed the basic properties of the aggregate traffic of the Sun network mainly to place in a broader perspective the per-workstation statistical description of the traffic, which will be the topic of Chapter 3. The reader may also find it useful to compare directly these results to our previous analyses of traffic measurements in a university environment, which we reproduce in Appendix A.

3 Traffic Patterns of Individual Workstations

In Chapter 2 we have examined the aggregate, system-wide network traffic. Our next step will be to analyze the traffic from the perspective of the basic units that generate traffic—individual workstations. Although individual workstations are not the only conceivable units for studying network traffic—two alternatives are individual processes and clusters of clients workstations—they are the most logical choice. Individual (client) workstations are associated with individual (human) users and, at least to a certain degree, produce traffic when users are active. In this chapter we will perform several types of stochastic analyses on the arrival processes associated with the receive and the send queues of individual workstations.

A workstation's send queue process will be constructed by extracting packets whose source address is the workstation's network address from the general traces. Similarly, packets whose destination address is the workstation's address will be selected from the general traces and assembled together as the receive queue process. These will include packets originating from the workstation's file server as well as packets from other workstations. In the case of a file server, the transmit queue will contain packets addressed to several clients, some or all of which may be simultaneously active, and the receive queue will contain packets from all the clients. Thus, for file servers the packet arrival processes are the superposition of more basic client-workstation processes. As discussed in Chapter 2, each of the records of these extracted (sub)traces will contain timing information that will allow us to compute interarrival times.

Of course, the processes obtained in the manner described above will not be the same as those that would be recorded if the only machines on the network were source and destination machines. On an Ethernet, as the offered load increases, the network contention also increases, resulting in collisions, and, ultimately, in transmission delays. Although we will not consider the effects of collisions, in this chapter we will show that in the Sun Engineering network there is only a moderate amount of interaction among the various machines, i.e., the proportion of time in which two or more pairs of communication machines are active simultaneously is small. However, we make no claim that our network packet-arrival processes have the same stochastic properties as the packet-arrival processes that would be produced in a contention-free network.

In the next section we will look from a macroscopic time scale at the load patterns produced by individual workstations. These patterns consist of sequences of busy and idle segments, which we will analyze in Section III. In Section IV we will look at histograms of interarrival times, and associate protocol properties with the peaks in the histograms.

The processes we are dealing with are nonstationary (Section V). By extracting from each arrival process a subset of the interarrival times, which we define as the "busy period" in Section VI, we will obtain processes with better stationary properties. Using these subsets, we will study the autocorrelation coefficients of packet arrival processes in Section VII. Finally, we will discuss whether network serialization and sharing or shared file servers produce correlations in pairs of otherwise unrelated processes.

II. The Network Load of Individual Workstations

The various client workstations listed in Table 3.3 each generate roughly the same total volume of traffic during the working hours of a day. Yet, user behavior— busy/idle cycles of packet transmission—varies a great deal from workstation to workstation; differences in the work patterns of individual users account for this variation. It is instructive to look at the Ethernet load generated by the send queue and at the load directed to the receive queue of a couple of workstations.

Figures 3.1 and 3.2 display these loads for the workstation *station 3*. (Each vertical line represents the average load over a 30-second interval.) We notice that communication follows an on-off behavior typical of a person's work pattern. The spikes in the graphs indicate that there is burstiness, by which term we imprecisely refer to a high peak-to-mean ratio, when the load is measured at 30-second intervals. However, it is unclear whether this interval or a shorter one is appropriate for studying burstiness. In Chapter 4 we will introduce a formal definition of (interarrival time) variability and propose quantitative techniques for measuring it.

FIGURE 3.1. NETWORK LOAD (AS PERCENTAGE OF TOTAL) WORKSTATION *STATION 3* (SEND QUEUE)

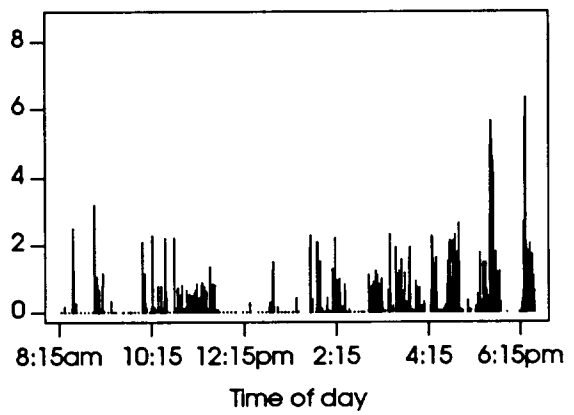
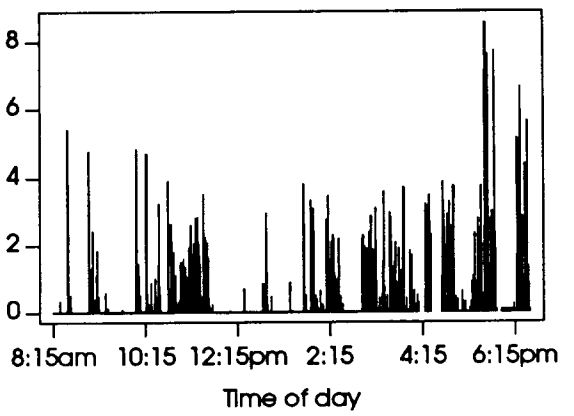


FIGURE 3.2. NETWORK LOAD (AS PERCENTAGE OF TOTAL) WORKSTATION *STATION 3* (RECEIVE QUEUE)



Station 3, a diskless Sun-3 model 50 with 4 Mbytes of physical memory, does not create a high load on the network in spite of its small memory, which could potentially cause a high paging rate. Nor did any of the other Sun-3/50's generate a high load. In the Berkeley study we saw that individual Sun-3/50's on occasion generate enough load to utilize one-fifth of the Ethernet raw bandwidth. This did not occur in the Sun environment, most likely because there was a better match between users' workloads and hardware capabilities than at Berkeley. In a university, where each machine is commonly shared by several student users, workstations tend to have shorter pauses (i.e.,

periods of idle time). Furthermore, students often overwork their hardware, sometimes at the expense of performance; in a company, the loss of productivity resulting from insufficient hardware power is not tolerable, and more powerful hardware is obtained for heavy users, while lighter users receive less powerful machines. Another reason for the network load difference is lower paging rates, which are the result of the improvements, mentioned earlier, in the virtual memory management scheme of SunOS 4.0. Finally, we note again that in the Sun network there are fewer diskless workstations, machines that depend on network communications for sending and receiving even the smallest bit of information to and from their file systems, the temporary file space, and the swapping area.

The dots visible in Figure 3.1 can be attributed to periodic messages exchanged between client and server, and are produced by small software tools that are frequently run on client workstations. Such tools typically monitor some performance index on the server (or on other machines), for instance the CPU load, displaying graphically the information on the user's display. Monitoring is performed by periodic packet transmittal, which introduces a deterministic component in the interarrival times, complicating our analysis but affecting the network load very little. In Figure 3.2, these periodic packets also appear as horizontal lines when relatively few other packets are sent.

FIGURE 3.3. NETWORK LOAD (AS PERCENTAGE OF TOTAL) WORKSTATION *STATION 16* (SEND QUEUE)

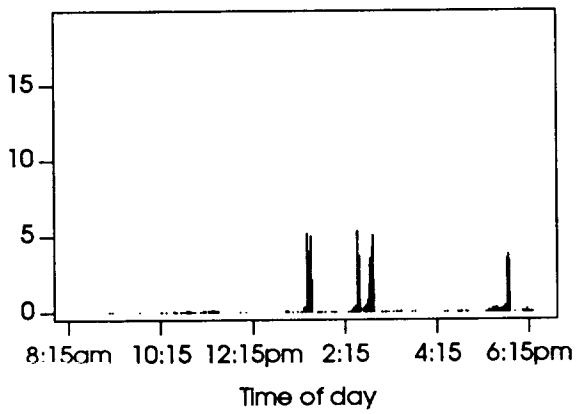
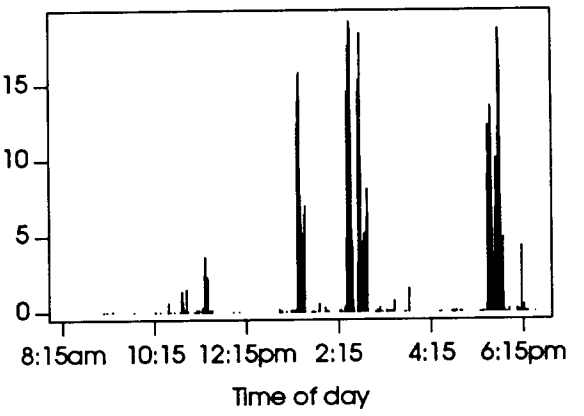


FIGURE 3.4. NETWORK LOAD (AS PERCENTAGE OF TOTAL) WORKSTATION *STATION 16* (RECEIVE QUEUE)



As another example of the types of traffic patterns produced by a client workstation, we show in Figures 3.3 and 3.4 the load generated on the Ethernet by those packets sent and received by *station 16*, a Sun-4 model 260, with 16 Mbytes of main memory and a 250-Mbyte local disk. Local disks on client workstations are typically configured to hold the root file system and the swapping partition, to both of which much of the UNIX disk traffic is directed. This disk traffic produces relevant network traffic if the machines are diskless. Hence, workstations with local disks, as in the *station 16* example, display lower network utilization. The figures show that while *station 16* does not keep the network as busy as as *station 3* does, *station 16* produces peak network loads that are higher and longer than *station 3*'s because its faster architecture can sustain higher transmission data rates. This pattern is probably caused by the workstation transferring data from other workstations to its local disk during its busy periods. The high

number of 1500-byte packets in Figure 3.6, plot of the packet length distribution in the receive queue, indicates the higher activity of its receive queue.

FIGURE 3.5. PERCENTAGE OF PACKETS VS. PACKET LENGTH WORKSTATION *STATION 16* (SEND QUEUE)

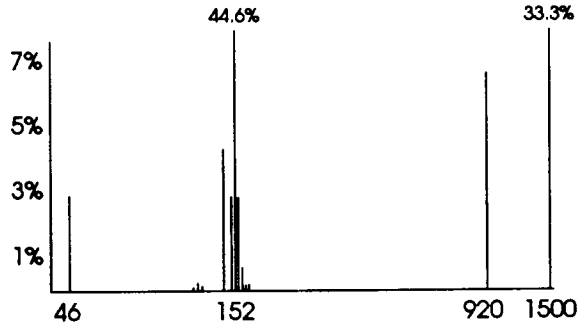


FIGURE 3.6. PERCENTAGE OF PACKETS VS. PACKET LENGTH WORKSTATION *STATION 16* (RECEIVE QUEUE)

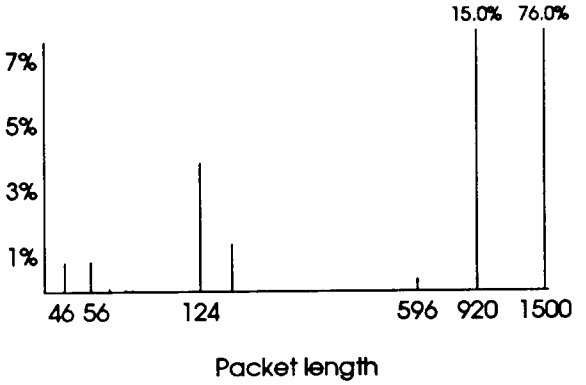


Figure 3.5 shows that medium-sized packets are over-represented in comparison with the network average shown in Figure 3.15. In contrast, medium-sized packets are under-represented in the receive queue (see Figure 3.6). With such unusual packet size distributions and network load patterns, traffic models based on *station 16* data will not be representative of typical-traffic patterns.

III. Busy and Idle Phases

Figures 3.1 through 3.4 reveal on a macroscopic time scale of busy/idle intervals just how variable network traffic patterns driven by individual user behavior can be. Busy periods—time intervals during which the network load is significantly different from zero—can range from 30 seconds to 30 minutes; idle periods from a few seconds to several hours. We have studied the autocorrelation coefficients of the lengths of busy and idle segments for the send and receive queues of several workstations. The values of the coefficients suggest that these processes could be modeled as alternating renewal processes, i.e., as processes in which successive intervals are sampled from two independent probability distributions. The series of correlation coefficients of an alternating renewal process alternates between a positive and a negative value [19, page 89]. The autocorrelation coefficients of the estimated busy/idle phases do alternate, although they decrease as the computed correlation is for busy/idle phases farther and farther apart (i.e., as the lag increases) and eventually become negligible. Another model, perhaps more suitable for these types of data, would be a first-order autoregressive process of the form $X_{i+1} = \alpha X_i + Y_i$, where Y_i is a purely random process and the coefficient α is negative. The autocorrelation coefficients of this type of autoregressive process also alternate, but their absolute values decrease as the lag increases.

Beside these general observations, we will not attempt to describe the stochastic properties of the busy/idle periods because we do not have enough data to make parameter estimation statistically meaningful. To discriminate between different alternative models requires longer data sets, perhaps spanning several days. Instead, we will

focus on the busy periods alone. Busy periods are the most important aspect of packet arrival processes: it is during these intervals that queue buildups occur; that network contention is reflected in the response time; and that, in general, protocol and system performance determines the shape of the arrival processes themselves. Later in this chapter we will provide a formal definition of "busy period" and characterize in detail busy periods of several client workstations and file servers.

IV. Histograms of Interarrival Times

In this section we will analyze histograms of interarrival times. Assuming that the series are stationary, interarrival-time histograms are, except for a scale factor, biased estimates of the marginal probability density functions of the interarrival times. In fact, the probability that interarrivals X_i's assume values in an interval of size δ centered at x can be estimated by

$$\hat{\Pr}(x, \delta) = \frac{\sum b(X_i)}{N}$$

where N is the number of elements in the time series and b is the function

$$b(t) = \begin{cases} 1 \text{ if } x - \frac{\delta}{2} \le X_i < x + \frac{\delta}{2} \\ 0 \text{ otherwise} \end{cases} .$$

If the time series is stationary, the estimated probability is unbiased. Hence

$$\Pr(x, \delta) = E(\hat{\Pr}(x, \delta)) = \lim_{N \to \infty} \hat{\Pr}(x, \delta) = \lim_{N \to \infty} \frac{\sum b(X_i)}{N} .$$

Since $\Pr(x, \delta)$ is related to the probability density function by

$$\Pr(x, \delta) = \int_{x-\frac{\delta}{2}}^{x+\frac{\delta}{2}} p(t)dt$$

by definition, if δ is small we can approximate the integral with the product $p(x)\delta$. Rearranging, we obtain

$$p(x) = \lim_{\delta \to 0} \frac{\Pr(x, \delta)}{\delta} = \lim_{\substack{N \to \infty \\ \delta \to 0}} \frac{\hat{\Pr}(x, \delta)}{\delta} = \lim_{\substack{N \to \infty \\ \delta \to 0}} \hat{p}(x) ,$$

where we have defined

$$\hat{p}(x) = \frac{\hat{\Pr}(x, \delta)}{\delta} = \frac{\sum b(X_i)}{N\delta} , \tag{3.1}$$

a sample estimate of the probability density function.

From equation (3.1) we can compute the expected value of $\hat{p}(x)$:

$$E(\hat{p}(x)) = \frac{E(\hat{\Pr}(x, \delta))}{\delta} = \frac{\Pr(x, \delta)}{\delta} = \frac{1}{\delta} \int_{x-\frac{\delta}{2}}^{x+\frac{\delta}{2}} p(t)dt .$$

Thus, in general,

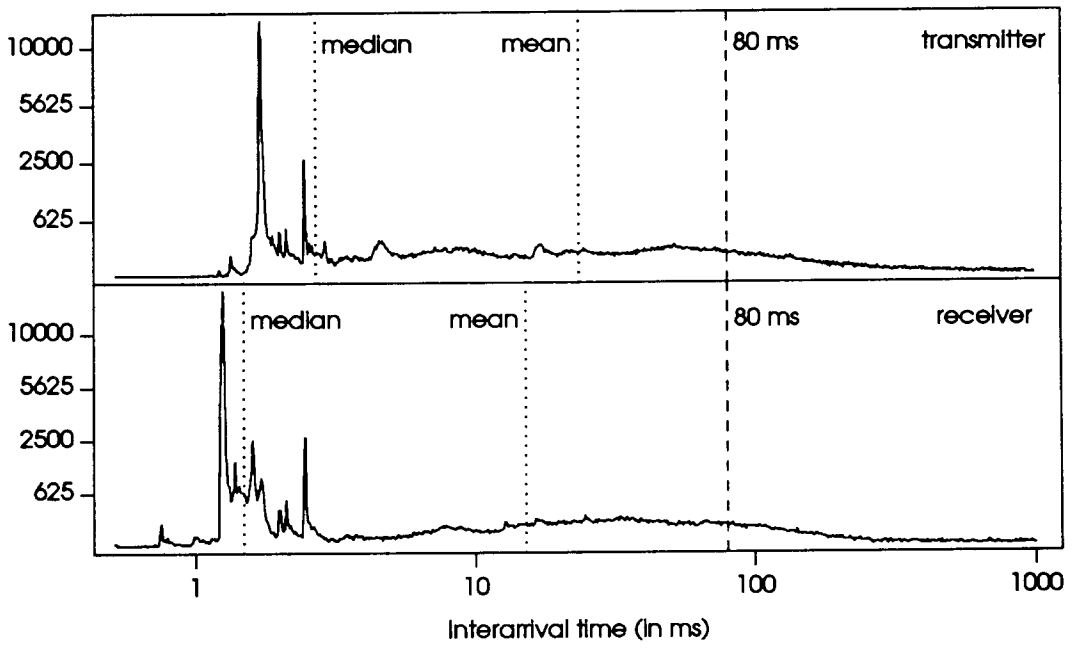
$$E(\hat{p}(x)) \neq p(x) ,$$

that is, $\hat{p}(x)$ is a biased estimate of $p(x)$. In addition, it can be easily shown that the bias is proportional to δ, the size of the interval.

In the figures of this section, we have chosen to indicate on the ordinates the values of the counts $\sum b(X_i)$ instead of the scaled version. This will allow the reader to relate more directly the height of the various points to the size of the data sets.

In Figure 3.7 we plot the histograms of the interarrival times of packets generated by the send queue and received by the receive queue of the client *station 25*. Each histogram contains 100000 interarrival times. The abscissas of each graph were modified by a logarithmic transformation; on the vertical axis, we report the square roots of the bin counts.

FIGURE 3.7. HISTOGRAM OF INTERARRIVAL TIMES – CLIENT *STATION 25*



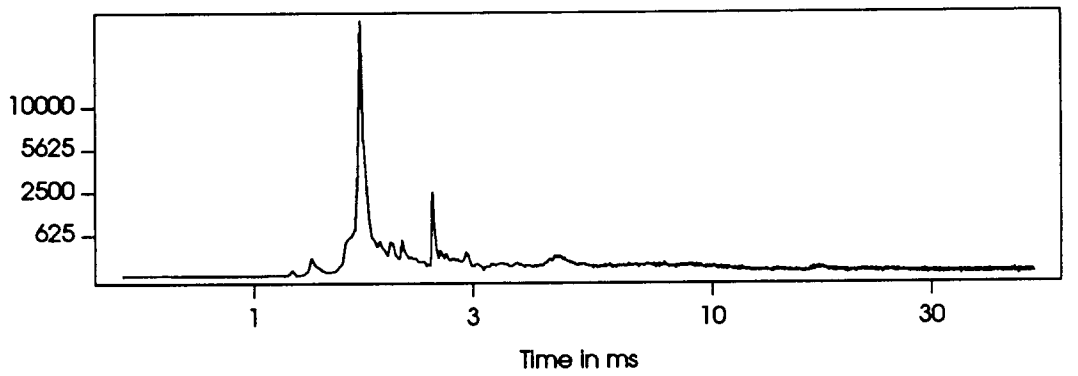
In Figure 3.7, and in the following histograms presented in this section, the estimation procedure is rather sophisticated: the size of the bins over which the histogram is computed is not constant, but increases exponentially, that is, the size of the bins is constant in logarithmic coordinates. The histogram is computed in a standard way, by counting the number of interarrival times that fall in each bin. Hence, in order to "correct" the shape distortion introduced by this method, one would have to multiply each point by a value obtained from a negative exponential curve.

Using exponentially increasing bins has several advantages. First, a single histogram can capture interarrival times spanning more than three orders of magnitude, which would not have been possible with equally spaced bins. Second, since the majority of interarrival times are concentrated in the 1 to 3 ms range, smaller bins on the lefthand

side region of the histograms allow us to display the curves with higher resolution in that region. Third, since the variance of interarrival times within classes of communication events grows with the increase in the mean interarrival times of these classes of events, the procedure matches more closely the size of the bins to the variance (or, more properly, to the standard deviation of interarrival times) and permits the representation of all of these classes as peaks in the histograms. In contrast, in a standard histogram, classes of interarrival events whose standard deviations are larger than the size of the bins could be invisible. For instance, without considering network contention and collisions, a class of closely spaced interarrivals, such as those resulting from protocol fragmentation, shows a small variance and produces a narrow peak. Instead, periodic (but less frequent) events generated by programs that access a peripheral (such as a magnetic disk) show a much larger variance, and need a larger bin in order to appear as peaks. Finally, a fourth advantage of this procedure is that it amplifies the values of the histogram for large interarrival times allowing us to discriminate trends and properties that we would otherwise miss: for instance, without this feature we would not have seen the decrease in the interarrival times after 100 ms in Figure 3.7.

In order to provide an explicit comparison between a histogram with constant bin size and one with exponentially increasing bin sizes, in Figure 3.8 we show a constant-size-bin histogram that corresponds to the first part of the top graph in Figure 3.7.

FIGURE 3.8. CONSTANT-SIZE-BIN HISTOGRAM OF INTERARRIVAL TIMES – CLIENT *STATION 25* (TRANSMITTER QUEUE)

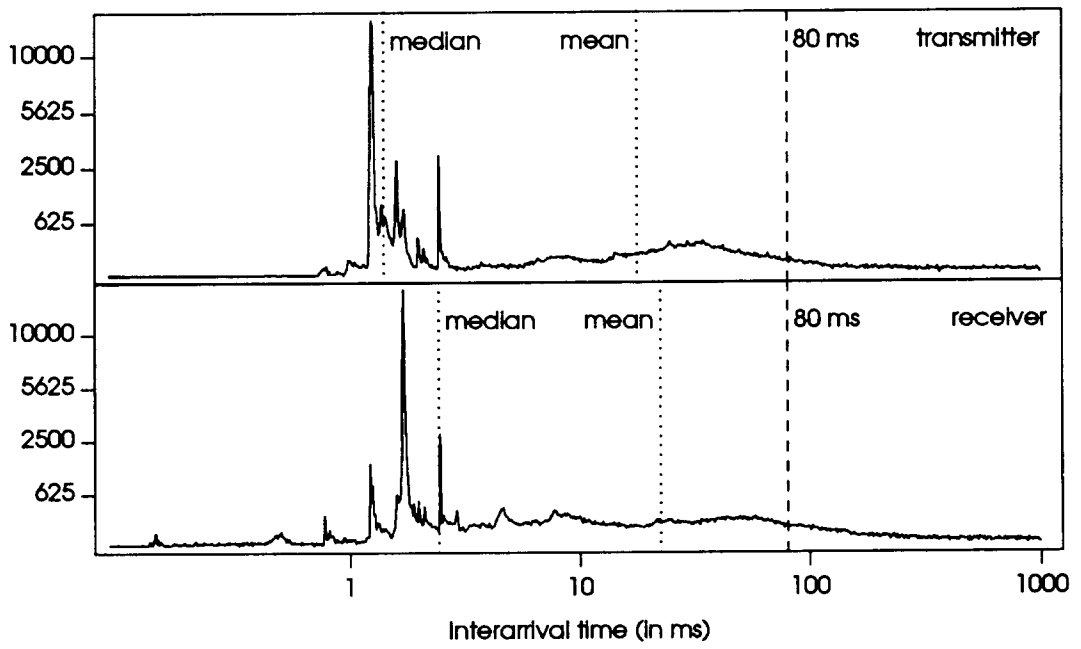


In this figure we have used the same convention as in Figure 3.7: the abscissa is displayed in a logarithmic scale, and on the ordinate we plot the square root of the counts. Observe that the highest peak in Figure 3.8 is higher than the highest in Figure 3.7 since we use a bin that is actually larger than the bin used for the corresponding region in Figure 3.7, which, therefore, has a higher resolution in that region. It is important to note that this higher resolution does not involve higher computational costs; on the contrary, in Figure 3.7—showing histograms with only 800 bins—bins below 5.4 milliseconds are smaller than the constant-size bin in Figure 3.8; this figure contains 2000 points, and hence has a bin of about 25 microseconds. Because of the logarithmic x-axis scale, which results in a higher concentration of points in the right half of the curve, in Figure 3.8 the histogram is finely jagged for times larger than 10 milliseconds. Also, notice how the curve is stretched and flattened in this area, making it difficult to discern any trends or patterns. Changing the y-axis scale does not help: as mentioned above, there are too few points in each bin

for any significant pattern to show up in the representation of Figure 3.8. Returning to Figure 3.7, we can appreciate how much easier it is to see the small bump at 13 milliseconds than it is in the constant-size-bin histogram.

A drawback of our variable-size-bin procedure, indicated by our analysis of the statistical estimate of probability density functions, is the increasing bias as interarrival times lengthen. However, since we do not plan to use these histograms (or probability densities) for quantitative analysis, but only to show qualitative properties of our arrival processes, this is not a major problem. Furthermore, given the nonstationarities present in our processes, histograms are affected by other errors that are difficult to quantify. Thus, it is not clear whether regular histograms could be used quantitatively anyhow.

FIGURE 3.9. HISTOGRAM OF INTERARRIVAL TIMES – SERVER *STATION 23*



In Figure 3.9 we show histograms of interarrival times for the send and receive queues of file server *station 23*. Although this figure and Figure 3.7 look roughly similar, there are several differences worth noting. The first portions of the histograms, showing interarrival times up to 10 milliseconds, depend on protocol time constants. As a result, the two workstations produce peaks at about the same points in the top histograms of each figure. However, whereas the highest peak in *station 25*'s send queue occurs at 1.71 milliseconds, the highest peak in *station 23*'s case occurs at 1.25 milliseconds. Since 1.2 milliseconds is the minimum time between two 1500-byte messages, *station 23* must be sending these fragmentation packets back-to-back. (In Chapter 2, we have discussed the consequences of minimum-interframe packets on the measurements.) These peaks correspond to the interarrival times between successive 1500-byte packets generated by the IP protocol fragmentation of 8-Kbyte NFS messages (see Appendix A for details on this type of message fragmentation). In *station 25*, these messages are write messages; in *station 23* they are replies to read requests. The relative shift can be

explained by the difference in speed between the two CPUs: *station 25* is a 15 MHz Motorola 68020 and *station 23* a 16 MHz Sparc, which, in spite of the comparable clock rate, has a much faster architecture. (Analytical and measurement studies [33,49,64] have pointed out that the bottleneck in the file server is most often the CPU, and, as a result, a network system of this type is most often configured with the fastest machine acting as a file server for several, less powerful clients.)

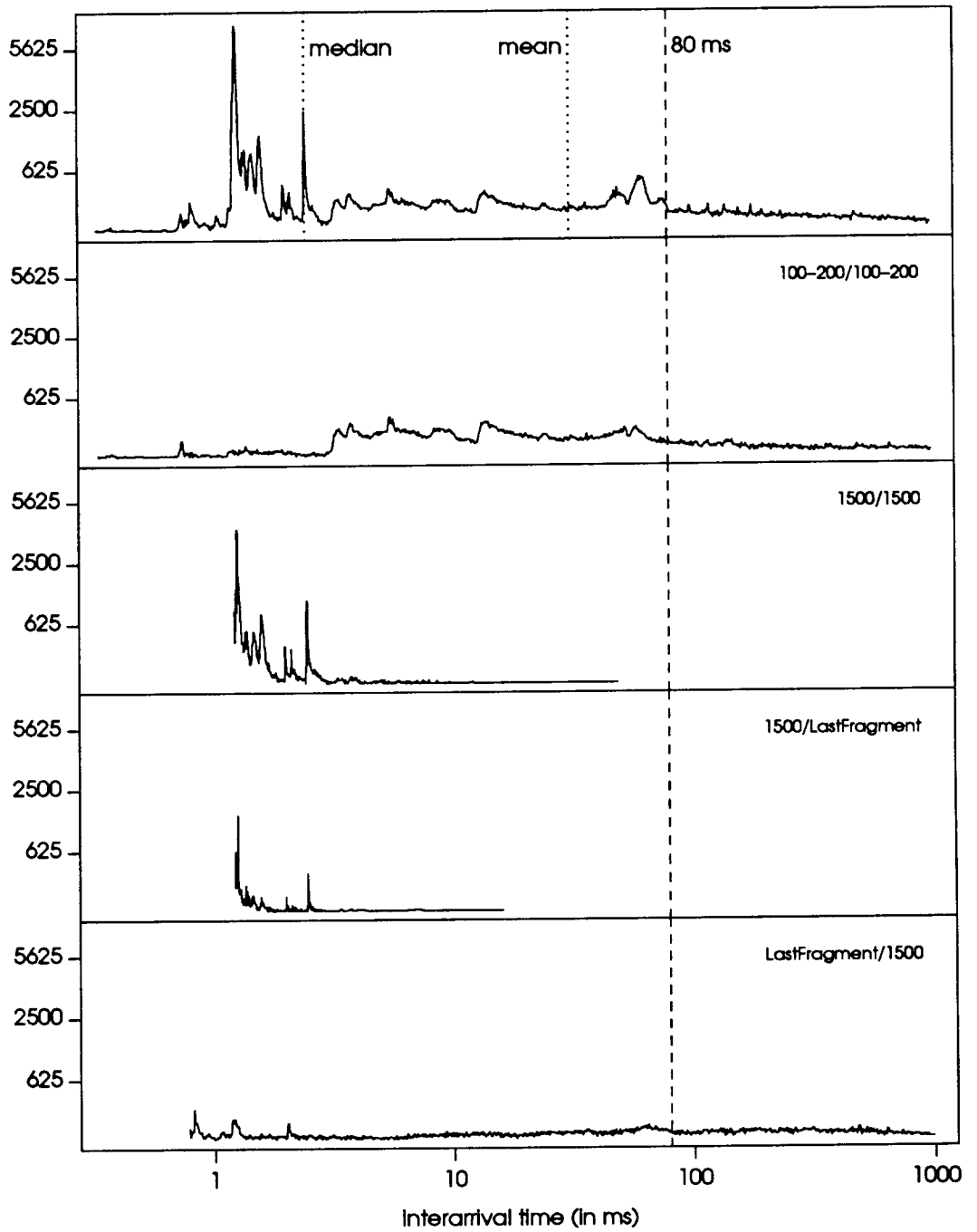
The same reasoning explains why the peaks of the transmitter are shifted to the right of those of the receiver in Figure 3.7 (the receiver queue of a client is related to the transmit queue of its file server). Because *station 23* is a file server, the situation is reversed in Figure 3.9. Also, since its receive queue is the result of the overlapped transmission of several clients of different speeds, the result is the multiplicity of peaks we see in the lower graph. *Station 23*'s simultaneous communication with several other workstations also explains why there is a nonzero probability that times between packets arriving at its receive queue are very small. This is shown by the extensions of the points to the left of 1 millisecond in Figure 3.9.

In the top graph of Figure 3.10 we show a histogram of 100000 interarrival times produced by packets arriving at the send queue of client workstation *station 15*. Although this machine has a local disk, it was relatively busy communicating with file servers and other clients, and its the estimated marginal distribution of interarrival times looks quite similar to those we have seen before. In the other four graphs in Figure 3.10, we show some principal components of the total traffic produced by the send queue. First, in the second graph from the top, we have the histogram produced by successive NFS procedure calls, the packets that in Figure 3.15 occupied the region of packet lengths between 100 and 200 bytes.

We recognize that these packets generate interarrival times that occupy primarily the middle region of the histogram, running from 4 to 80 milliseconds. NFS remote procedure calls require substantial processing time, and hence more time between successive send or receive operations, because they often require disk accesses, for instance during path-name translations; because of more complex protocol headers; because of the need for marshaling argument data types; because of the need to convert the data to an external representation so that it can be received by machines with other number representations or storage format conventions; and so on. In this region the curve is more or less flat, which means, recalling the way in which the variable-size-bin histogram works, that the interarrival time density function should be a negative exponential. The time between events in a Poisson process is exponentially distributed. However, it is doubtful that a Poisson process would model the data appropriately, since it appears that successive arrivals in the "100-200/100-200" class are strongly correlated.

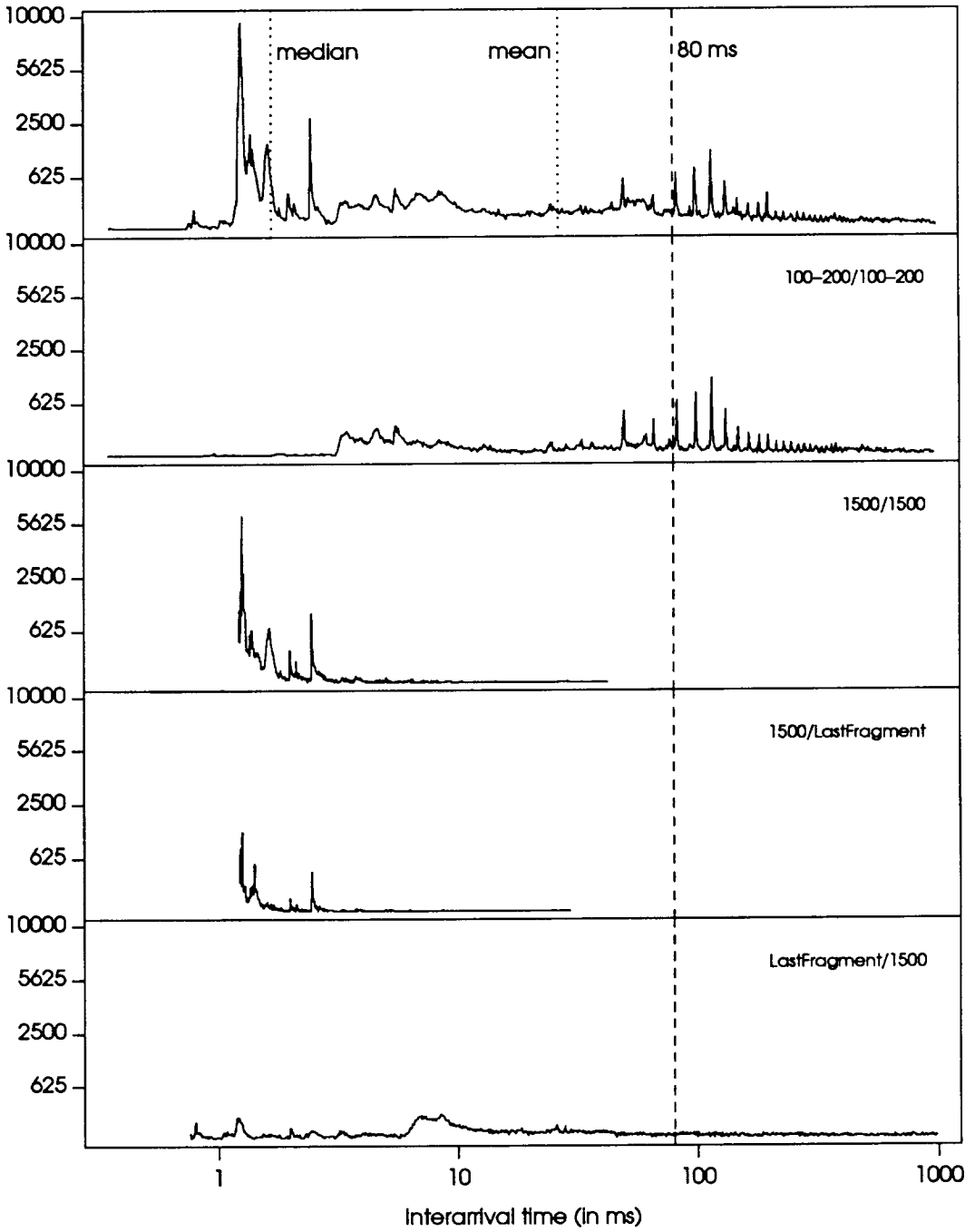
When the IP protocol layer fragments 8-Kbyte messages, it generates six fragments, the last of which is shorter; this produces four "1500/1500" interarrival times and one interarrival that we will call "1500/LastFragment". The distribution of these times is shown in the next two plots of Figure 3.10. These plots indicate that the interarrival times generated by the last fragments are not different from those of the previous "1500/1500" fragments. Peak locations and histogram domains are very similar. Peak values in the first of the two histograms are in a ratio of 4 to 1 to the peak values in the second histogram; this ratio corresponds exactly to the ratio between the frequencies of "1500/1500"

FIGURE 3.10. HISTOGRAMS OF INTERARRIVAL TIMES – CLIENT *STATION 15* (SEND QUEUE)
BREAKDOWN OF INTERARRIVAL TIMES BY PACKET LENGTHS



pairs and "1500/LastFragment" pairs. We see that protocol fragmentation produces small interarrival times and calculate that about 44.5 percent of the interarrival times belong to these two classes. We observe also that the smaller peaks in these plots may be the result of transmission delays due to network contention and transmission reschedulings caused by collisions.

FIGURE 3.11. HISTOGRAMS OF INTERARRIVAL TIMES – CLIENT *STATION 15* (RECEIVE QUEUE)
BREAKDOWN OF INTERARRIVAL TIMES BY PACKET LENGTHS



Most often these 8-Kbyte messages are part of larger user-level data transfers; thus, it is quite common that two or more of them follow each other. If this happens, the packet after the last fragment of one message is the first 1500-byte fragment of the following message and it is interesting to look at the distribution of the times between these two classes of packets. We show this distribution in the last histogram of Figure 3.10. In

this case we have a flat curve across the entire domain and the same considerations we made above for the middle part of the second plot apply here as well.

In Figure 3.11 we show plots equivalent to those of Figure 3.10 for the receive queue of workstation *station 15*. These histograms are, in general, quite similar to the previous ones. A distinctive feature, however, is the comb-like sequence of peaks in the aggregate histogram, which would not have been as apparent without the use of variable-size bins. As the second plot shows, the peaks are caused by interarrival times between medium-sized NFS packets. We do not know the source of these consecutive short NFS packets, although we have determined that all of the packets follow each other within a relatively short interval of real time. Most likely, they were generated by a particular user application. Because these interarrival process patterns are unusual, we decided not to be concerned with this level of detail in the models.

Another interesting feature of the *station 15* receive-queue histograms appears is in the last plot (the "LastFragment/1500" plot), in which there is a small peak centered at 8 milliseconds and extending from about 7 to 9 milliseconds. These delays between successive 8-Kbyte messages appear to be caused by disk accesses: *station 15*'s receiver activity consists of packets coming mostly from its file server.

Finally, as an example of the histograms produced by a less utilized workstation, in Figure 3.12 we display graphs of interarrival times for client *station 11*. These plots were based on roughly four-hour traces rather than 100000 interarrival times. Since the traffic of this workstation is highly unbalanced between the transmit and receive queues, had we considered only 100000 arrivals for each queue, we would have covered two quite different intervals of real time. Here, also, we have truncated the plots at the interarrival time of 1 second because there were quite a few long interarrivals that would have excessively compressed the more interesting areas of the histograms.

The swollen intermediate part and the absence of the typical high peaks between 1 and 2 milliseconds in the top graph, representing the interarrival times between packets transmitted by *station 11*, indicates that the transmitter did not transfer many data. The NFS protocol caches for 30 seconds file descriptor information (see Appendix A), which then needs to be revalidated—a process that generates medium-sized interarrival times. Since NFS is a request-response protocol, the receiver graph also shows a developed intermediate area, if compared with Figures 3.7 and 3.9. Like *station 16*, *station 11* is not a workstation on which to base typical-traffic models.

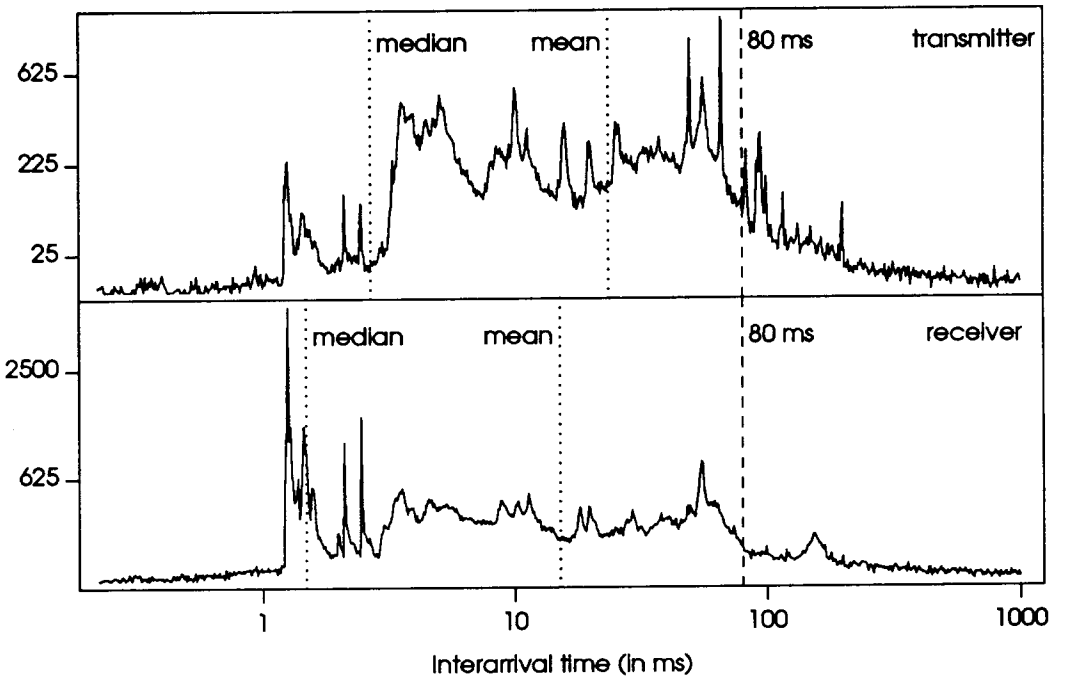
The histograms of interarrival times that we have discussed in this section are related to the marginal probability distributions of interarrival times. Alone, they do not fully describe the stochastic processes we are studying; it is necessary to represent the joint probability structure of each of these processes, that is, for interarrival times,

$$F_{X_1 X_2 \cdots X_n}(x_1, x_2, \ldots, x_n) \Pr(X_1 \le x_1, X_2 \le x_2, \cdots, X_n \le x_n) \tag{3.2}$$

for all n.

Using joint distribution and density functions is very complicated; however, since most of the properties of stochastic process that are of interest in the description of packet arrival processes depend only on the first- and second-order joint moments [8, Chapter 6], we will complement the histogram analysis with the study of the

FIGURE 3.12. HISTOGRAM OF INTERARRIVAL TIMES – CLIENT *STATION 11*



correlation properties of packet arrival processes. We will carry out the correlation analysis after discussing, in the next section, whether the arrival processes are stationary.

V. Stationarity of Packet Arrival Processes

In this section we determine whether the stream of packet arrivals of a single user workstation constitutes a stationary point process. A process is (strictly) stationary if its joint distribution (3.2) is invariant under translations in time (or serial number), that is, formally, if

$$F_{X_{i+1}X_{i+2}\cdots X_{i+n}}(x_{i+1}, x_{i+2}, \ldots, x_{i+n}) = F_{X_{j+1}X_{j+2}\cdots X_{j+n}}(x_{j+1}, x_{j+2}, \ldots, x_{j+n})$$

for all i, j, and n. The property of stationarity is a highly desirable one for a point process: stationary processes benefit from a wealth of analysis techniques and lead to simpler models. Unfortunately, as is often the case, only rarely do "real-life" processes conform to this simplifying property.

Requiring that the marginal and joint probability distributions be time invariant is quite restrictive. Since we will base our descriptions of packet-arrival point processes only on first- and second-order properties, we will also choose a weaker definition of stationarity: a process is stationary if its first- and second-order descriptors are time invariant. In addition, since we will work both with count and interarrival time descriptions, we observe that the two are complementary aspects of an arrival process: a count process and an interarrival time process. The count and interarrival time processes are related because the sum of the counts in an interval of size t is less than or equal to k if and only if the sum of k interarrival times is larger than t. If we assume that a process has a point at the origin,

we can write

$$\Pr(\sum_{i=1}^{n} c_i \le k) = \Pr(\sum_{j=1}^{k} X_j > t) .$$

Although from a strictly probabilistic viewpoint, if either of these processes is stationary, the other is not necessarily so (for details see [19, page 67]), from a practical viewpoint we will assume that the stationarity of one of the two processes will imply the stationarity of the other.

Classical time series analysis considers departures from the assumption of stationarity that have the form $X_t = Y_t + Z_t$, where the series Z_t is stationary and Y_t is not. If the series Y_t varies slowly with respect to Z_t, it is called a *trend*. A good practical rule regarding stationarity is given by Granger [32], who defines "trend in mean" as comprising all frequencies whose wavelengths exceed the length of the time series. This definition is important because it sheds light on the notion of "relevant time scale" that we will introduce shortly.

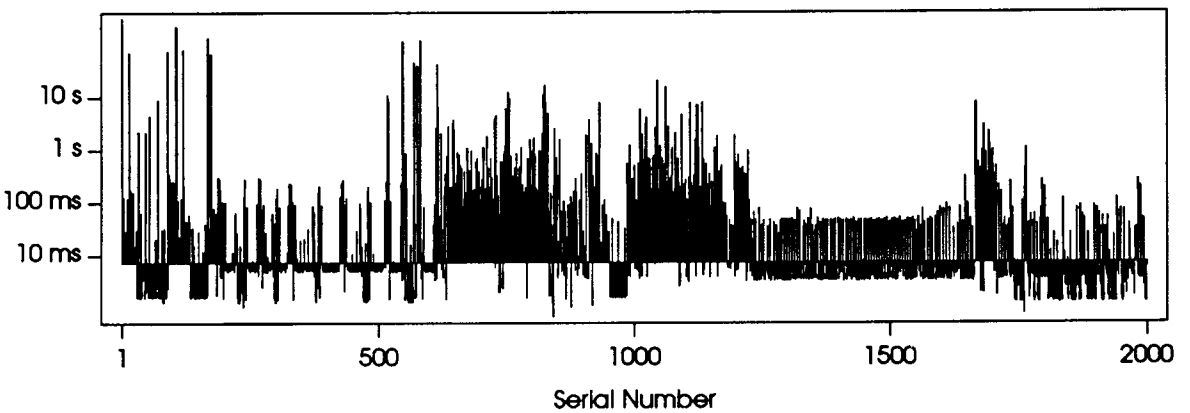
Series with trends can be dealt with by transforming the data so as to remove the trend. A typical transformation is by way of a linear filter:

$$W_t = \sum_u a_{t-u} X_u .$$

If the filter is high-pass, the low frequency components, i.e., the trends, in the original series are removed. A special high-pass filter can be obtained by differencing the series—an approach stressed by Box and Jenkins who maintain that a time series should be differenced until it becomes stationary [10]. (The differencing operation replaces two successive points in a time series with their difference.)

Another technique used to remove trends in classical time series analysis is by means of an interpolating curve. The differences between the curve and the original series are then taken, the residual providing estimates of the local variations. However, this is feasible only when the number of points in the series is small.

FIGURE 3.13. TIMES BETWEEN SUCCESSIVE ARRIVALS



The use of data-transformation techniques is not advisable with our arrival processes because much of the information we need in order to correlate properties of the processes to protocol and system time constants would be lost. Moreover, nonstationarities in our arrival processes do not satisfy the definition of trend given above. As an example, in Figure 3.13 we plot 2000 interarrival times, amounting to almost half an hour of real time, for the receive queue of *station 11*. On the vertical axis we report the interarrival times through a logarithmic transformation and on the abscissa we plot the serial number. An artificial zero line has been placed at the value corresponding to the median of this series; in this way, we can distinguish graphically short interarrival times embedded into sequences of long ones, since the former are printed toward the bottom of the graphs and the latter toward the top. As clearly seen in this figure, the sequence consists of segments, which we call phases, ranging from a few units to a few hundred interarrivals, in which the characteristics of the arrival process appear constant. Transitions between phases occur abruptly, that is, with high frequency components, thus, these transitions could not be classified as "trends" according to Granger's definition.

The question of whether a time series is stationary can also be a question of time scale. For example, all would agree that the load patterns in Figure 3.13 seem, to visual inspection, nonstationary. Yet, this particular user might follow the same work patterns and produce roughly the same workload in successive half hours. Or the work (and network load) patterns might follow a daily or weekly cycle. Therefore, if we looked at a period of several days, instead of just at half an hour, the data might look more clearly stationary.

There are at least two time scales of interest in our study. First, the busy/idle time scale about which we have already commented in the previous sections. This involves time intervals from many seconds to many minutes during which the network load generated by a workstation changes appreciably. A second time scale of interest, one over smaller time intervals, ranging from a few milliseconds to a few seconds, is one over which the packet rate of the sender's queue changes appreciably. A fine time scale is more important than a coarse one in the context of our study of the variability of packet arrival processes that are produced and consumed in an interactive system, in which response time is the key performance index. Hence, for us the "relevant time scale" will be the second: a time scale that will capture the queuing dynamics of packet arrival processes.

An important consequence of our approach is that we can drop long interarrival times since they will not affect the performance of the system over small time periods. In particular, we may exploit this in order to remove nonstationary components from the packet traces. It is clear that a nonstationary sequence at the coarser time scale will result in a nonstationary sequence at the finer time scale. If we, instead, restrict our attention to a subset of the arrival stream, for instance to a subset corresponding to a region of high network load, the resulting processes at the finer time scale may be stationary. We will explore this idea in our study of busy periods.

VI. The Busy Periods

In this and the next section, we will focus exclusively on busy periods of packet transmission, as defined below. Our methodology will be to extract from the full traces the sets of interarrival times occurring during busy periods, and to treat these as a single sequence. This approach is analogous to our extraction from the aggregate traces of the packets transmitted by a particular workstation and computation of their interarrival times: in both instances, we narrow the objects of our stochastic analysis to certain subsets of the general traces.

Our first step is to define the subset consisting of *busy periods*. A busy period could simply be defined as a period during which a user is active at the workstation. But drawing the line between busy and idle spans on this basis is far from satisfactory. In a long compilation task (or any long computation accessing the file system) that proceeds without user supervision, CPU and network activities are not related to the presence of a user at the workstation. Conversely, it is quite possible for a user to be active, without generating any network traffic. A more satisfactory criterion for the definition of busy period would be interarrival time length: a workstation is deemed to be busy when the interarrival times at its send and receive queues are shorter than a specified length.

To choose the cut-off point, we return to the histograms of interarrival times that we previously examined in Section IV. In each of those figures, which we have truncated at 1000 milliseconds, we see that at approximately 80 milliseconds the curve drops quite noticeably. (Dashed lines have been drawn at the 80 millisecond mark.) The longer interarrival times do not pop up randomly in the time series, but tend to occur in clusters during periods of low activity such as when a user logs off or leaves a workstation idle. Typically, only less than eight percent of the points occur after the 80 millisecond mark; yet, because of their large values and their occurrence in clusters, this small fraction of interarrival times accounts for most of the nonstationary behavior of a workstation. Hence, we chose 80 milliseconds to be the cut-off point: whenever the interarrival time exceeds this value, the machine is not deemed to be busy. We experimented with a few other values for the cut-off point in the range from 60 to 100 ms; no significant difference was evidenced and we conclude that the sensitivy of the following analyses to the 80 millisecond mark is minimal.

VII. Autocorrelation Coefficient Series

Correlation is a measure of causation in the data, and is often (but not always) associated with cause-effect phenomena. For example, the transmission of a packet may generate other transmissions within short, predictable times. We have already seen that this is the case for packet fragmentation of 8 Kbyte NFS messages. As another example, because of packet serialization over the network, there is a causal relation between interarrival time and packet length, since transmission of one packet must be completed before the next packet can be sent. Correlation coefficients provide a quantitative measure of the linear correlation among successive elements of a series.

The sample autocorrelation function of a series is defined as the (auto)covariance function divided by the variance of the series:

$$\rho_k = \frac{\mathrm{cov}(X_i, X_{i+k})}{\mathrm{var}(X)} = \quad (k = \cdots, -1, 0, 1, \cdots),$$

where the index k is called the *lag* and the covariance is by definition

$$\mathrm{cov}(X_i, X_{i+k}) = E\big((X_i - \bar{X})(X_{i+k} - \bar{X})\big) .$$

Since the covariance at lag 0 is equal to the variance, the autocorrelation coefficient at lag 0 for any series is always 1.

It is clear that a sequence of interrival times all smaller or all larger than the mean will result in a positive contribution to the autocorrelation coefficients. Since it is quite common in packet arrival processes to find relatively extended sequences of short or long interarrivals, we would expect to find positive correlation coefficients.

It is rather difficult to establish confidence intervals for the series of autocorrelation coefficients. We can, however, define the y-axis interval within which, with some probability α, all correlation coefficients should lie were successive elements of the series mutually independent. If a significant number of correlation coefficients is found outside this interval, we will consider the series correlated and discard the assumption of independence.

If a sequence is sampled from a random process so that the samples are independent and identically distributed, all autocorrelation coefficients should be 0 except for the coefficient at lag 0, which is always 1. In this case, Kendall and Stuart [45] show that the mean and the variance of the ρ_k's are

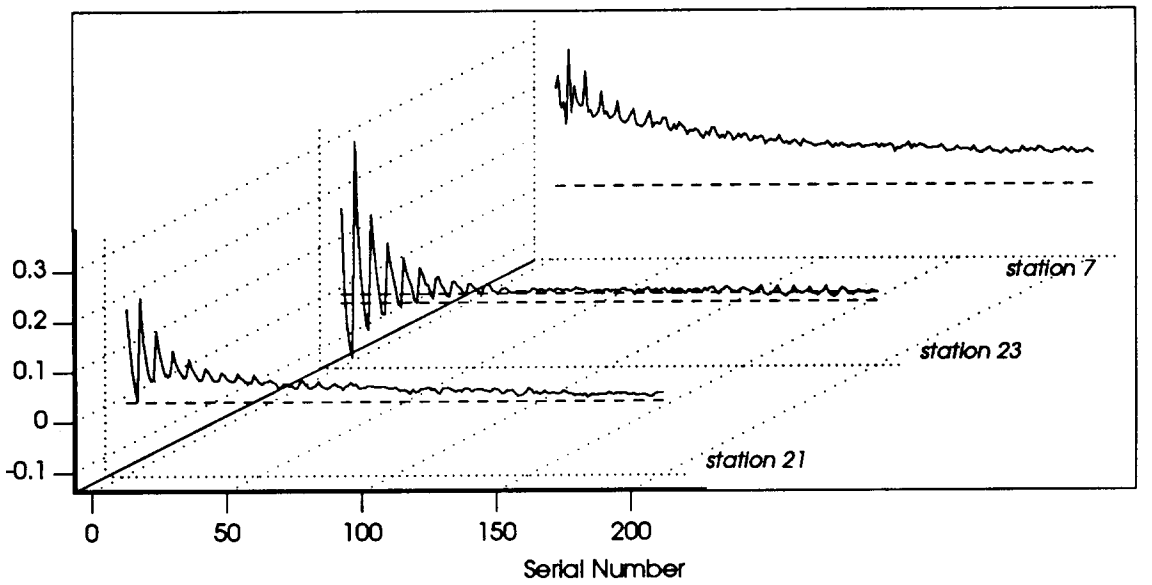
$$E(\rho_k) \approx -\frac{1}{N} \quad \text{and} \quad \mathrm{var}(\rho_k) \approx \frac{1}{N} ,$$

where N is the cardinality of the series. They also show that, under weak conditions, ρ_k is asymptotically normally distributed. Therefore, the $100(1-\alpha)$ confidence limits for the autocorrelation coefficients are $-1/N \pm z_{\alpha/2}/\sqrt{N}$, where $z_{\alpha/2}$ is such that, for the standard normal distribution function, $\Pr(|Z| > z_{\alpha/2}) = \alpha$.

In Figure 3.14 we show the series of autocorrelation coefficients (sometimes called *correlogram*) for the send queues of three file servers, *station 21*, *station 23*, and *station 7*. In this figure we have drawn as dashed lines the confidence intervals that allow us to reject the assumption of independence with a 99 percent certainty. While the correlograms of *station 21* and *station 7* are always positive (accordingly, we have only drawn the upper limit of the confidence bands), *station 23*'s correlogram has some negative values. This is an indication (even after the truncation of interarrival times) that the series contains some sequences of alternating short and long interarrival times, which are often artificially produced by the software monitoring tools that we mentioned in Section II and are revealed only when users are idle. We could remove from the series such sequences, but we are reluctant to manipulate the data more than we have already done. We will keep in mind, however, that some interarrival time series, such as that of *station 23*, contain sequences that may not be fully representative of user generated busy periods.

In all three workstations there are periodicities that generate small spikes at lag values multiple of six. If we recall the discussion of Figures 3.10 and 3.11 in Section IV, in which we showed that the interarrival times classified as "LastFragment/1500" were more or less uniformly distributed, we realize that in groups of successive 8-Kbyte transfers with high probability the "LastFragment/1500" interarrivals are larger than the others generated by the message fragmentation. Thus, in sequences of successive 8-Kbyte

FIGURE 3.14. AUTOCORRELATION COEFFICIENTS FOR THE SEND QUEUES OF THREE FILE SERVERS



message transfers, we have longer interarrival times spaced exactly six interarrivals apart. The impact of these is reflected upon the shapes of the correlograms. Since there are fewer and fewer of those groups, as the number of successive groups increases, the size of the periodic spikes decreases as the lag increases.

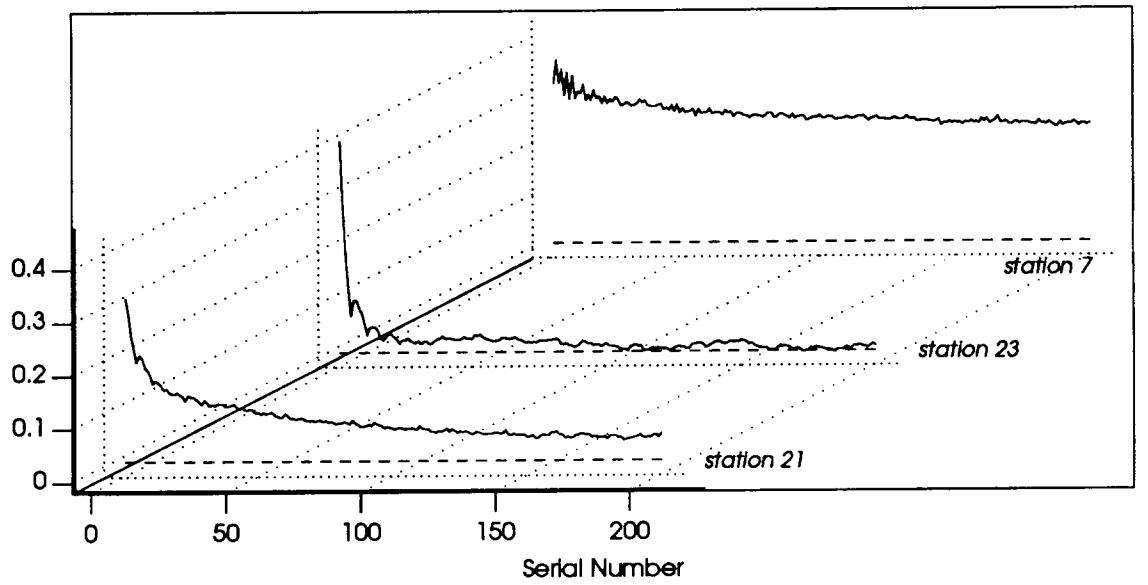
As expected, the series of correlation coefficients decreases towards zero as the lag grows. Although in the cases of *station 21* and *station 23* the coefficients approach levels indicating that after some delay the correlation effects are lost, *station 7* shows a significant long-term correlation. Since all machines run the same type of software and communicate using the same protocols, we attribute *station 7*'s long-term correlation to nonstationary components in its arrival process.

Finally, note that, had we computed the correlograms using all interarrival times, including those larger than 80 milliseconds, the details that emerge in these graphs would have disappeared, overwhelmed by the dominant nonstationary components.

In Figure 3.15 we show the correlation coefficients of the receive queues for the same three servers. Three observations deserve to be made. First, unlike the plots of Figure 3.14, those in Figure 3.15 are not spiky. Since the receive queue of a server collects the aggregate traffic from all of the server's clients, various groups of successive 8-Kbyte messages mix together, resulting in randomization of packet interarrival times.

Second, the correlation coefficients of *station 7* decrease towards zero even more slowly than *station 7*'s send queue coefficients, suggesting that nonstationary components are even more dominant here. Table 3.3 indicates why this may be so. According to that table, *station 7* receives 20 percent fewer packets and 50 percent fewer bytes than it generates. This suggests that *station 7*'s average receive-queue packet is smaller. Smaller packet size in turn suggests that there are fewer long data-transferring packets, which are associated with short interarrival times; thus, the receive queue has fewer short

FIGURE 3.15. AUTOCORRELATION COEFFICIENTS FOR THE RECEIVE QUEUE OF THREE FILE SERVERS



interarrival times than the send queue. We have verified that the fewer long packets are more unevenly distributed in the traces for which we computed the autocorrelation coefficients. This uneven distribution is the source of nonstationarity.

Finally, the short-term correlation of the receive queues is generally higher than that of the corresponding send queues. Such higher short-term correlation is produced by longer sequences of long interarrival times in the receive streams. Table 3.3 indicates that the receive queues of each of the file servers *station 21*, *station 23*, and *station 7* contain a larger proportion of small packets. In each case, the average packet size is smaller in the receive queues than in the send queues. This should be expected, since client workstations have to page in programs, an activity that requires the servers to transmit large packets. Another source of large packets are file-system read and write operations. Since reads, which are transmitted by servers, outnumber writes, which are received, the send queues of servers contain a larger number of large packets.

In Figures 3.16 and 3.17 we show the correlograms for the send and receive queues, respectively, of four client workstations: two diskless ones, *station 13* and *station 25*, and two with disks, *station 8* and *station 11*. *Station 13* and *station 25* display very similar correlation series, characterized by high initial values that subsequently decline steeply. *Station 8* with its local disks generates long sequences of medium-sized, remote-procedure-call packets. These sequences, which produce long interarrival times, occur within the time series in clusters, creating long-term autocorrelation. In *station 11*'s case, we have another indication that the data is highly nonstationary (compare with Figure 3.12 in Section IV).

Interpreting a correlogram is complicated. Often, there are spurious correlations, and the presence of nonstationary data components alters the shapes of the correlation curves. Using as point processes for our autocorrelation study what we have called busy

FIGURE 3.16. AUTOCORRELATION COEFFICIENTS FOR THE SEND QUEUES OF FOUR CLIENT WORKSTATIONS

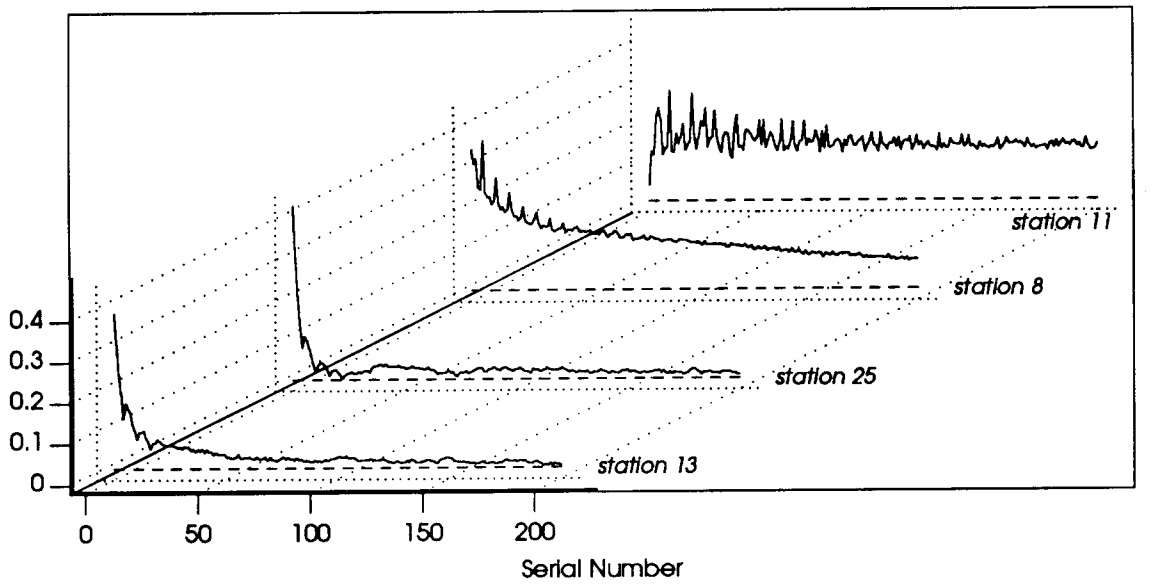
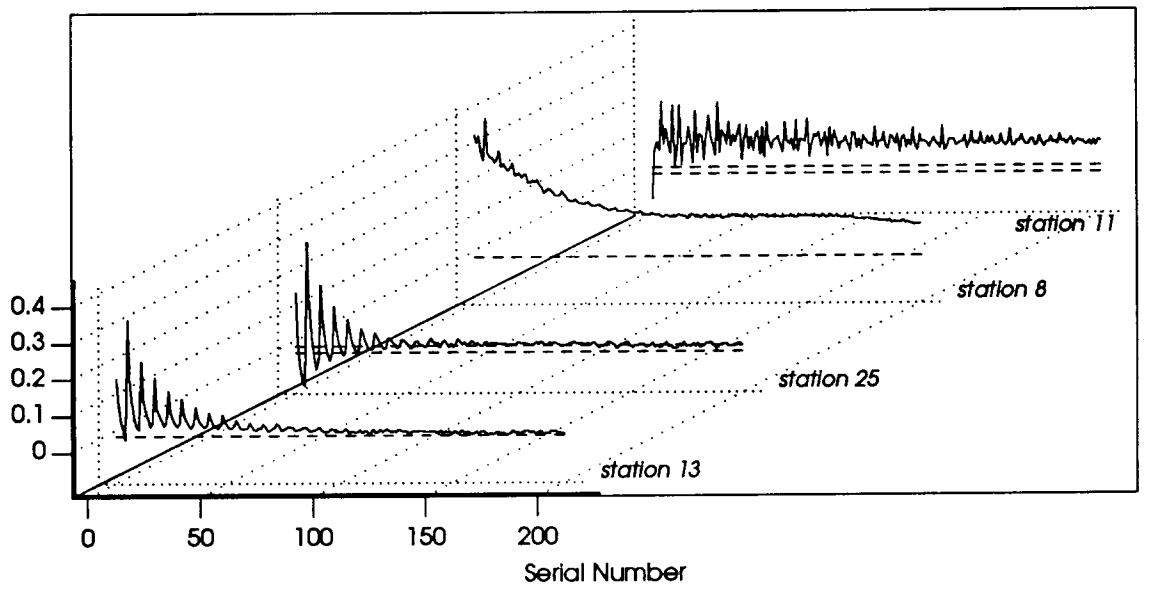


FIGURE 3.17. AUTOCORRELATION COEFFICIENTS FOR THE RECEIVE QUEUES OF FOUR CLIENT WORKSTATIONS



periods has simplified the problem of dealing with "real-life" processes, for many packet arrival processes (notably most diskless workstation processes) become stationary. Others are not quite stationary, but in most cases the range of "variability" has been reduced. Although we could have used a more aggressive technique to remove nonstationarities, our method has the fundamental advantage of not transforming the timing relationships between successive arrivals, which allows us to relate more directly interarrival times to protocol features. In Chapter 4, we will discuss a new way of looking at autocorrelation

coefficients through indices that we will associate with the variability of packet arrival processes. We will also continue there our discussion of the effects of nonstationarity.

VIII. Concurrency on the Network

Analysis of the behavior of single workstations must take into account interactions between machines, if any. Without a clear understanding of such interactions, models of packet arrival processes cannot be adjusted to encompass network configurations other than the one studied. There are three major sources of interaction among workstations: first, direct communication among them, since the protocols used for this communication are request-response; second, concurrent access to the same file server; and third, serialization in the network. During all of these types of interaction there is simultaneous activity by two or more workstations. Thus, we will begin by looking at the number of concurrently communicating machines. In the next section, we will consider the effects of shared resources.

The histograms in Figure 3.18 show the number of 1-second intervals during which a given number of workstations were active. The chosen interval length of 1 second is short enough to be meaningful for the queuing behavior of transmit and receive queues, yet long enough to allow time for the statistical interaction among machines to develop. Figure 3.18 is built from a subset of all packets recorded in our traces; in order to compare more directly this figure to the following two figures, we consider only those machines in the Sun Engineering network that communicate with the six major file servers: *station 5, station 7, station 9, station 21, station 23,* and *station 28.* Between them, these machines account for 82.4 percent of the total traffic. (The remaining 17.6 percent is primarily traffic between pairs of clients as well as a small amount of internetwork traffic.)

FIGURE 3.18. HISTOGRAMS OF THE NUMBER OF 1-SECOND INTERVALS IN WHICH WORKSTATIONS WERE ACTIVE

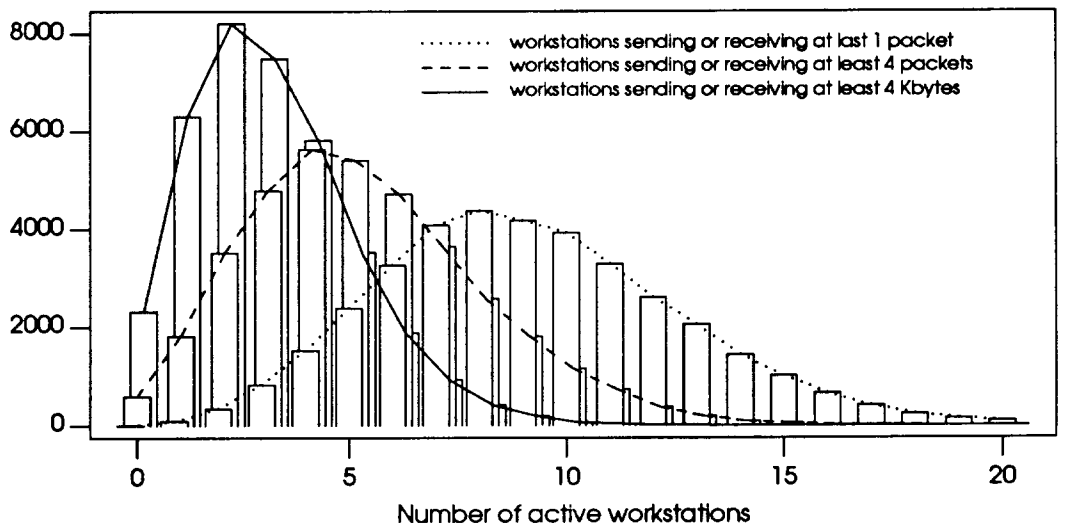


Figure 3.18 consists of three separate sets of histograms, reflecting three different criteria for defining an "active" workstation. In the first set, a workstation was considered active if it sent or received at least one packet during a 1-second interval; in the second

set, if it sent or received at least four packets during a 1-second interval; and in the third set, if it sent or received at least 4 Kbytes during a 1-second intervals. The second and third tests were devised to exclude workstations that transmit only occasionally or periodically but infrequently. Interpolation curves for the three sets have also been drawn. The median for the curve for the first set (indicated by a dotted line) is between 9 and 10 workstations; for the second set (indicated by a dashed line), between 5 and 6; and for the third, approximately 3.

In Figure 3.19 our unit of study is the individual file server rather than the single workstation. We use the same three criteria for an active machine as in Figure 3.18. Finally, Figure 3.20 presents histograms for workstations in communications with the busiest file server, *station 21*.

FIGURE 3.19. HISTOGRAM OF THE NUMBER OF 1-SECOND INTERVALS SHOWING ACTIVITY OF FILE SERVERS

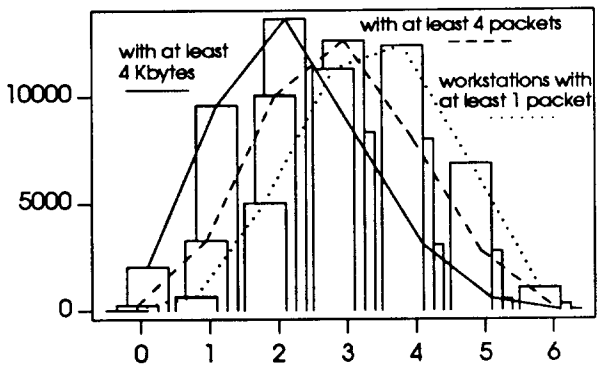
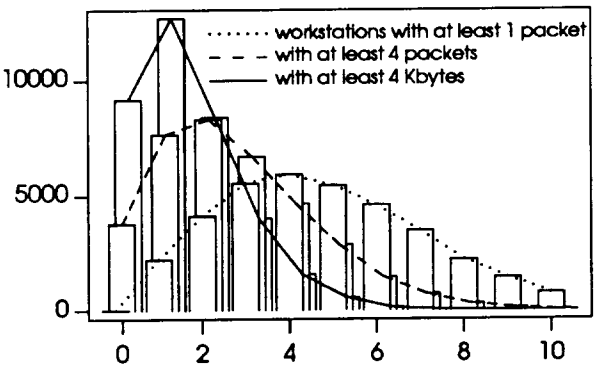


FIGURE 3.20. HISTOGRAM OF THE NUMBER OF 1-SECOND INTERVALS SHOWING ACTIVITY OF BUSIEST FILE SERVER



All these graphs show a consistent view: there is only a moderate amount of interaction among workstations and between workstations and file servers. For instance, notice that the statistical norm in Figure 3.20 in the case corresponding to the criterion that calls for 4 Kbytes to be exchanged before a machine be considered active is 1. Nevertheless, it is necessary to investigate whether this partial interaction is reflected in the second order statistics.

It is also worth estimating the probability that k or more consecutive packets are exchanged between two machines. We know that machines tend to produce bursts of messages, thus, low values for this probability distribution would also indicate that there is network contention.

In Figure 3.21 we display the sample probability that k packets are exchanged (in either direction) between two machines. (Here also we considered only those machines in the Engineering network that communicated with at least one of the six major file servers, that is *station 21, station 7, station 23, station 28, station 5,* and *station 9.*) Of course, for this probability to be meaningful, we have assumed that the distribution is stationary, when in reality one would expect that, for a given value of k, the probability becomes higher as the average network load decreases. If we call this probability $Pr_{AB}(S = k)$, we have

$$Pr_{AB}(S = k) = Pr\left[p_n \in \{A, B\}, p_{n+1} \in \{A, B\}, \cdots, p_{n+k-1} \in \{A, B\}, p_{n+k} \notin \{A, B\} \right]$$

in which p_n identifies the nth packet, and $\{A, B\}$ represents the set of packets exchanged between machines A and B, so that $Pr_{AB}(S = 1)$ is the probability that the next packet is not between A and B. Then,

$$Pr_{AB}(S \geq k) = \sum_{j=k}^{\infty} Pr_{AB}(S = j) = 1 - \sum_{j=1}^{k-1} Pr_{AB}(S = j) \tag{3.3}$$

is the probability that k or more packets are exchanged between two machines. Probability function (3.3) is displayed in Figure 3.22.

FIGURE 3.21. PROBABILITY THAT EXACTLY K PACKETS ARE TRANSMITTED BETWEEN TWO WORKSTATIONS

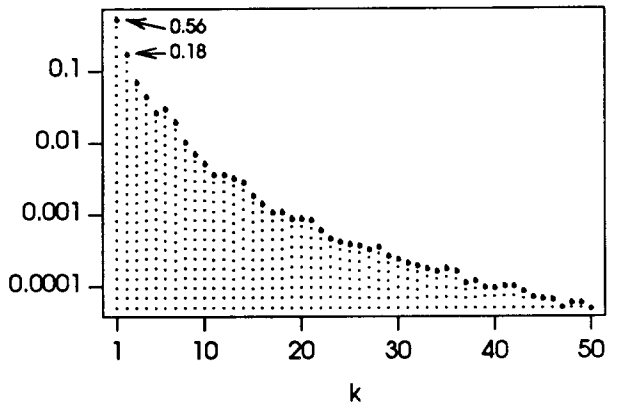
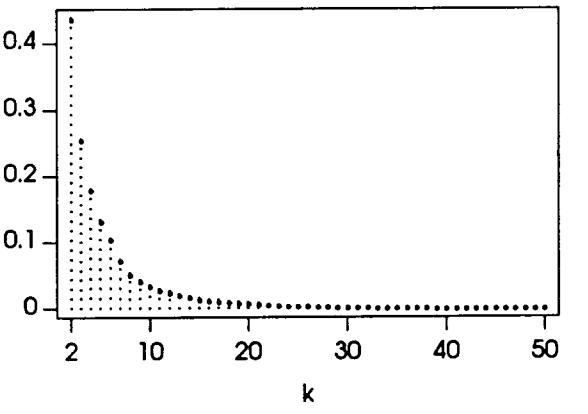


FIGURE 3.22. PROBABILITY THAT K OR MORE PACKETS ARE TRANSMITTED BETWEEN TWO WORKSTATIONS



The relatively low probabilities are additional evidence that some network contention is present. In the following section, we determine whether this network contention creates statistical interaction among workstation arrival processes.

IX. Statistical Interactions Among Workstations

Mutual interactions among personal workstations may have a significant impact on the packet arrival processes that they generate. Personal workstations, as opposed to a distributed system composed of several cooperating but autonomous nodes, which must share access to common data structures as well as other resources, contend with each other for only two types of resources: shared file servers and the shared network (in this context, "network" refers to the set of links, routers, and interfaces that are shareable). Although distributed applications can be built on top of personal workstations, there were few instances of distributed programs in use in the Sun system that we studied; most users accessed their workstations, the file servers, and the other resources independently of each other. Thus, interaction caused by mutual exclusion synchronization of distributed programs is not relevant. Rather, interaction arises from contention over file servers and the network. Both sources of interaction will be examined in this section.

Packet transmissions from autonomous machines originate independently of each other. They will, however, be serialized on the network whenever two stations begin to transmit simultaneously. Serialization obviously creates a causal relationship between

successive arrivals. In addition, concurrent file server accesses are related by sharing of the server's CPU and disks. Yet, it is necessary to determine whether serialization and sharing result in significant interaction among workstations.

The technique we have chosen for capturing and describing possible interaction among machines is the analysis of the cross covariance (in the form of cross correlation) of various packet arrival processes detailed below. The covariance between two series X_i and Y_i of means μ_x and μ_y respectively is defined by

$$\text{cov}(X, Y) = E((X - \mu_x)(Y - \mu_y)) . \tag{3.4}$$

Since the covariance depends on the units in which X and Y are measured, we will use a normalized version of the covariance, the correlation coefficient, obtained by dividing Equation (3.4) by the product of the standard deviations of X and Y. It can be easily proven that the correlation coefficient between two series is a number between -1 and $+1$. Finally, we observe that the correlation coefficient at lag 0 between two (distinct) random series, unlike the autocorrelation for which the coefficient at lag 0 is always 1, will be in general different from 1.

The packet arrival processes that we will analyze fall into two basic categories, which we define as *opposite queues* and *parallel queues*. Opposite queues are a pair of arrival processes one at a send queue and one at a receive queue; parallel queues refer instead to either two send or two receive queues. Opposite queues can be classified into four types according to the machines they are associated with: 1) a single workstation (i.e., both the send and receive queues belong to the same workstation); 2) one client and one server; 3) two clients on different file server clusters; and 4) two independent servers.

In the single workstation case, correlation is to be expected: the nature of the transmission sent to the server determines the content and size of the response and the time it is sent. In the second case we are not sure whether to expect a causal relationship because several different clients contribute to the queue of the server. The third case considers whether or not there is correlation between opposite queues of two clients that use different file servers, allowing us to isolate the effects of the network on correlation. Finally, the fourth case, like the third, in which we consider two independent servers, helps to detect causation attributable to the network. In this last situation, we would expect a stronger causal relationship for two reasons: first, servers are more heavily used than clients, and, second, clients in one cluster often mount file systems from a server from another cluster of workstations. As a result, a client that loads a file from the second server in its memory may need to flush some of its memory pages to the swapping area, which is managed by the first server. Thus, in this example, there is a causal relationship between the send queue of the second server and the receive queue of the first.

Parallel queues can be classified into three types according to the relationship between the two machines to which the queues belong: 1) two clients from the same cluster; 2) two clients from two different clusters; and 3) two servers. Studying parallel queues is important because they replicate what occurs when processes are superimposed. In the case of two clients from the same cluster, we examine the interactions at the server and on the network of the two send (or two receive) queues. In contrast, in

the case of two clients from different clusters, the queues' sole interaction occurs on the network; thus (as in case three and four for opposite queues), we can isolate the effects of the network interaction. Finally, it is also important to see how two servers behave with respect to the superposition of their queues.

We will study both opposite and parallel queues in terms of packet counts and interarrival times. Note that we cannot use subsets of measured series for these calculations. Since we are trying to see evidence of temporal cause/effect phenomena, we cannot remove a portion of the measured arrivals, as we did in the study of the autocorrelation coefficients. Instead, for each pair of arrival processes, we will include all arrivals in the time series.

Packet counts are computed by choosing a constant interval with which to divide the time axis, and counting how many arrivals occur within each interval. The choice of the interval over which to compute the packet counts is important and represents a trade-off between computational cost and accuracy: small intervals will increase cost, while large intervals will result in loss of burstiness information. In line with Cox and Lewis [19], we found that choosing the number of intervals roughly equal to the number of interarrival times gives good results. For the analysis in the rest of this chapter we have used intervals of 50 milliseconds for computing packet counts.

In the correlation computations, we use arrival streams generated by nine machines: three file servers—*station 21*, *station 23*, and *station 7*—and six clients. *Station 25* and *station 14* are clients of *station 23*. *Station 27*, *station 3*, and *station 13* are clients of *station 21*. *Station 11* is a client of *station 7*. *Station 25*, *station 14*, *station 3*, and *station 13* are diskless and thus depend on their servers for obtaining, in addition to general file services, the object code of programs that are executed, and swap space.

In all figures below, except for Figure 3.23, in which we needed more resolution, we show the first 200 correlation coefficients. The notation "*station 3->station 21*" will be used to indicate the correlation curve between *station 3*'s send queue and *station 21*'s receive queue. Analogously, with the notation "*station 3–station 21*" we will indicate *station 3*'s and *station 21*'s parallel queues. Notice that we cover all cases: if there is a "*station 3->station 21*" curve, there will also be the "*station 21->station 3*" one. In each figure, we indicate with dashed lines the 95 percent confidence intervals for the null hypothesis that packet counts (or packet interarrival times) are not correlated (refer to Section VII for more details on the meaning of these confidence bands).

A glance at Figures 3.23 through 3.32 quickly reveals that all the queues, with the noticeable exception of those in Figure 3.23 and those in Figure 3.25, appear to be uncorrelated. There are several spurious correlation coefficients (for instance in the graphs of Figures 3.28 and 3.29), which are generated by the few large sample points present in the series. In addition, although not every autocorrelation curve is within its the confidence interval, in almost all cases the coefficients are very small in magnitude and display random fluctuations. Only opposite queues for packet counts of type 1, i.e., belonging to the same workstation (in Figure 3.23), and opposite queues of type 2 for packet counts (in Figure 3.25) show noticeable correlation. (In type-2 opposite queues, in which we correlate a queue of a client and that of its server, the correlation is not as pronounced as in type-1 queues, but we have indicated earlier that, since many clients contribute to the queue of the server, the causal effects may be reduced.) The

FIGURE 3.23. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: PACKET COUNTS SEND AND RECEIVE QUEUES ON SAME WORKSTATIONS

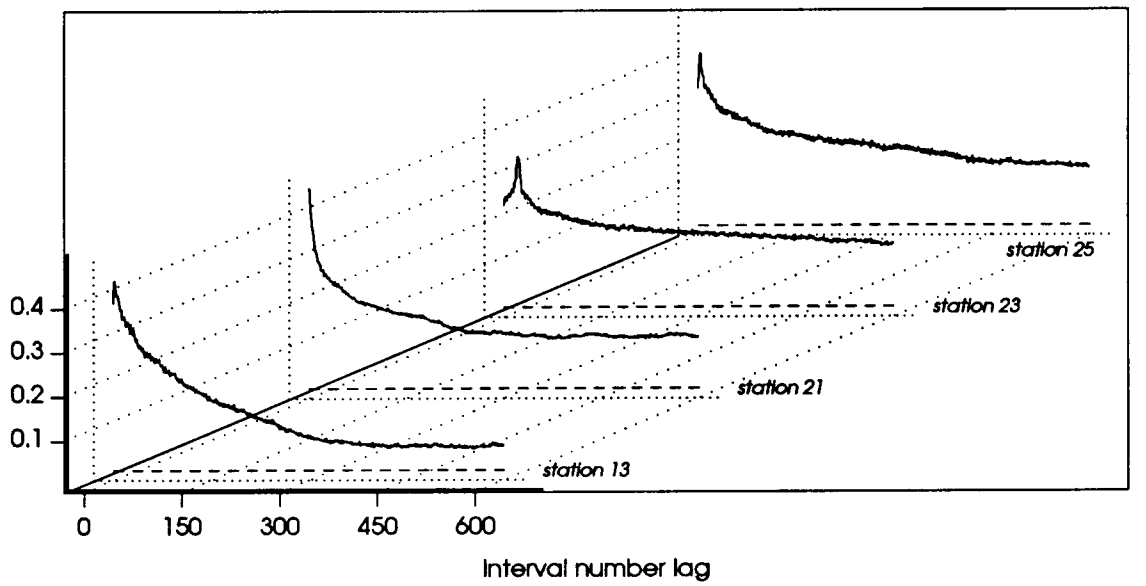
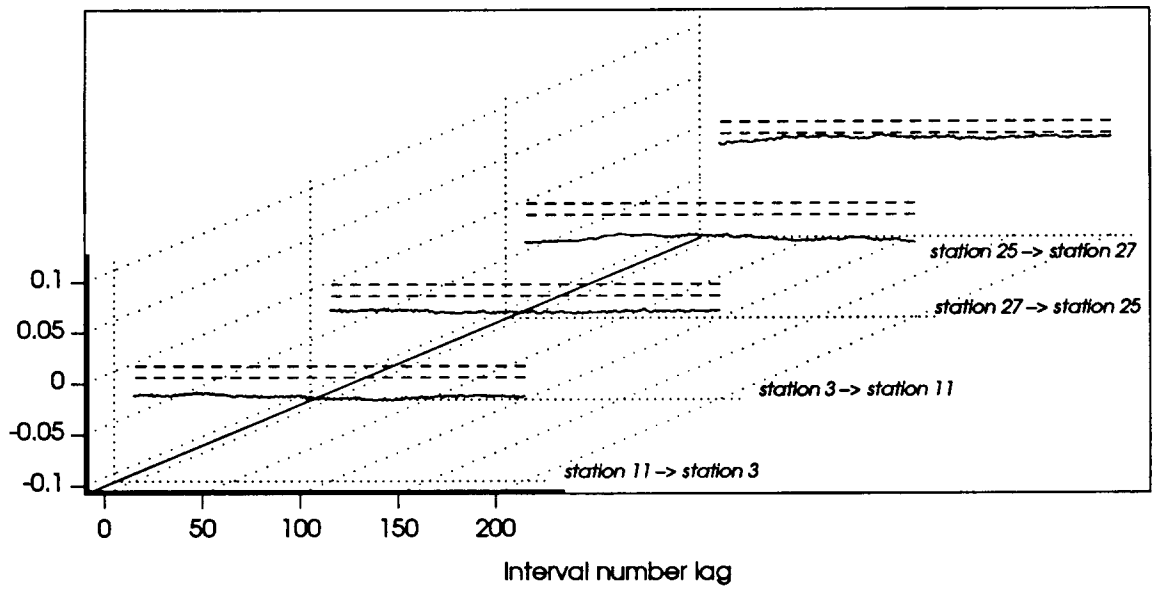


FIGURE 3.24. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: PACKET COUNTS SEND AND RECEIVE QUEUES ON CLIENTS OF DIFFERENT CLUSTERS



association detected between the send and receive streams of a workstation is attributable to the use, by many applications, of remote procedure call protocols, which suspend themselves until an answer to the sole outstanding message is received, resulting in interlocks between send and receive operations. In addition, since in systems that use request-response protocols operations are typically initiated by clients (and not by servers), pauses in a user's activity that result in long interarrival times in the send stream

FIGURE 3.25. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: PACKET COUNTS SEND AND RECEIVE QUEUES ON A CLIENT WORKSTATION AND A FILE SERVER

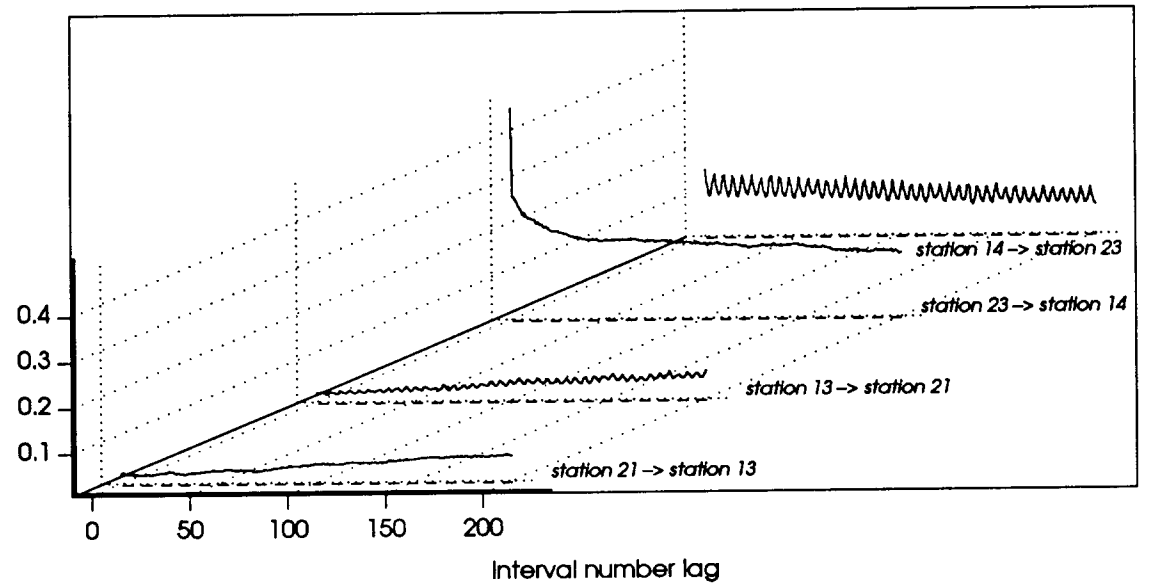
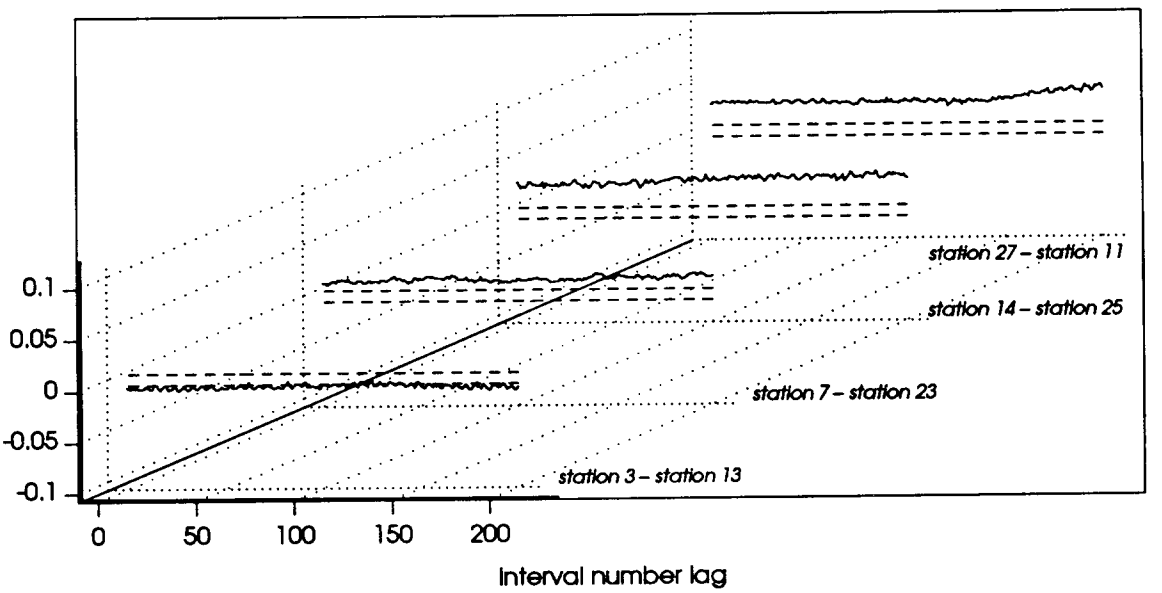


FIGURE 3.26. CROSS CORRELATIONS BETWEEN PARALLEL SEND QUEUES: PACKET COUNTS



are almost invariably associated to pauses in the receive streams.

In all other cases, if any causal relationships do exist, they are overwhelmed by the large variability of the arrival processes. In fact, it is the greater variability of interarrival time processes in comparison to that of packet count processes (as we have observed earlier, the range of the probability distribution functions for packet counts is rather limited) that washes out the correlation in Figure 3.28, which displays the cross correlations

FIGURE 3.27. CROSS CORRELATIONS BETWEEN PARALLEL RECEIVE QUEUES: PACKET COUNTS

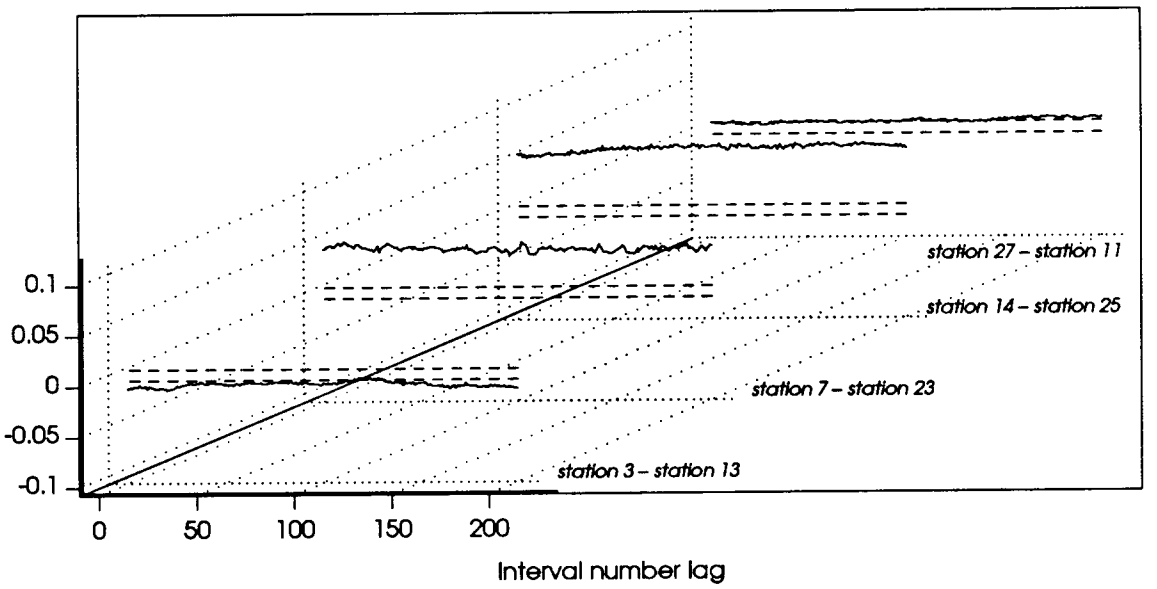
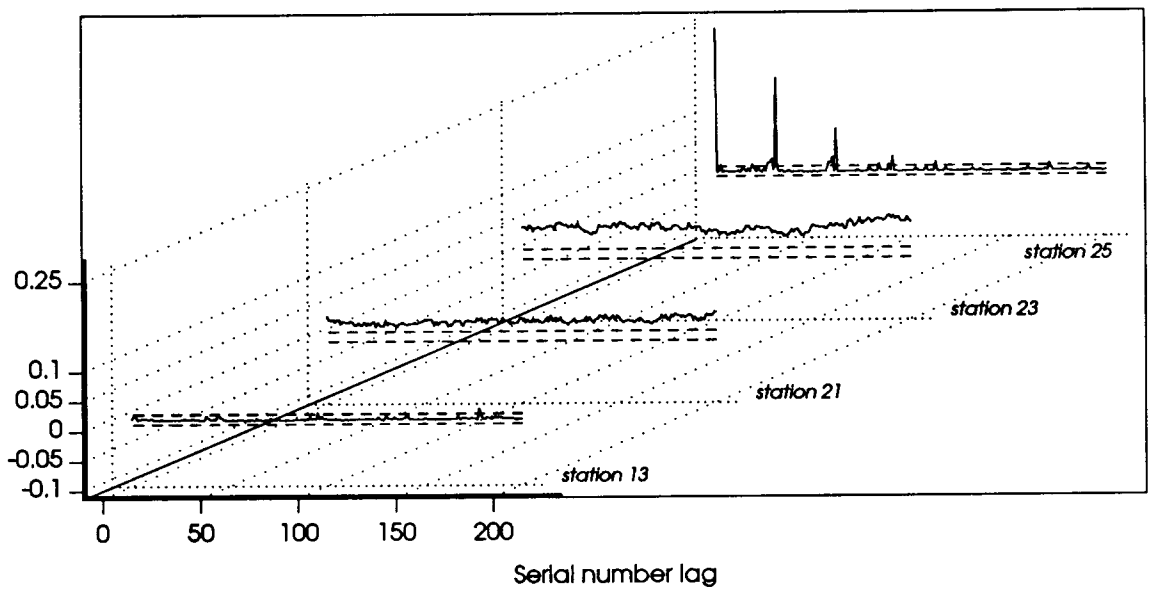


FIGURE 3.28. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: INTERARRIVAL TIMES SEND AND RECEIVE QUEUES ON SAME WORKSTATIONS



between opposite queues on the same workstation for interarrival time processes.

X. Summary

In this chapter we have analyzed the properties of the traffic generated and received by single client and file server machines. From the prospective of a macroscopic time scale, workstations produce network loads that can be divided into light (or

FIGURE 3.29. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: INTERARRIVAL TIMES
SEND AND RECEIVE QUEUES ON CLIENTS OF DIFFERENT CLUSTERS

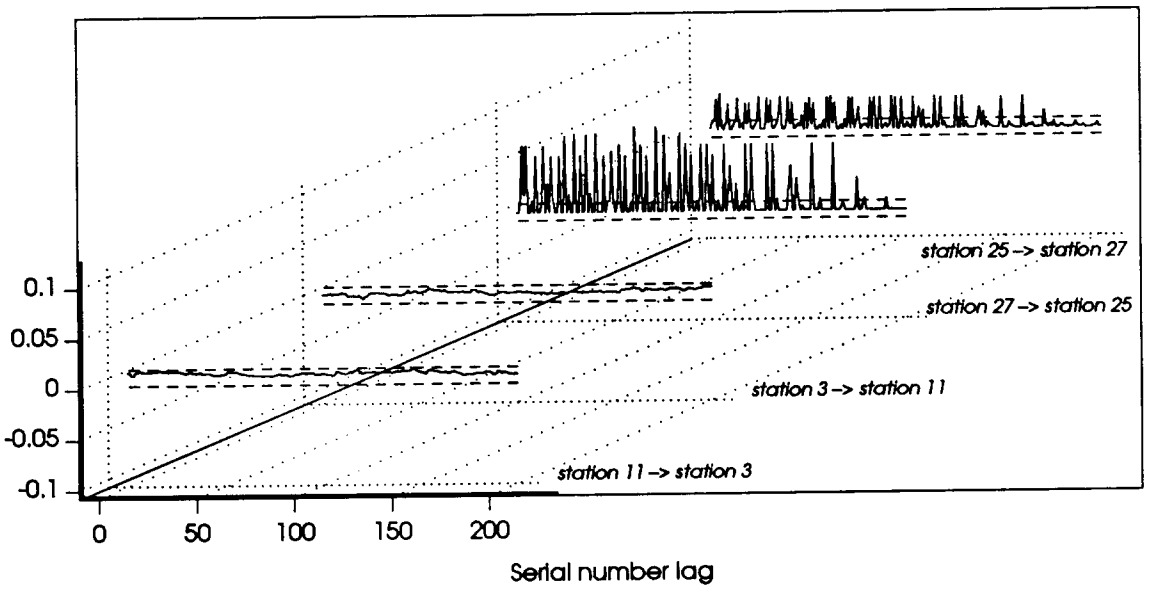
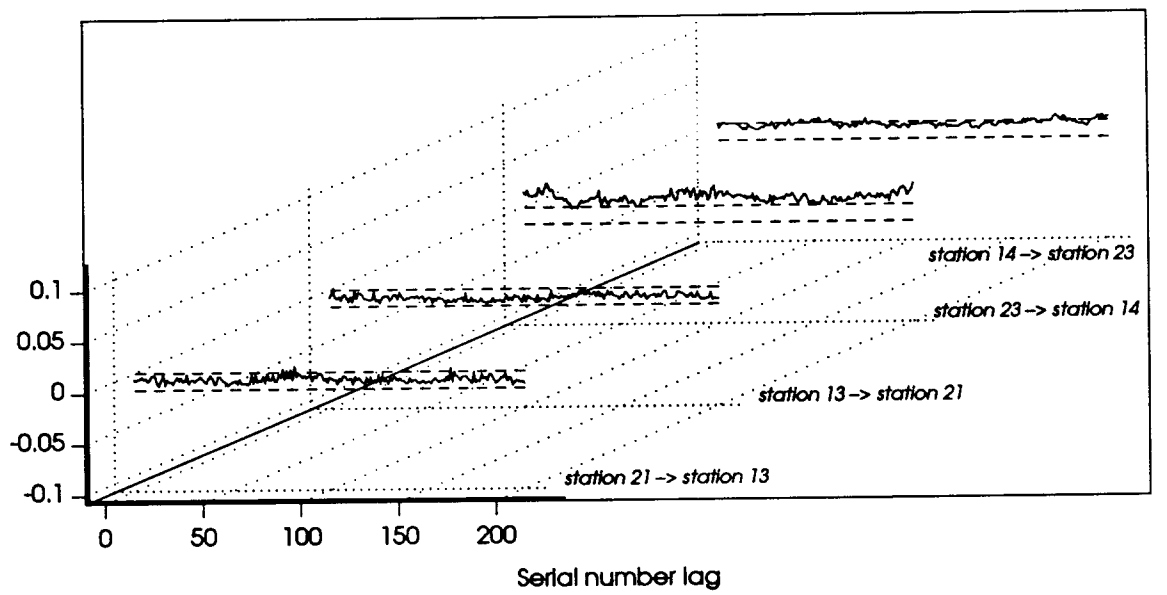


FIGURE 3.30. CROSS CORRELATIONS BETWEEN OPPOSITE QUEUES: INTERARRIVAL TIMES
SEND AND RECEIVE QUEUES ON A CLIENTS WORKSTATION AND A FILE SERVER



nearly zero) and heavy phases. We argued, in Section III, that traffic models of variability need only be concerned with what we have termed "busy" periods: intervals of time during which the arrival process produces a "substantial" arrival rate. In Section VI, we then considered what constitutes a substantial arrival rate. We concluded that the busy period should be defined as the interval of time during which interarrival times are no longer than 80 milliseconds.

FIGURE 3.31. CROSS CORRELATIONS BETWEEN PARALLEL SEND QUEUES: INTERARRIVAL TIMES

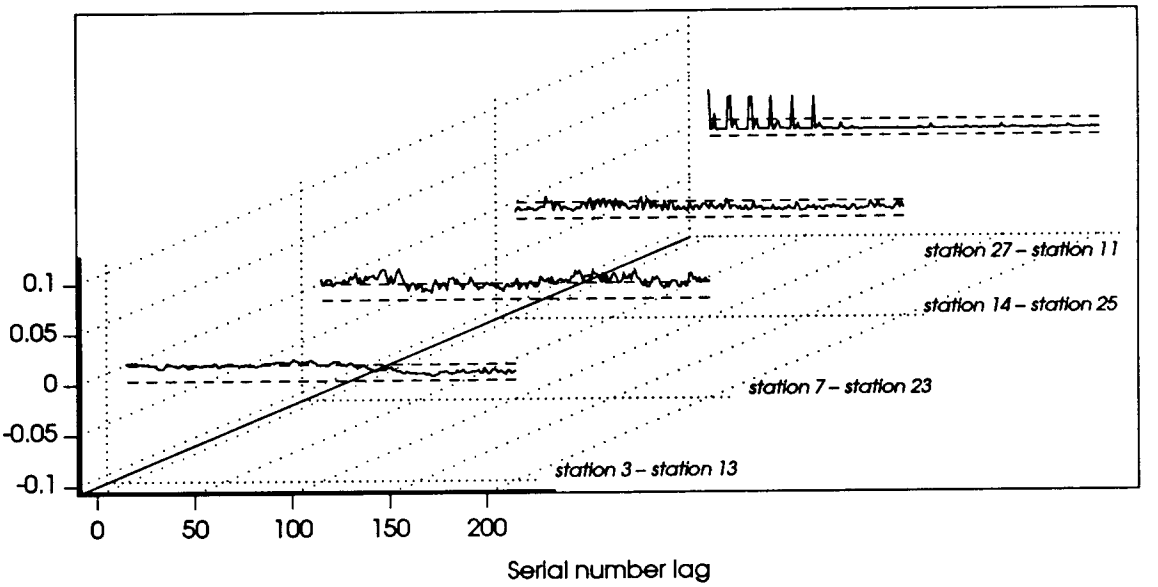
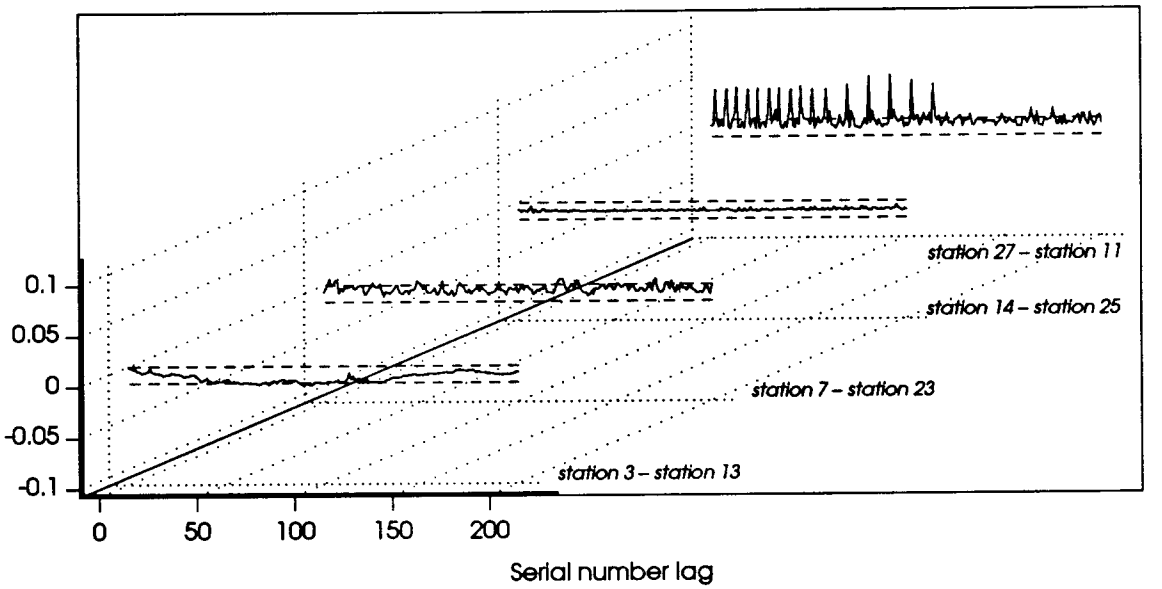


FIGURE 3.32. CROSS CORRELATIONS BETWEEN PARALLEL RECEIVE QUEUES: INTERARRIVAL TIMES



The major reason for focusing on busy periods rather than on the entire sequence of interarrival times is that busy periods are probably more stationary than the full series. However, upon closer examination, we discovered that, although in most cases busy periods appeared to be weakly stationary processes, there were also some that were clearly nonstationary.

We looked at histograms of interarrival times, which are related to the marginal probability density functions of interarrival times. The decomposition of the histograms by packet-length classes has shown that short interarrival times are produced by long packets, which are transmitted in quick sequence by the IP fragmentation layer. Sequences of shorter packets, produced individually by the NFS protocol layer, tend to generate long interarrival times. Typically, packet arrival processes from file servers and diskless machines are more stationary than those from machines with local disks. Whereas in file servers and diskless machines short interarrival times are more uniformly distributed because of the high communication demands of these machines, in machines with local disks we often observe long stretches of medium-sized packets and long interarrival times, which produce appreciable changes in the characteristics of the packet arrival processes.

We studied the autocorrelation structure of packet arrival processes during busy periods. Often, interpreting the series of correlation coefficients is not simple, but, whenever possible, we have tried to relate the shape of the series to protocol and system features.

We then considered the interactions among various types of packet arrival processes, and found that the only non-negligible statistical dependence occurs between the send and receive processes produced by the same workstation.

4 Indices of Dispersion

Since the first analyses of computer traffic in the mid- and late 1970s [47,80], which showed that packet arrival processes are highly variable, researchers have frequently described computer communication patterns as "bursty". Yet few have bothered to define burstiness. Most seem to invoke the term bursty when confronted with processes whose interarrival time distributions show greater variability than Poisson processes. (Exceptions do exist; see for instance [62], in which a distribution with coefficient of variation 0.3 is termed bursty.) The vagueness surrounding the concept of burstiness stems both from its use to denote different types of variability in many disparate situations and from the difficulty of characterizing in meaningful ways the capricious nature of packet arrivals.

The variability of packet arrival processes is strikingly manifested in the following Figures 4.1 and 4.2, which represent respectively the times between subsequent arrivals and the times between every four arrivals for the messages sent by a single-user workstation to its file server over a local-area network. In each figure, the logarithm of the interarrival time is on the ordinate and the serial number on the abscissa, and artificial zero lines have been placed at the median values of the series. This graphical arrangement allows us to view short interarrival times, which would otherwise be obscured by long ones: the former are printed toward the bottom of the graph and the latter toward the top.

FIGURE 4.1. TIMES BETWEEN SUCCESSIVE ARRIVALS

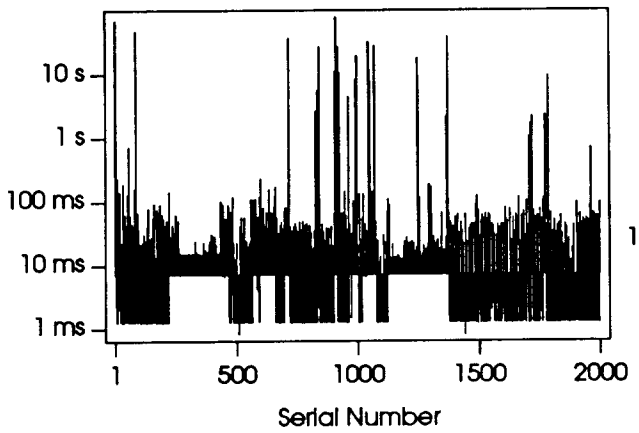
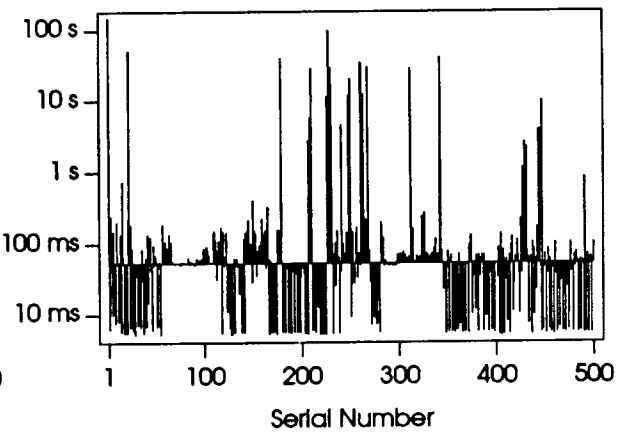


FIGURE 4.2. TIMES BETWEEN EVERY FOUR ARRIVALS



The variance of the interarrival times in Figure 4.2 is about six times larger than the variance of those in Figure 4.1. Clearly, the larger dispersion of values in the second figure stems from the clustering of small and large interarrival times in separate groups in the first figure. This bunching is caused by protocol features, such as fragmentation of large messages, that generate short interarrivals in batches. Similarly, a sequence of related remote procedure calls may generate a group of longer interarrivals because, for each request, the destination has to execute a procedure before returning the answer, which in turn will trigger the next request. Thus, the increase in variance is related to the temporal structure in the data and is not captured by simple burstiness indices such as the coefficient of variation, the peak-to-average ratio, an so on. In this chapter we will use the term variability to refer explicitly to changes in the variance of the sum of consecutive interarrivals or in the variance of arrival counts over larger and larger time intervals.

Variability in packet arrivals has been connected to the queuing delays packets are subject to: the general rule is that more variability corresponds to longer delays. However, except in a few simple cases, the precise relationship between variability and queuing delays is difficult to represent analytically. Several attempts to resolve this issue have been made; in particular, Fendick and Whitt's approach [26], which uses a statistical index that models the variability of the arrival process but also captures the dependency between the interarrival and the service times, is worth mentioning.

In this chapter we take a more narrow, focused approach: we characterize the variability of measured packet arrival processes with indices of dispersion functions and discuss the merits of these indices as well as the pitfalls of their indiscriminate use. Indices of dispersion have long been known in the statistics community as a powerful tool in the analysis of the second-order properties of point processes [11, 19], but, despite the flourishing in recent years of measurements and analyses of computer traffic data (for a survey see [65]), they have been rarely, if ever, applied to computer traffic measurements. Here, we demonstrate that indices of dispersion are valuable and valid tools for characterizing the variability of packet arrival processes. We also discuss how standard analytical models should be fitted to traffic measurements in order to take into account the variability of the data.

This chapter is organized into three major parts. In Section II, we define the index of dispersion for intervals and the index of dispersion for counts, and review their basic properties. We then calculate one of these two indices for each of three classes of analytical models that are often used to represent bursty point processes: renewal models with hyperexponential interarrival times, batch Poisson processes, and Markov-modulated Poisson processes. Although the results are in most cases not new, the exercise serves the important purpose of clarifying the meaning and use of indices of dispersion. In Section III, we estimate the indices of dispersion for several measured packet arrival processes generated by single-user workstations communicating with file servers over a local-area network. We show how nonstationary data introduce difficulties, and suggest that semi-Markov models may model accurately both short- and long-term variability. Finally, in Section IV, as an example of how standard arrival models can incorporate the variability that we have analyzed, we develop a procedure to fit a Markov-modulated Poisson process to our arrival processes.

II. Indices of Dispersion

A. The Index of Dispersion for Intervals

Let us consider first describing point processes in terms of the lengths of the intervals between subsequent arrivals. For packet-arrival processes, these intervals, which we will call *interarrival times*, are defined as the length of time between the beginning of the transmission of a given packet and the beginning of the transmission of the previous packet. (Cox and Miller [18, page 339] define the same quantity more esoterically as *backward recurrence-time*.) Notice that under this definition the transmission time of the previous packet is included in the interarrival time; thus, for networks that allow variable packet sizes, this definition introduces a source of dependency between intervals and packet lengths. (A constant packet size imposes a lower bound on interarrival times, but not the dependency.)

The variance of the sum of two random variables depends on the covariance between them, and, if they have common variance, is given by

$$\text{var}(X_{i+1} + X_{i+2}) = 2\text{var}(X) + 2\text{cov}(X_{i+1}, X_{i+2}) ,$$

and, in general, for the sum of n variables we have

$$\text{var}(X_{i+1} + \cdots + X_{i+n}) = n\,\text{var}(X) + 2\sum_{j=1}^{n-1}\sum_{k=1}^{j}\text{cov}(X_j, X_{j+k}) . \tag{4.1}$$

We have indicated with var(X) the common variance of the X_i (we will also write $E(X)$ for the common mean), and thus have assumed implicitly that the processes under consideration are at least weakly stationary, i.e., that their first and second moments are time invariant, and that the autocovariance series depends only on the distance k, the *lag*, between samples: $\text{cov}(X_i, X_{i+k}) = \text{cov}(X_j, X_{j+k})$, for all i, j, and k.

It is the dependency on the autocovariance, or, equivalently, on the autocorrelation, that makes the variance of the sum of intervals useful in describing arrival processes. In fact, in situations like those of Figure 4.1 and Figure 4.2, in which interarrivals smaller than the mean as well as interarrivals larger than the mean are grouped together, the covariance will assume positive values.

We will use the variance above, normalized by the factor $n\,E^2(X)$, as a measure of the variability of packet arrival processes. The sequence of values

$$J_n = \frac{\text{var}(X_{i+1} + \cdots + X_{i+n})}{n\,E^2(X)} , \tag{4.2}$$

with $n = 1, 2, \cdots$, is called *index of dispersion for intervals* (IDI).

Notice that J_1 is $C_J^2 = \text{var}(X)/E^2(X)$, the squared coefficient of variation for intervals. As a result of the normalization, for a Poisson process J_n has constant value 1 for all n; for a renewal process, whose interarrival times are identical and independently distributed (i.i.d.), J_n is also a constant in n of value C_J^2. Using equation (4.1) and the definition $\rho_n = \text{cov}(X_i, X_{i+n}) / \text{var}(X)$, we can express relation (4.2) in terms of the autocorrelation coefficients at lag n:

$$J_n = C_J^2\left[1 + 2\sum_{j=1}^{n-1}(1 - \frac{j}{n})\rho_j\right] , \tag{4.3}$$

which shows that point processes with positive correlation coefficients have monotonically increasing IDI curves. Notice also that the limit of equation (4.3), when it exists, is proportional to the sum of all correlation coefficients (plus 1), that is,

$$\lim_{n\to\infty} J_n = C_J^2 \left[1 + 2\sum_{j=1}^{\infty} \rho_j \right] . \tag{4.4}$$

The asymptote in the limit above depends on the sum (integral) of all the correlation coefficients. Since the interarrival times typically become statistically independent as the lag increases (the causation effects triggered by a packet transmission, such as additional queuing and increased disk activity, diminish as time increases), making their autocorrelation coefficients decrease to 0, for practical purposes the limit will be reached for a finite value of j. As noted above, packet-arrival processes normally have positive autocorrelation coefficients since both interarrivals shorter than the mean interarrival time and those longer than the mean interarrival time tend to occur in separate bursts. In packet-arrival processes, we would thus expect the IDI sequence to increase with n. Notice that, if the data are not stationary, we can still compute an estimate of J_n; however, equations (4.1), (4.3), and (4.4) are no longer generally valid.

B. The Index of Dispersion for Counts

We can also analyze point processes from the perspective of packet counts–the number of packets in an interval. We can define for packet counts a function similar to the index of dispersion for intervals. The *index of dispersion for counts* (IDC) is the variance of the number of arrivals in an interval of length t divided by the mean number of arrivals in t:

$$I_t = \frac{\text{var}(N_t)}{E(N_t)} , \tag{4.5}$$

where N_t indicates the number of arrivals in an interval of length t. The IDC has been so defined in order that for a Poisson process the value of the IDC is 1, for all t.

In estimating the IDC of measured arrival processes, we will only consider the time at discrete, equally spaced instants τ_i ($i \geq 0$). Indicating with c_i the number of arrivals in $\tau_i - \tau_{i-1}$, we have

$$I_\tau = \frac{\text{var}(\sum_{i=1}^{n} c_i)}{E(\sum_{i=1}^{n} c_i)} = \frac{\text{var}(c_\tau)}{E(c_\tau)} \left[1 + 2\sum_{j=1}^{n-1} (1 - \frac{j}{n}) \xi_j \right] , \tag{4.6}$$

where $\text{var}(c_\tau)$ and $E(c_\tau)$ are the common variance and mean of the c_i's, and ξ_j is the autocorrelation coefficient of the c_i's at lag j.

Notice that, in general, I_t will not be constant for renewal processes, in which counts in disjoint intervals are correlated, save for some notable cases such as the Poisson process. However, observing that the sum of the counts in an interval of size t is less than or equal to k if and only if the sum of k interarrival times is larger than t (assuming that the process has a point at the origin), $\Pr(\sum_{i=1}^{n} c_i \leq k) = \Pr(\sum_{j=1}^{k} X_j > t)$, it can be proved that the limits

of the IDI and IDC are equal: $\lim_{n\to\infty} J_n = \lim_{t\to\infty} I_t$ [20]. While we can always estimate $var(N_t)/E(N_t)$, the representation on the righthand side of equation (4.6) is valid only if the data are stationary. Finally, we observe that both the IDC and the IDI are dimensionless quantities: they do not depend on the dimensions of the variables used in their estimation.

In the next three subsections, we will calculate the IDI for the class of renewal models with hyperexponential interarrival times, the IDC for batch Poisson processes, and present Heffes and Lucantoni's derivation of the IDC for Markov-modulated Poisson processes [35].

C. IDI for Processes with Hyperexponential Interarrival Times

The hyperexponential distribution of order k, H_k, is the weighted sum (mixture) of k exponential distributions:

$$F_{H_k}(t) = \Pr(H_k \le t) = \sum_{i=1}^{k} \alpha_i (1 - e^{-\lambda_i t})$$

with weights $\alpha_i > 0$, satisfying $\sum_{i=1}^{k} \alpha_i = 1$, and rates of the exponential distributions $\lambda_i > 0$. Because it is characterized by a coefficient of variation greater than 1, the hyperexponential distribution is often used to approximate the interarrival-time distribution of bursty processes. In the remainder of this section we will only consider H_2.

The mean of a H_2 distribution is

$$E(H_2) = \mu_1 = \frac{\alpha\lambda_2 + (1-\alpha)\lambda_1}{\lambda_1\lambda_2}$$

and the variance

$$\text{var}(H_2) = \frac{2(1-\alpha)\lambda_1^2 + 2\alpha\lambda_2^2 - ((1-\alpha)\lambda_1 + \alpha\lambda_2)^2}{\lambda_1^2\lambda_2^2},$$

in which we have set $\alpha_1 = \alpha$ and $\alpha_2 = 1 - \alpha$.

It is interesting to study the range of the squared coefficient of variation of intervals for the hyperexponential distribution. This is the constant value of the IDI of a renewal process whose interarrival time distribution is H_2. The coefficient of variation depends on three quantities: α, λ_1, and λ_2; thus, if we choose a value for $\mu_1 = \hat{\mu}$, which we keep constant, to derive

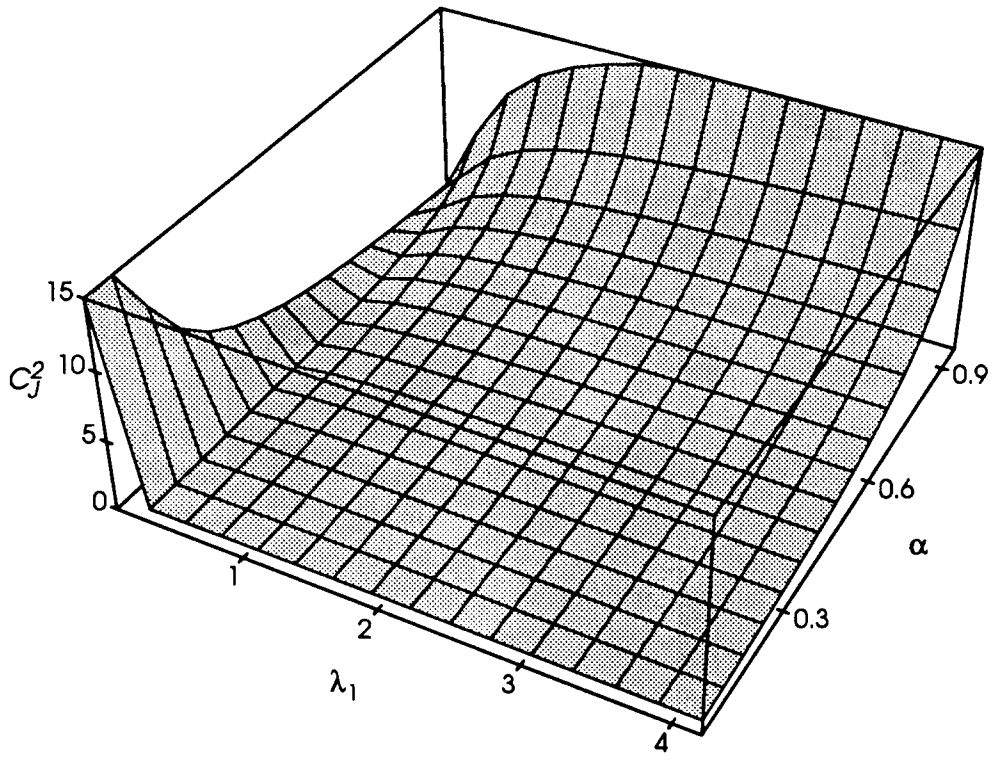
$$\lambda_2 = \frac{(1-\alpha)\lambda_1}{\hat{\mu}\lambda_1 - \alpha},$$

with the constraint $\lambda_1 > \alpha/\hat{\mu}$, we can obtain a formula for the coefficient of variation that depends only on α and λ_1:

$$C_J^2(\alpha, \lambda_1) = \frac{(1+\alpha)\hat{\mu}^2\lambda_1^2 - 4\alpha\hat{\mu}\lambda_1 + 2\alpha}{(1-\alpha)\hat{\mu}^2\lambda_1^2}.$$

In Figure 4.3 we show a three-dimensional plot of C_J^2 for $\hat{\mu} = 5$, which shows that the squared coefficient of variation increases both as α increases to 1 and (when λ_1 approaches $\alpha/\hat{\mu}$) as α decreases to 0.

FIGURE 4.3. SQUARED COEFFICIENT OF VARIATION FOR H_2



Indeed, we have

$$\lim_{\alpha \to 1} C_J^2(\alpha, \lambda_1 = const) = \infty, \quad \lim_{\lambda_1 \to \infty} C_J^2(\alpha = const, \lambda_1) = \frac{1+\alpha}{1-\alpha}, \quad \text{and} \quad \lim_{\alpha \to 0, \lambda_1 \to \frac{\alpha}{\mu}} C_J^2(\alpha, \lambda_1) = \infty,$$

which proves that, for a given value of the mean arrival rate, the variability increases when α approaches the limits of its domain: 0 or 1. But this is inappropriate when the purpose of an approximation based on hyperexponential interarrival times is to model arrival processes markedly different from a Poisson process and with a large coefficient of variation. When α is nearly 0 or nearly 1, for "most of the time", the approximation process has interarrival time exponentially distributed with rate λ_2 or λ_1 respectively.

D. IDC for Batch Poisson Processes

The batch Poisson process is a generalization of the Poisson process in which a random number of simultaneous arrivals, p_i, replaces the original single arrival. The p_i's are i.i.d. and the total number of arrivals in an interval of duration t is $p(t) = \sum_{i=1}^{N(t)} p_i$, where $N(t)$ is the number of original Poisson arrivals. Since the p_i's are independent of $N(t)$, the mean number of arrivals in an interval of size t is clearly $E(N(t))E(p_i) = \lambda\, t\, E(p_i)$, where λ is the arrival rate of the original Poisson process.

In order to compute the IDC for a batch Poisson process we need var($p(t)$). Using the properties of conditional probability, we have

$$\text{var}(p(t)) = E(p^2(t)) - E^2(p(t)) = E\Big[E(p^2(t)\,|\,N(t))\Big] - E^2\Big[E(p(t)\,|\,N(t))\Big] .$$

We expand the first of the last two terms using the definition of variance

$$\text{var}(p(t)) = E\Big[\text{var}(p(t)\,|\,N(t))\Big] + E\Big[E^2(p(t)\,|\,N(t))\Big] - E^2\Big[E(p(t)\,|\,N(t))\Big] ,$$

and then condense the two rightmost terms immediately above using the definition of variance again:

$$\text{var}(p(t)) = E\Big[\text{var}(p(t)\,|\,N(t))\Big] + \text{var}\Big[E(p(t)\,|\,N(t))\Big] .$$

Because $N(t)$ is independent of p_i, we can finally derive

$$\text{var}(p(t)) = E(N(t)\text{var}(p_i)) + \text{var}(N(t)E(p_i)) = \lambda t(\text{var}(p_i) + E^2(p_i)) .$$

From definition (4.5), the IDC is then

$$I_t = \frac{\text{var}(p_i)}{E(p_i)} + E(p_i) .$$

Notice that the IDC is constant since a batch Poisson process is a case of a regenerative process with independent increments.

When the distribution of batch arrivals is geometric, i.e., $\Pr(N = n) = (1-p)p^{n-1}$, with $0 < p < 1$, the IDC of the batch Poisson process becomes $I_t = \frac{1+p}{1-p}$.

It should be pointed out that, when a renewal approximation is used for bursty non-renewal processes, a fitting based on the value of the moments may not be the best. As we will see later when we estimate the indices of dispersion for packet arrival processes, the IDI asymptote may be two orders of magnitude larger than the squared coefficient of variation, that is, the value of J_1, or between the values for large t and small t of the IDC. If the main objective of a renewal approximation is to capture the variability of a point process, the fitting should be done in such a way that the resulting constant index of dispersion of the model intersects the estimated index of dispersion of the point process at an intermediate point. For instance, if a batch-Poisson model is used with geometrically distributed batch sizes, the parameter λ could be set on the basis of the estimated mean and the parameter p set to a value that gives $(1+p)/(1-p)$ an appropriate position on the estimated IDC of the point process.

E. IDC for Markov-Modulated Poisson Processes

The Markov-modulated Poisson process (MMPP) is a model that has received much attention in recent years. It is a powerful, analytically treatable model that can represent aggregate traffic generated by the superposition of several point processes. The MMPP process is a doubly stochastic Poisson process whose arrival rate varies according to the state of an n-state irreducible continuous-time Markov chain, and that, unlike renewal models, can represent correlations between interarrival times. When the Markov chain is in state i, the arrival process is Poisson with rate λ_i. In the following we will only consider a 2-state MMPP.

The 2-state MMPP process is fully specified by four parameters: two transition rates of the Markov chain and two arrival rates (one for each state). It is described by the infinitesimal generator matrix $\mathbf{Q}$ of the embedded Markov chain and by a diagonal matrix $\mathbf{\Lambda}$ whose elements are the Poisson rates of the two states:

$$\mathbf{Q} = \begin{bmatrix} -\sigma_1 & \sigma_1 \\ \sigma_2 & -\sigma_2 \end{bmatrix} \quad \text{and} \quad \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

The first moment of the time between arrivals of an MMPP can be expressed as

$$\mu_1 = \mathbf{p}\left[\mathbf{\Lambda} - \mathbf{Q}\right]^{-2}\mathbf{\Lambda}\mathbf{e},$$

where $\mathbf{p}$ is the vector $\left[\frac{\lambda_1\sigma_2}{(\lambda_1\sigma_2 + \lambda_2\sigma_1)}, \frac{\lambda_2\sigma_1}{(\lambda_1\sigma_2 + \lambda_2\sigma_1)}\right]$ and $\mathbf{e}$ the vector (1, 1). The second product moment is

$$\mu_2 = 2\mathbf{p}\left[\mathbf{\Lambda} - \mathbf{Q}\right]^{-3}\mathbf{\Lambda}\mathbf{e}.$$

For derivations of the above results see [56] or [28]. After expanding and simplifying, we obtain for the two moments

$$\mu_1 = \frac{\sigma_1 + \sigma_2}{\lambda_1\sigma_2 + \lambda_2\sigma_1} \quad \text{and} \quad \mu_2 = \frac{2(\lambda_1\sigma_1 + \lambda_2\sigma_2 + \sigma_1^2 + 2\sigma_1\sigma_2 + \sigma_2^2)}{(\lambda_1\sigma_2 + \lambda_2\sigma_1)(\lambda_1\sigma_2 + \lambda_2\sigma_1 + \lambda_1\lambda_2)},$$

from which we can easily derive the squared coefficient of variation for intervals: $C_1^2 = \mu_2/\mu_1^2 - 1$.

Heffes and Lucantoni [35] derive a formula for the IDC of a 2-state MMPP process:

$$I_t = 1 + \frac{2\sigma_1\sigma_2(\lambda_1 - \lambda_2)^2}{(\sigma_1 + \sigma_2)^2(\lambda_1\sigma_2 + \lambda_2\sigma_1)} - \frac{2\sigma_1\sigma_2(\lambda_1 - \lambda_2)^2}{(\sigma_1 + \sigma_2)^3(\lambda_1\sigma_2 + \lambda_2\sigma_1)t}\left(1 - e^{-(\sigma_1 + \sigma_2)t}\right). \tag{4.7}$$

The asymptote of the IDC is

$$I_\infty = 1 + \frac{2\sigma_1\sigma_2(\lambda_1 - \lambda_2)^2}{(\sigma_1 + \sigma_2)^2(\lambda_1\sigma_2 + \lambda_2\sigma_1)},$$

and it is also straightforward to verify that

$$\frac{I_\infty - I_{t_0}}{I_\infty - 1} = \frac{1 - e^{-rt_0}}{rt_0}, \tag{4.8}$$

where $r = \sigma_1 + \sigma_2$ can be interpreted as the "rate" at which the IDC approaches its asymptote. Equation (4.8) can be used to estimate r for a measured arrival process since the lefthand side can be easily evaluated from a point at t_0 on the IDC and the estimated IDC asymptote; r can then be obtained by solving (4.8) numerically.

Of the three processes we have analyzed, only the MMPP can be used to represent correlations between subsequent arrivals. A model based on hyperexponential interarrival times is less appropriate than a batch Poisson model to approximate highly variable non-Poisson measured processes since its interarrival-time distribution is close to an exponential distribution when its coefficient of variation is large. In the next sections, we examine measured arrival processes, estimate their indices of dispersion and, finally, outline a procedure to fit the four MMPP parameters to bursty arrival processes.

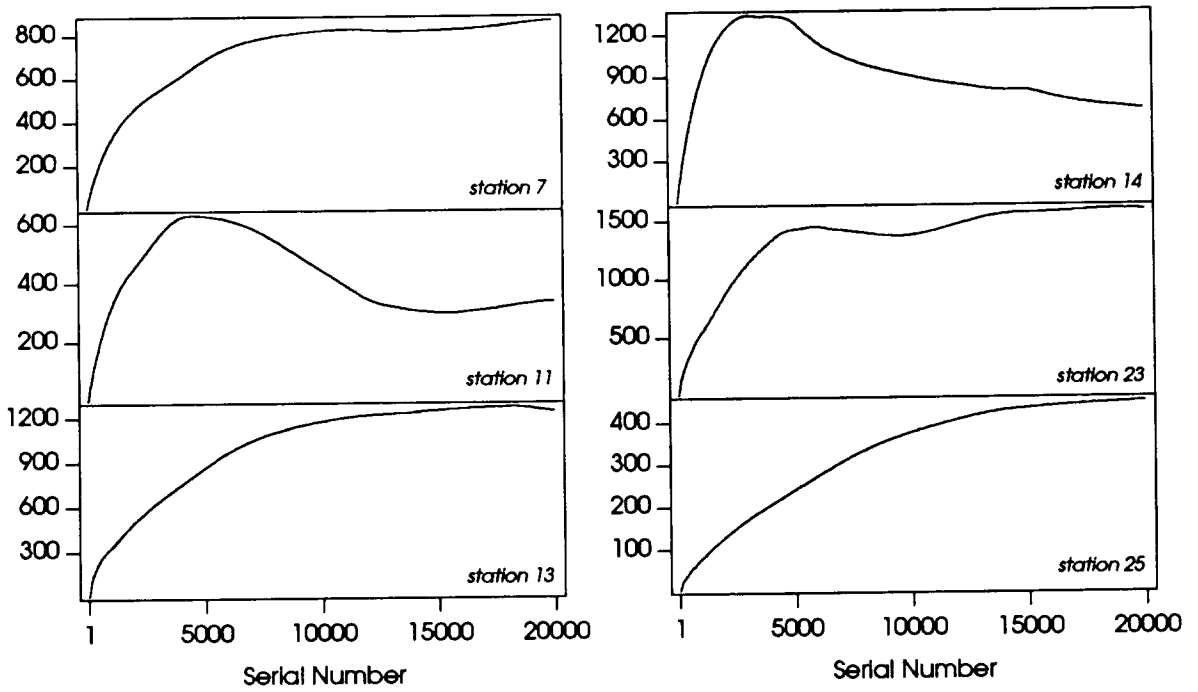
III. Estimated Indices of Dispersion

A. Estimated Indices of Dispersion for Intervals

In this section, using index-of-dispersion analyses, we look at the second-order joint probability structure of the packet arrival processes generated by six individual workstations. In Figure 4.4 we plot the estimated index of dispersion curves for the send queues of these workstations. Each series contains 100000 interarrival times. The IDI curves are estimated for up to 20% of the length of the original series of data; after this point, with few remaining degrees of freedom, any further estimate would have been inaccurate. For details on how to estimate the indices of dispersion and how to evaluate the precision of the estimates, refer to [19].

The IDI at lag n is the variance of the sum of n successive interarrival times, and the IDI curve indicates the change in the variance as n increases. What appears to be very large variability (the starting values of the IDI for the six curves are, proceeding from *station 7* to *station 25*, 7.2, 3.5, 6.4, 6.8, 13.0, 5.4, while the maxima range from 400 to more than 1500) is caused primarily by nonstationary components in the data. Most remarkable is the effect on the curve of *station 14*, which increases sharply until approximately lag 3000, then stabilizes for the next 2000 points, and begins decreasing from lag 5000 onwards. From equation (4.3) we would think that the autocorrelation coefficients of this series of data become negative at around lag 5000, but this is not the case: the coefficients are all positive and only slowly decrease, another sign of nonstationary data.

FIGURE 4.4. ESTIMATED INDEX OF DISPERSION FOR INTERVALS – WORKSTATION SEND QUEUES

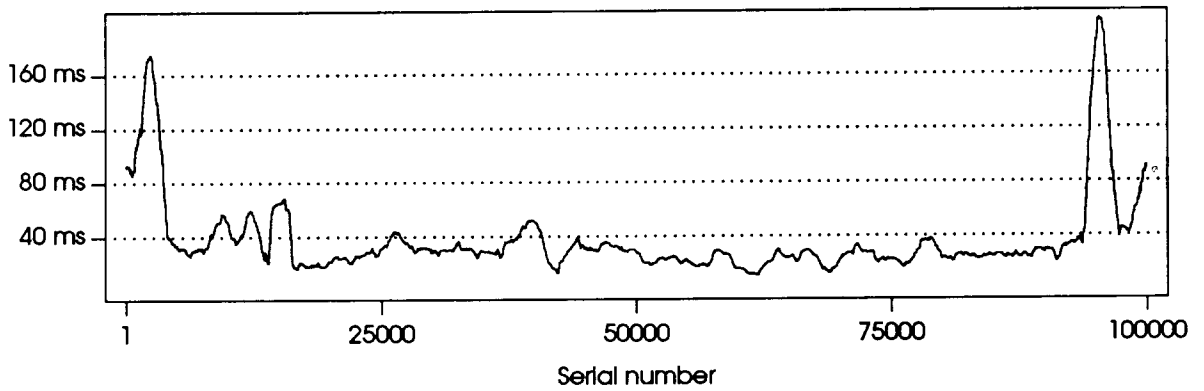


A look at the smoothed interarrival-time curve of *station 14*'s arrival process, illustrates what is at work. (The smoothing was done by lowpass filtering the data in the

frequency domain.) In Figure 4.5, the two peaks of long interarrival times, one at the beginning and the other at the end of the graph, cause the total variance and the IDI to decrease as more points are averaged together. Nonstationary data with a minority of large values clustered together generate a gradually decreasing IDI curve such as *station 14*'s.

It could be argued that the apparent nonstationarity of the data in Figure 4.5, which spans a period of about 1 hour, is a function of the time scale. This particular user might follow the same work patterns and produce roughly the same workload in successive hours; thus, viewed over a period of several hours, the data might appear to be (more) stationary.

FIGURE 4.5. SMOOTHED INTERARRIVAL TIME CURVE – WORKSTATION *STATION 14*



For our purpose of analyzing variability however, the relevant time scale is not one of several hours, or even minutes, but a shorter one, for a workstation will generate a constant packet rate in intervals ranging from milliseconds to seconds. Analyzing the variability of point processes at a micro level, i.e., in terms of the queues to which the processes are fed, implies a time scale that is this short. In contrast, a study of packet arrival processes from the perspective of user behavior would involve a much longer time scale, one defined by the busy/idle intervals of user behavior, which would range from several minutes to hours.

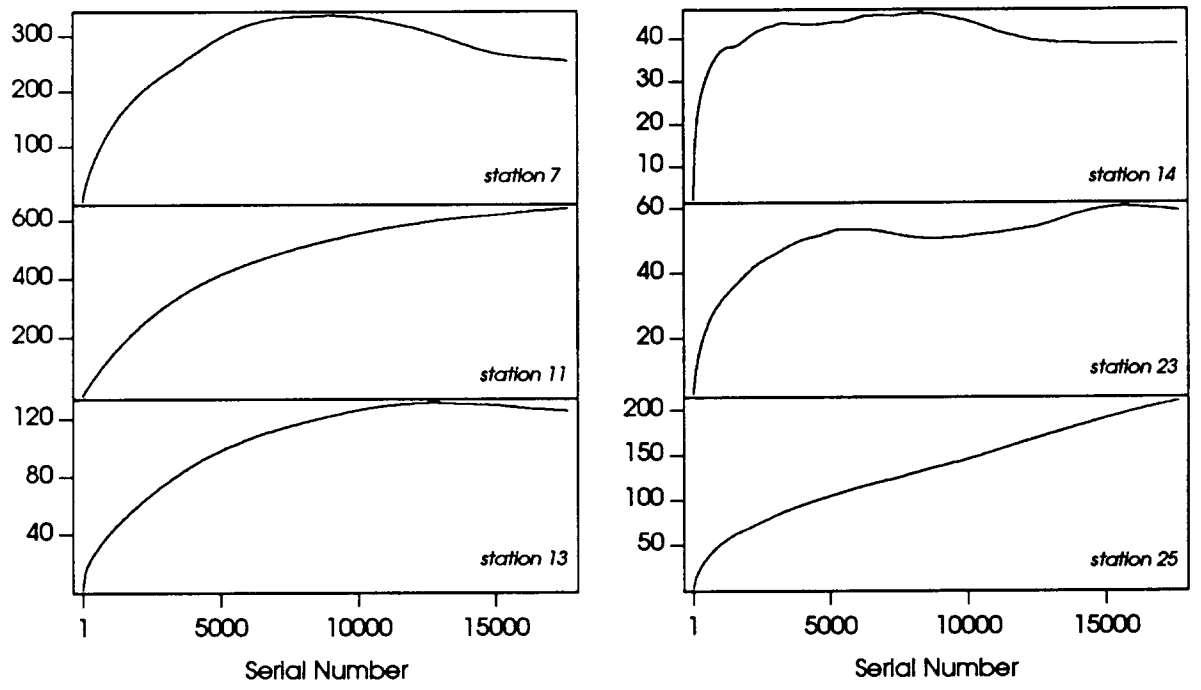
Nonstationary behavior in these types of arrival processes is the norm. It is virtually impossible to isolate a stationary segment of a process long enough for the estimation of many important statistical parameters. It is possible, however, to identify segments of time during which the process has roughly the same characteristics. One can then juxtapose the various segments with the same properties, assemble several series of arrivals, each of which is what we have called a *phase*, and derive the statistical description for each phase. As a very simple illustration of this procedure, let us re-examine the processes shown in Figure 4.4 by considering subsets of interarrival times shorter than a specified length.

The cut-off time chosen should be short enough to eliminate the two peaks shown in Figure 4.5 but sufficiently long to capture not only the fast protocol transactions such as file path-name translations, but also the slower disk transactions. In Figures 3.10 and 3.11,

the height of the histograms of interarrival times decreases visibly at or around 80 ms. This decrease indicates that at around 80 ms there is a transition from frequent system-generated events, driven by network protocols and disk-driver software, to infrequent user-generated ones, such as keystrokes. All of the workstations in our traces show similar histograms, an indication that the results described in the reminder of the chapter are relatively insensitive to variations of this parameter. Thus, we set the threshold at 80 ms, a value that we have indicated in Figures 3.10 and 3.11 with a dashed line.

The validity of this approach in reducing the nonstationarity measured by the index of dispersion for intervals can be sustained with the following argument. The probability density functions of the interarrival times for packet arrival processes typically have a very large mass at the beginning and very long and low tails, which results in the median being smaller than the mean value. (In Figures 3.10 and 3.11 we show the mean and median of the histograms.) The index of dispersion for intervals, as we have seen in equation (4.1), depends on the autocovariances of the interarrival times. The autocovariance at lag k is defined as $E\left[(X_i - E(X))(X_{i+k} - E(X))\right]$ where $E(X)$, as usual, indicates the mean interarrival time. We see that, for each lag k, the few long interarrival times, because of their large differences from the mean, account for much of the total covariance. Conversely, the large number of small interarrivals closer to the mean have relatively little effect on the covariance. Hence, removing the largest interarrival times from a time series that is skewed towards small values reduces the index of dispersion for intervals and, if the large values are clustered as in Figure 4.5, may prevent the IDI's of packet arrival processes, which normally have positive correlation coefficients, from not being monotonic.

FIGURE 4.6. ESTIMATED INDEX OF DISPERSION FOR INTERVALS – INTERARRIVALS SMALLER THAN 80 MS



In Figure 4.6 we re-evaluate the indices of dispersion for the subset of interarrivals that are smaller than 80 ms. We find that, in three cases (*station 13*, *station 14*, and *station 23*), the asymptotic value is reduced by a more than, or nearly, one order of magnitude. The series of data are 4% to 11% shorter than their counterparts in Figure 4.4 because the interarrivals greater than 80 ms have been dropped.

Analysis of the smoothed interarrival times of these truncated series, which are shown in Figure 4.7, helps us understand the dynamics of the IDI behavior. Each of these curves was obtained by averaging together 200 successive interarrival times. We notice that the two peaks in *station 7*'s interarrival times are responsible for the decreasing slope of that machine's IDI. The almost linearly increasing slope of *station 25*'s IDI can be understood by looking at two segments of the machine's interarrivals: one between 20000 and 40000 and a second between 60000 and 80000. The first interval contains interarrival times smaller than the mean; the second, interarrival times larger than the mean. Since the size of each of these segments happens to be equal to the length of the interval over which we estimate the IDI, upon a moment's reflection one will realize that the variance of the sum of consecutive interarrivals will increase linearly.

The range of *station 11*'s IDI in Figure 4.6 is the same as that in Figure 4.4, but the second curve increases more gradually and is monotonic. The smoothed interarrival times of *station 11* in Figure 4.7 explain why the range of the truncated series is the same as the original one: despite the elimination of longer interarrival times, the series remains highly variable. The user of this particular machine was not active for much of the time represented in Figure 4.6; during the inactive stretches, programs left running in the machine invoked remote procedure calls but did not transfer data. The remote procedure calls produced longer interarrival times, the lack of data transfers did not produce the shortest interarrival times. The nature of *station 11*'s arrival processes reminds us that the study of indices of dispersion is laden with complexities.

B. Estimated Indices of Dispersion for Counts

Figure 4.8 shows the estimated index of dispersion for counts. The IDC was evaluated only up to one-sixth of the total time length of the traces, or about 11 min. In each of the graphs, packet counts were estimated in slots of size 50 ms.

A description of a point process in terms of counts is statistically equivalent to a description in terms of intervals [20]. However, they are equivalent only through their complete joint distributions. If we restrict ourselves to first- and second-order properties, the two characterizations are separately informative. For instance, histograms of packet counts, of which we show two examples in Figures 4.9 and 4.10, are rather different in shape from the histograms of intervals shown in Figures 3.7 and 3.9. Analogously, the estimated IDC curves in Figure 4.8 are substantially different from their IDI counterparts in Figure 4.4. However, their technical interpretation, since equations (4.6) and (4.3) are quite similar, proceeds along the same lines.

FIGURE 4.7. SMOOTHED CURVES OF TRUNCATED INTERARRIVAL TIME SERIES

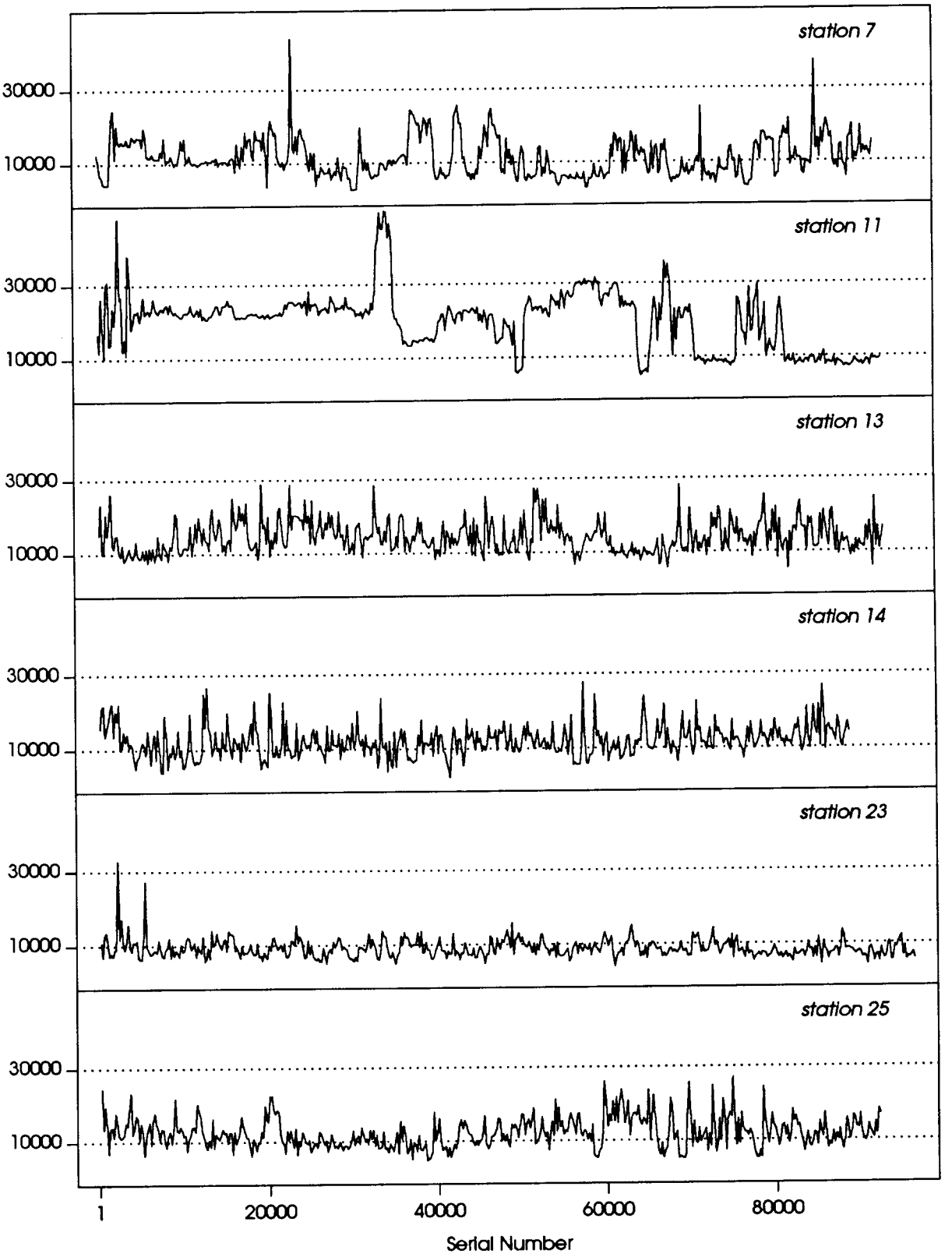


FIGURE 4.8. ESTIMATED INDEX OF DISPERSION FOR COUNTS – WORKSTATION SEND QUEUES

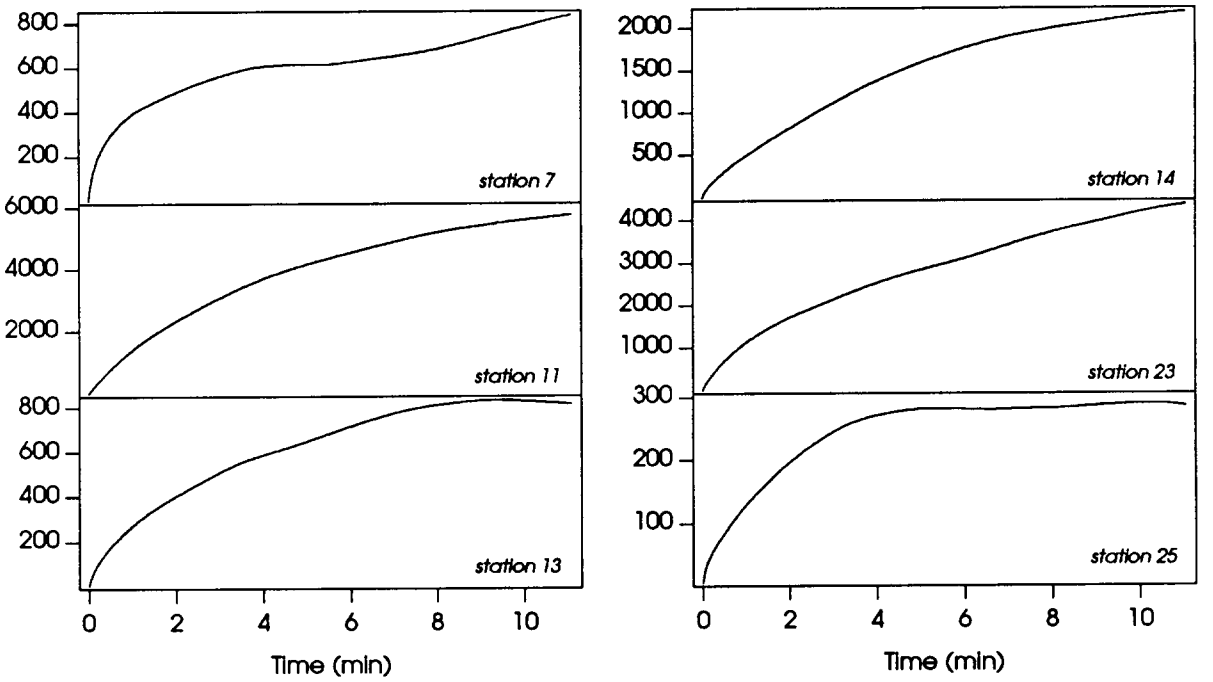


FIGURE 4.9. HISTOGRAM OF PACKET COUNTS WORKSTATION STATION 3

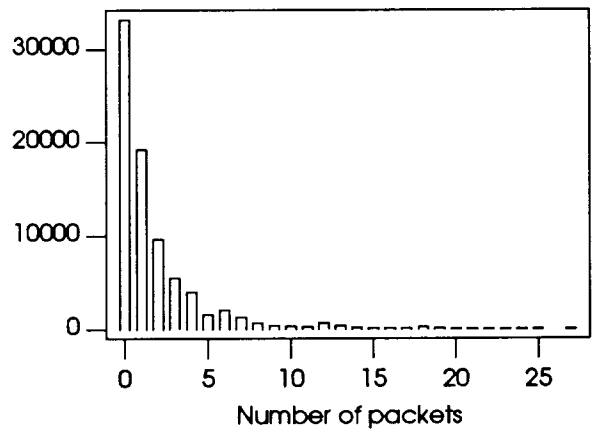
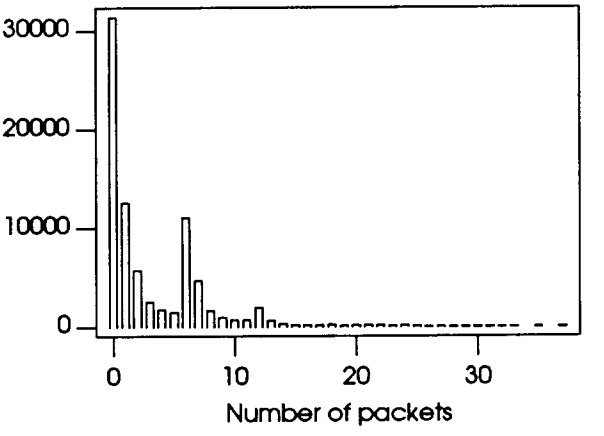


FIGURE 4.10. HISTOGRAM OF PACKET COUNTS WORKSTATION *STATION 23*



Here also we see the effect of nonstationary components. The IDC of *station 11* is particularly affected by the grouping of arrivals in some regions of the domain of the packet count process. *Station 11* has an almost linearly increasing IDC, which can be attributed to the same sort of nonstationary data structure underlying the linearly growing IDI of *station 25*. In general, since the domain of values of counts is rather limited (especially so when packet counts are estimated over relatively short intervals, as in our case), we can say that IDC curves are more sensitive than IDI curves to the presence of

clusters of arrivals. The limited range of the domain of packet counts results in probability density functions of counts with very short tails.

It is important to notice that the asymptotes in Figure 4.8 are different from those in Figure 4.4. This is yet another confirmation of the presence of nonstationary data. We have recalculated (but not shown here) the IDC's for interarrivals less than 80 ms. In this case, there is a much better agreement between the data and the limiting result that states that I_∞ and J_∞ are equal (Section II.B). Indeed, if the shapes of the IDI and IDC curves for an arrival process appear monotonic, and the limits of the two indices are the same, one can assume with a considerable degree of confidence that the data are stationary.

In this section we have shown and interpreted the IDI and IDC of several measured packet arrival processes. The probabilistic definitions of these indices makes them suitable for describing the variability of point processes. We next show how this variability representation can be incorporated into analytical modeling.

IV. Fitting a 2-State MMPP to an Arrival Process

In this section we present a procedure that can be used to fit an MMPP model to packet arrival processes of the type we have been describing as long as the nonstationary data components are somewhat controlled. To provide a concrete example, we will work with the data of workstation *station 13*, one of the most stationary data sets. *Station 13*'s indices of dispersion appear in Figures 4.4 and 4.8. For the mathematical terminology, refer to the MMPP definitions in Section II.E.

We can write the following system of three equations, representing, from top to bottom, the mean interarrival time of an MMPP, the asymptote I_∞, and what we have defined in equation (4.8) as r, the rate at which the IDC approaches the asymptote:

$$\begin{cases} \dfrac{\sigma_1 + \sigma_2}{\lambda_1\sigma_2 + \lambda_2\sigma_1} = a \\ 1 + \dfrac{2\sigma_1\sigma_2(\lambda_1 - \lambda_2)^2}{(\sigma_1 + \sigma_2)^2\,(\lambda_1\sigma_2 + \lambda_2\sigma_1)} = b + 1 \\ \sigma_1 + \sigma_2 = c \ . \end{cases} \tag{4.9}$$

The quantities a and b represent, respectively, the estimated mean of the interarrival times of a measured point process and the estimated value of the IDC asymptote minus 1, both of which can be obtained with modest effort. An initial value for the parameter c, an estimate of r, can be computed numerically, as indicated in Section II.E, from b and I_{t_0}, the IDC at time t_0. The choice of t_0 is not crucial, as we can repeat part of the procedure we are about to describe until we reach a satisfactory approximation based on some measure of the goodness of fit.

We begin by solving the three equations (4.9) to obtain the values of λ_1, σ_1, and σ_2 as functions of a, b, c, and the unknown λ_2:

$$\begin{cases} \lambda_1 = \dfrac{2 + abc - 2a\lambda_2}{2a - 2a^2\lambda_2} \\ \sigma_1 = \dfrac{abc^2}{2 + abc - 4a\lambda_2 + 2a^2\lambda_2^2} \\ \sigma_2 = \dfrac{2c(a\lambda_2 - 1)^2}{2 + abc - 4a\lambda_2 + 2a^2\lambda_2^2} \end{cases} . \tag{4.10}$$

Next, we equate the formula of the squared coefficient of variation for an MMPP, $C_f^2 = \mu_2/\mu_1^2 - 1$, to the square of the estimated value of the coefficient, which we call d, in order to determine the value of the unknown λ_2. Since μ_1 and μ_2 depend only on the four MMPP parameters, we can substitute the values above to obtain a formula for d in λ_2:

$$d = \frac{2a\lambda_2^2 + (2ac + abc - 2)\lambda_2 - 2c(b + 1)}{2a\lambda_2^2 + (2ac - abc - 2)\lambda_2 - 2c} . \tag{4.11}$$

The expression of λ_2 in terms of the quantities a, b, c, and d is simple but rather tedious and we will omit the details here. Note, however, that for the righthand side of equation (4.11) the limit as λ_2 approaches infinity is 1 and the limit as λ_2 goes to 0 is $b + 1$.

To fit a 2-state MMPP to a measured arrival process we set the four parameters as follows:

1. From the data, estimate a, the mean interarrival time; b, the limiting value of the IDC minus 1; and d, the squared coefficient of variation of the interarrival times.
2. Using b, t_0, and I_{t_0}, the value of the IDC at time t_0, estimate numerically an initial value for the rate c by solving equation (4.8).
3. From the solutions to equation (4.11), obtain a value for λ_2, and use it to derive values for λ_1, σ_1, and σ_2 from equations (4.10). (Note that, in general, there are two solutions for λ_2 from equation (4.11).)
4. Compute, based on the current values of the parameters, the goodness of the approximation by comparing the estimated IDC with the theoretical one calculated by equation (4.7). A typical test for the goodness of the fit is one that evaluates the sum of the squared distances between the estimated and the theoretical IDC curves (for some applications it may be worth evaluating the goodness of fit only over a portion of the domain of the IDC). Finally, adjust the value of c as appropriate to improve the fit and repeat steps 3 and 4 of this procedure until the approximation is satisfactory. (A smaller c will make the IDC reach the asymptote more slowly.)

We now apply the procedure outlined above to the arrival process generated by *station 13*'s send queue. We use the index of dispersion for counts computed only for interarrival times smaller than 80 ms. The resulting IDC is related to *station 13*'s IDI shown in Figure 4.6. The estimated mean interarrival time is 0.01376 s, the value of b is 112, and the squared coefficient of variation 1.794. Using the value 20 s for t_0 and setting the estimated I_{20} at 54, we obtain for c a value of 0.073 $(sec)^{-1}$. The resulting parameters (all of them rates with dimensions $(sec)^{-1}$) are

$$\lambda_1 = 9.61\,,\quad \lambda_2 = 77.37\,,\quad \sigma_1 = 0.0680\,,\quad \sigma_2 = 0.0051\,. \tag{4.12}$$

There is another symmetric set of solutions, which corresponds to the above with the subscripts 1 and 2 exchanged.

In Figure 4.11 we plot the righthand side of equation (4.11), the squared coefficient of variation for intervals of an MMPP with parameters given in (4.12) above. (In this and in the following two figures, black dots indicate the position of the two sets of solutions on the curves.)

FIGURE 4.11. SQUARED COEFFICIENT OF VARIATION OF X_i

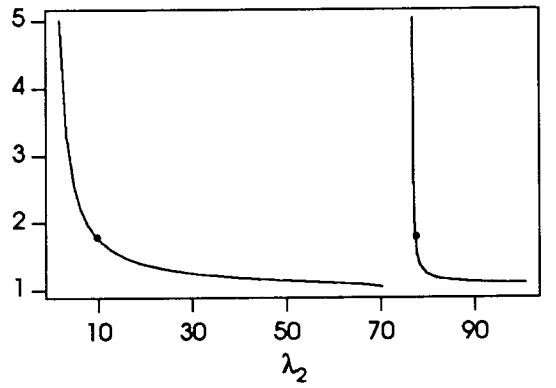


FIGURE 4.12. POISSON RATE λ_1

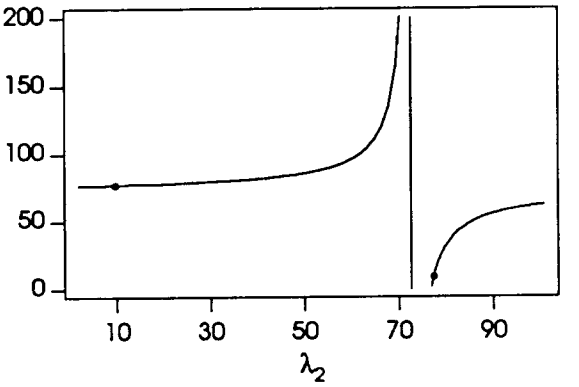


FIGURE 4.13. MARKOV CHAIN TRANSITION RATES

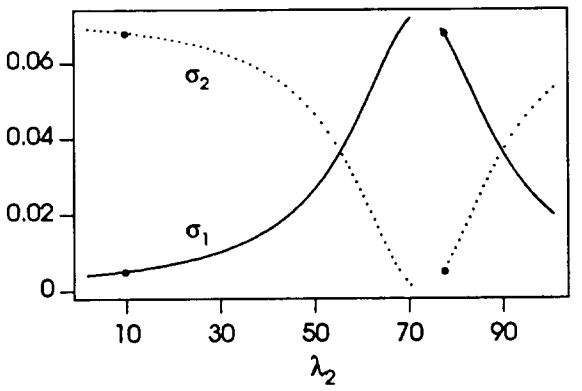
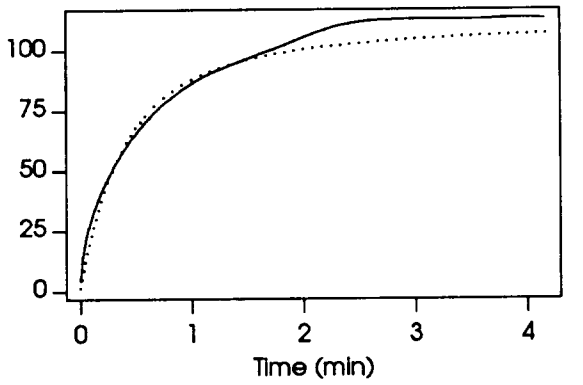


FIGURE 4.14. IDCS OF DATA AND FITTED MODEL (DOTTED)



Since d must be larger than 1, observe that we have not drawn the segment of the squared coefficient of variation curve for points where the function in Figure 4.11 is smaller than 1. Figure 4.12 shows a plot of the first of equations (4.10); there is a vertical asymptote at $\lambda_2 = 1/a$. Here, we have eliminated from the domain the region in which the equation generates a negative value for the Poisson rate. Finally, in Figure 4.13 (only on points belonging to the domain of the previous two functions) we show the curves for the transition rates of the Markov chain.

In Figure 4.14 we plot the estimated IDC for the data (depicted as a continuous curve) as well as the model IDC (depicted as a dotted curve). The fitting is very good in the region of the domain from 0 to 1.5 min, the portion of the IDC that is most likely to affect interactive queues.

V. Summary

In this chapter we have used the index of dispersion for intervals and the index of dispersion for counts to characterize arrival processes consisting of packets sent by workstations in an Ethernet local-area network. By evaluating the indices for three analytical models we have illustrated some of their properties. We have suggested that renewal approximations of non-renewal point processes may benefit from index of dispersion analyses. Because indices of dispersion reveal a good deal about the correlation structure of point processes occurring in communication networks, index-of-dispersion analysis should be adopted as one of the basic tools for examining these point processes.

We have introduced a procedure for utilizing indices of dispersion to fit an MMPP model to measured data. We propose testing the quality of the approximation by comparing the IDI or IDC of the data with the model's corresponding indices. Our purpose has not been to propose models for packet arrival processes, or to demonstrate that any one model best approximates workstation traffic, but to demonstrate how to incorporate short- and long-term variability characterizations based on indices of dispersion into arrival models.

5 Models of Variability

Modeling always involves some degree of approximation. We have seen how complex and varied the features of packet arrival processes are. Not only do users produce different types of workloads, but also, at a lower level, communication requests use a multitude of packet sizes and several different protocol types. In building models, one has to strike a balance between accuracy of feature representation on one side and simplicity and mathematical tractability on the other. However, often researchers opt for simplicity of mathematical analysis and ignore key features of their point processes. Although this approach can lead to useful approximations, unrealistic models are often obtained, which cannot be viewed and used with full confidence.

In this chapter we attempt to develop models of packet arrivals that are based on our knowledge of packet length and interarrival-time patterns present in our data and of certain statistical properties of these patterns. Packet length is not a factor in the Markov-Modulated Poisson-Process (MMPP) characterization of packet-arrival processes that was outlined in Chapter 4. The MMPP model is built on data for packet counts—groups of packets arriving in the same interval—and does not incorporate any data on packet length. However, packet length information is important if traffic models developed on the basis of our data are to be used in connection with studies of non-Ethernet networks, for variable packet lengths can have a major impact on traffic patterns in those networks. For instance, in an Asynchronous Transfer Mode (ATM) network, the transmission unit, the *cell*, has a constant length of 53 bytes, of which 48 are for the user's *payload*. Whereas a 46-byte Ethernet packet would correspond to a single ATM cell, a 1500-byte packet would require 32 ATM cells. Thus, for models of Ethernet workstation traffic to have more general applicability, they should include packet length information, and should model interarrival times rather than packet counts. Hence, we will seek two-dimensional point process models in which the coordinates of each point are an interarrival time and a packet length.

Our analysis of packet lengths in Chapter 2 indicated that packet lengths in our traces tend to be either shorter than 200 bytes or longer than 500 bytes. Furthermore, short packets were typically associated with long interarrival times (recall that we associate with an interarrival time the first packet of each packet pair generating the interarrival time), while long packets typically produced short interrival times. These patterns partition the packet-length and the interarrival-time spaces and suggest that an appropriate model might be one with two states: one for the short packets, and one for the long packets. (Notice that the two states in the MMPP model of Chapter 4 are really

only a mathematical construction and are not grounded on empirical reality.)

If the model is to have two states, we must determine how long each state should last. The answer to this question is provided by data on the lengths of sequences of short packets—strings of back-to-back short packets—and on the lengths of sequences of long packets. Section II examines our data on sequence length. We observe that the short- and long-packet sequence series are independent, and moreover, in each series, the probability density decreases approximately exponentially with the sequence length. These characteristics suggest the geometric distribution would model reasonably well the sequence lengths in each of the two states, and that a two-state Markov chain would be an appropriate model for the alternating process of short and long sequences.

Our next step is to propose a model that incorporates the Markov chain and produces interarrival times and packet lengths. We suggest that packet lengths and interarrival times can be generated in each state using probability density functions estimated from the data. Were these probability functions sampled independently, we would have a semi-Markov process. Since this is not the case with our processes, in Section III we extend the two-state semi-Markov process to the case in which the probability functions in the two states are not independent. We solve the model in terms of the autocovariances of the two component functions and fit it with the measured data. Unfortunately, the model's covariance does not approximate well the covariance of the data for intermediate and large values of the lag.

In order to understand the reasons for this divergence, we next generate several artificial point processes each of which contradicts one of the generalized model's assumptions. We discover that the problems are caused by the nonstationary components that are present in the measured arrival processes. Because of this negative result, we are forced to conclude that the extended semi-Markov model can only be used to represent accurately packet arrival processes over short time scales. However, we observe that the queuing dynamics of interest in interactive distributed systems, such as networks of personal workstations, occurs over short time scales.

II. Models for Packet Sequences

In Figure 3.15 we show the distribution of packet-lengths for aggregate traffic in the Sun Engineering network. We see that, with almost no exception, packets are either shorter than 200 bytes or longer than 500 bytes. Short packets are produced either by TCP character traffic, as discussed in Appendix A, or by Remote Procedure Call (RPC) packets, as discussed in Chapters 2 and 3. In contrast to the Berkeley network described in Appendix A, in which character traffic represented a sizable proportion of the total traffic (see Figure A.3), in the Sun Engineering network character traffic accounts for only about 6 percent of the total (see Figure 3.15). Instead, RPC packets account for more than 34 percent of the total packet traffic in the Sun network. The majority of long packets are 1500-byte fragments of larger messages generated by higher protocol layers. Fragmentation, involving a straightforward sequence of operations that can be streamlined, results in short interarrival times between fragments. As we have shown in Figures 4.10 and 4.11, when two successive messages are fragmented, the interarrival time between the last fragment of the first message and the first fragment of the second is normally large. (We have defined these pairs of fragments as "LastFragment/1500".)

Thus, since the distribution of packet lengths can be divided into two regions each of which has individual interarrival time properties and is produced by different system components, it is reasonable to classify packet lengths into two classes, *small* and *large*, depending on which of the two groups they belong to.

We will now compute the distribution of the lengths of short- and long-packet sequences—strings of consecutive short and strings of consecutive long packets. Following the methodology of Chapter 3, we will limit the data set only to pairs of packets with interarrival times of less than 80 milliseconds (as usual, each interarrival time will be associated with the first of the two packets that produced it). In Figure 5.1 we show the histogram of short-packet sequence lengths for the send queue of *station 25*; in Figure 5.2, the histogram of long-packet sequence lengths for the same station. Figures 5.3 and 5.4 display the same type of histograms for *station 25*'s receive queue. Both pairs of graphs are semi-logarithmic.

FIGURE 5.1. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 25* – SEND QUEUE

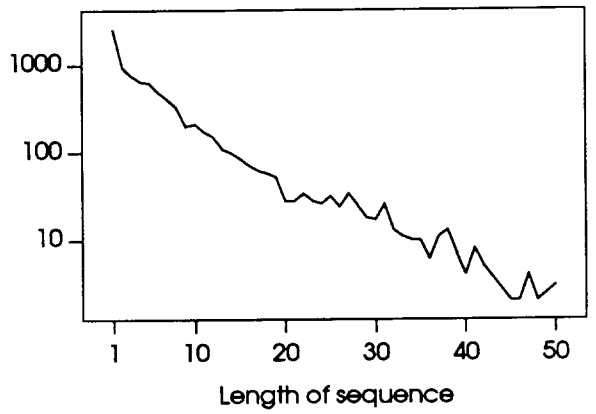


FIGURE 5.2. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 25* – SEND QUEUE

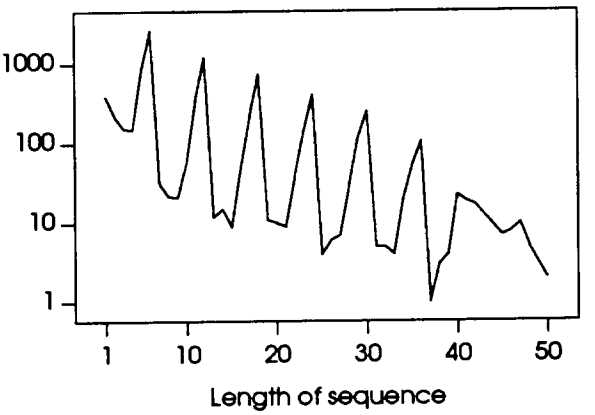


FIGURE 5.3. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 25* – RECEIVE QUEUE

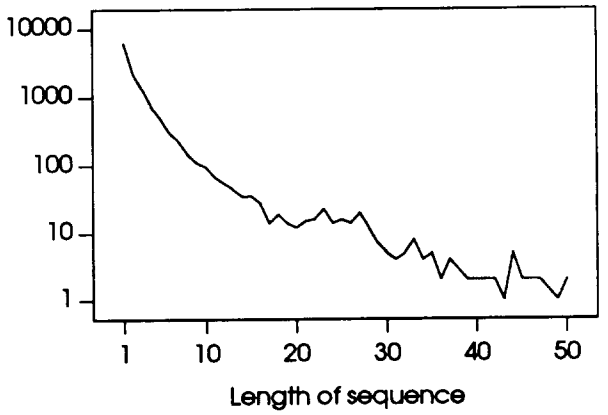
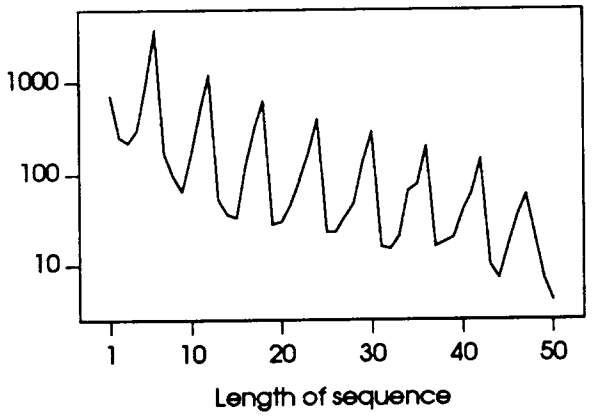


FIGURE 5.4. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 25* – RECEIVE QUEUE



In Figures 5.2 and 5.4 we can clearly see the effects of the 8-Kbyte message fragmentation, which produces fragments in groups of six. Often, two or more groups follow each other producing peaks at values that are multiples of six. However, since we are

only considering interarrival times shorter than 80 milliseconds, in some cases we discard longer interarrival times produced by "1500/LastFragment" pairs (see Figures 4.10 and 4.11). When this happens, the lengths of some sequences of long packets are no longer a multiple of six. The existence of a few non-NFS sources of large packets and of concurrency among streams produced by independent processes running on the client workstation explains why Figures 5.2 and 5.4 do not consist simply of single lines at values that are a multiple of six.

The graphs in Figures 5.1 and 5.3 are approximated well by straight lines. Since these histograms are plotted as semi-log graphs, in which straight lines correspond to exponentially decreasing curves, the associated functions are negative exponentials. Although the graphs in Figures 5.2 and 5.4 show a more complex structure, the peaks nearly fall into a straight line and so do moving-average plots of the two graphs. Hence, we decide to ignore the level of detail represented by the peaks and to consider also the sequences of long packets as produced by exponentially decreasing probability density functions.

In Figures 5.5 and 5.6 we show the autocorrelation coefficients from lag 1 to lag 100 for the series of short-packet and long-packet sequence lengths, respectively, of *station 25*'s send queue. The dashed lines indicate the 99 percent confidence bands for uncorrelated series with the same number of elements. We see that the autocorrelation level is very small, and, for practical purposes, both series can be considered self uncorrelated.

FIGURE 5.5. CORRELOGRAM FOR THE SERIES OF LENGTHS OF SHORT PACKET SEQUENCES – *STATION 25*'S SEND QUEUE

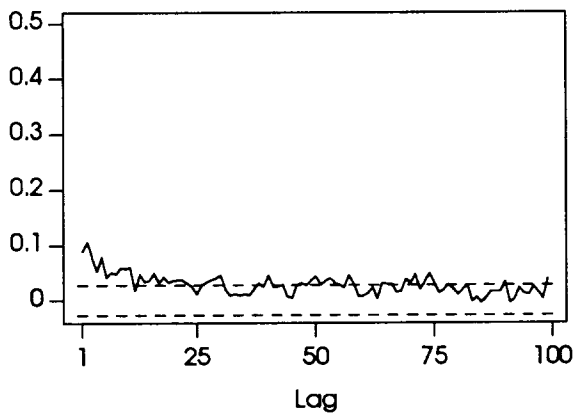
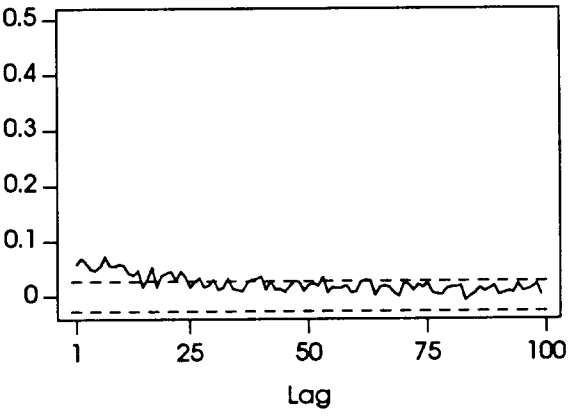


FIGURE 5.6. CORRELOGRAM FOR THE SERIES OF LENGTHS OF LONG PACKET SEQUENCES – *STATION 25*'S SEND QUEUE



To confirm that there is no autocorrelation, we show in Figures 5.7 and 5.8 scatter plots at lag 1 of the lengths of short- and long-packet sequences, respectively. Each point in these graphs has for coordinates two successive sequence lengths, L_i on the x-axis and L_{i+1} on the y-axis. Although in these figures we have truncated the range of possible lengths at a maximum of 50 packets, in doing so we have dropped less than 0.5 percent of all sequences. Since sequences have discrete values, for clearer visualization of the point distribution we have added a random displacement to the coordinates of each point so that a point will lie anywhere in a square of size 1 centered on the original point position. The two plots are symmetric with respect to the line $y = x$, an indication

that the probability density functions of the sequence lengths have been sampled independently.

FIGURE 5.7. SCATTER PLOT OF LENGTHS OF SEQUENCES OF SHORT PACKETS – *STATION 25*'S SEND QUEUE

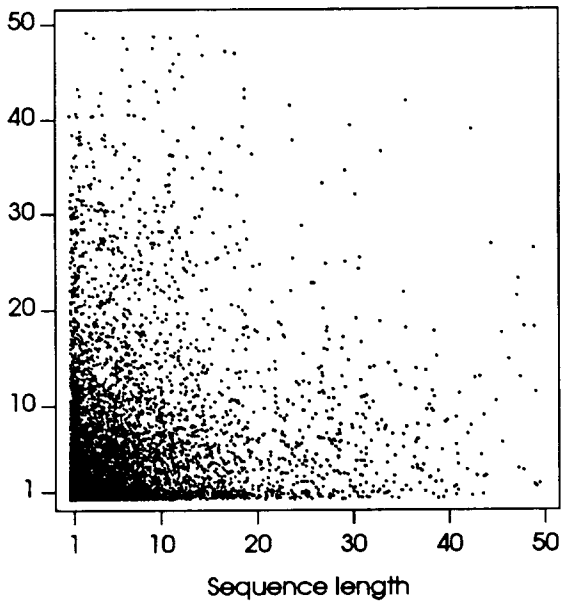
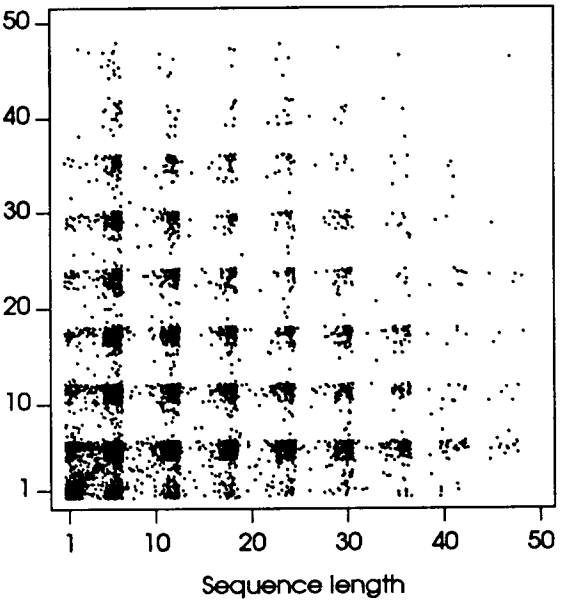


FIGURE 5.8. SCATTER PLOT OF LENGTHS OF SEQUENCES OF LONG PACKETS – *STATION 25*'S SEND QUEUE



We conclude that the series whose histograms are shown in Figures 5.1 through 5.4 are produced by discrete probability functions that decrease roughly exponentially and are statistically independent within themselves. We have also verified that they are independent of each other. A geometric probability function should provide a good approximation for these data. However, since *station 25* is a diskless client workstation, before we can claim that geometric density functions are appropriate for modeling the short- and long-packet sequence lengths of our arrival processes, we must verify that client workstations with local disks and file servers also satisfy the assumptions of geometric probabilities.

In Figures 5.9 through 5.12 we show histograms of long- and short-packet sequence lengths for file server *station 21*. The histogram plots of the short-packet sequence lengths are characterized by steeply decreasing curves rather than straight lines in the region of small values. This higher proportion of short sequence lengths is caused by network contention. In the case of the receive queue, concurrent file accesses by different client workstations may result in a client inserting a large packet into a sequence of short packets transmitted by another client. In the case of the send queue, the concurrent activity is caused by the various server processes that handle client requests. The presence of multiple server processes produces asynchronous traffic, which again results in the intermixing of long and short packets addressed to different clients.

Asynchronous processes also play a role in the packet arrival processes of client workstations. Workstation users run multi-window interfaces, and in each window they run separate processes that access the file server independently and asynchronously

FIGURE 5.9. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 21*'S – SEND QUEUE

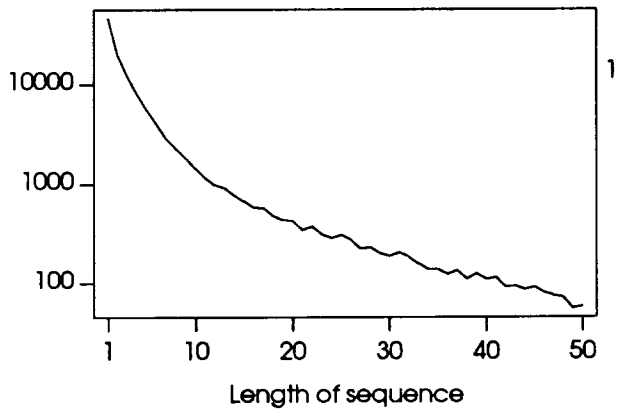


FIGURE 5.10. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 21*'S – SEND QUEUE

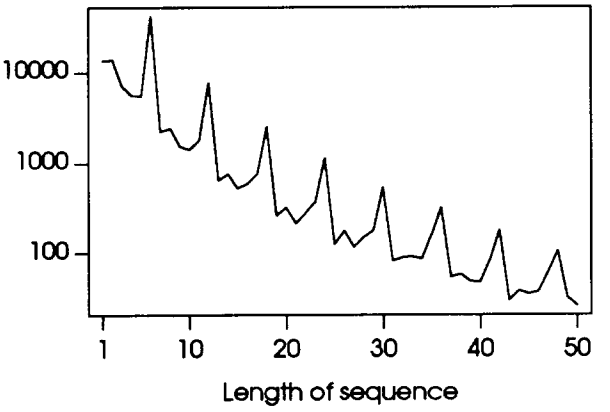


FIGURE 5.11. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 21*'S – RECEIVE QUEUE

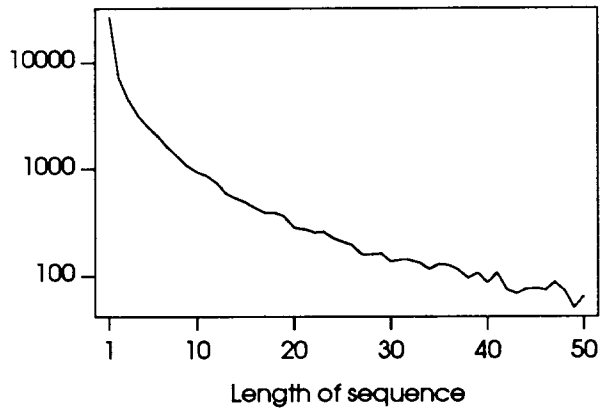
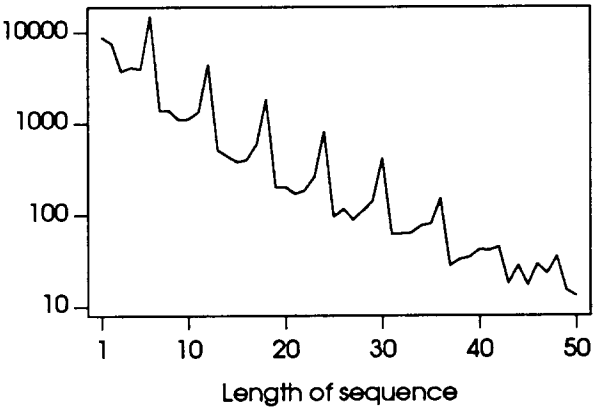


FIGURE 5.12. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 21*'S – RECEIVE QUEUE



with respect to each other. Returning to Figures 5.1 and 5.3 for a moment, we see that in fact the curves in these two graphs also deviate slightly from a straight line in the short length region.

Figures 5.10 and 5.12 prompt two observations: first, these two curves (or, more precisely, their peaks) follow very nearly a straight line. Second, in comparison with Figures 5.2 and 5.4, there is a large proportion of sequence lengths occurring at values other than multiples of six. This is explained by the concurrency among different clients simultaneously communicating with the file server.

In Figures 5.13 through 5.16 we show the histograms of short- and long-packet sequence lengths of *station 11*, a client workstation with local disks. From these plots it is obvious that the "straight line" model does not apply to this machine. We have already observed in Chapter 3 (see Sections 3.IV and 3.VII) that *station 11* has a highly nonstationary behavior; unfortunately, such nonstationarity characterizes the interarrival times of most clients with local disks. Local disks are typically used as a swap partition and as a cache for frequently accessed files. Hence, if they are sufficiently large, these disks can reduce significantly the network traffic, resulting in sparse, highly irregular arrival

processes.

FIGURE 5.13. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 11* – SEND QUEUE

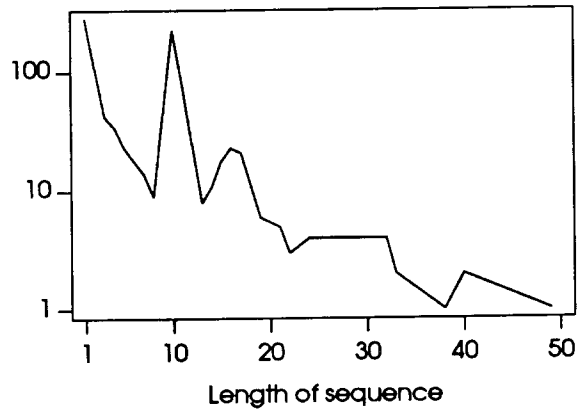


FIGURE 5.14. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 11* – SEND QUEUE

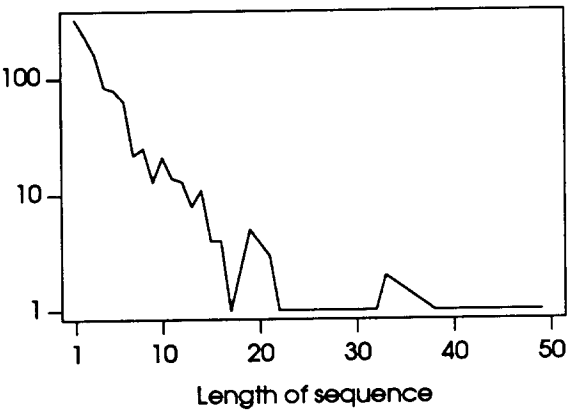


FIGURE 5.15. HISTOGRAM OF SHORT-PACKET SEQUENCE LENGTHS *STATION 11* – RECEIVE QUEUE

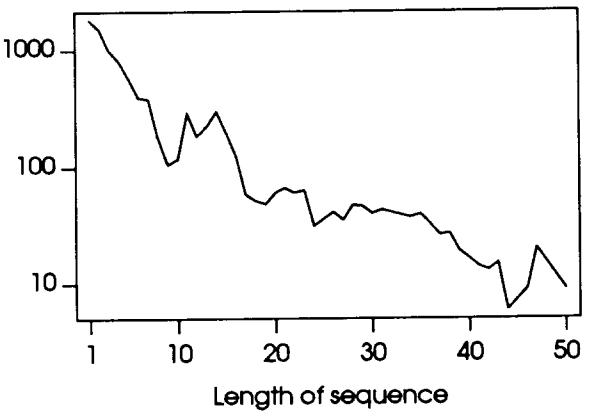
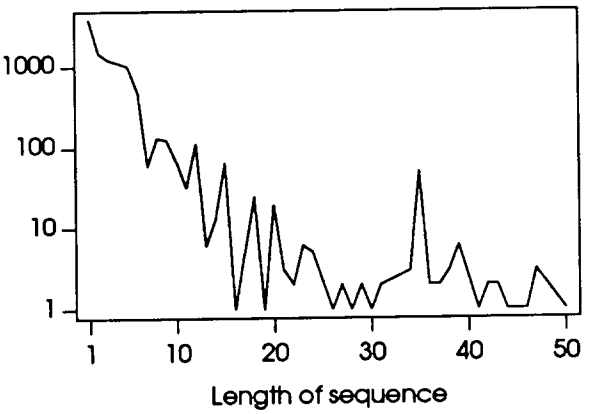


FIGURE 5.16. HISTOGRAM OF LONG-PACKET SEQUENCE LENGTHS *STATION 11* – RECEIVE QUEUE



The fundamental difference between packet arrival processes generated by client workstations with local disks and those generated by diskless clients is that in the former the ratio of large packets to small packets is much lower than in the latter. Long packets, with their short interarrival times and low variability, tend to stabilize the arrival processes of diskless machines. The relatively large number of short packets, with their longer interarrival times and much higher variability, in the processes produced by disk-based clients, coupled with low packet arrival rates, results in unstable processes.

In Figures 5.17 and 5.18 we show the first 100 autocorrelation coefficients of short- and long-packet sequence lengths of the process generated by the *station 11*'s send queue. As in the other correlograms, we indicate within dashed lines the interval where with probability 0.95 the correlation coefficients of an uncorrelated process would lie. The confidence band is larger than in the previous cases because *station 11*'s send-queue arrival process is composed of many fewer arrivals. These graphs show that the sequence lengths of both short and long packets are essentially independent processes: the peak in Figure 5.17 around lag 63 is spurious. Indeed, it is produced by a small group

of relatively long sequences and by a single long sequence 63 sequences apart from the group. Since the sequences are independent and we have no other reason to believe that the straight line model would not (approximately) apply were *station 11*'s arrival process of higher intensity, we will consider this model valid also for clients with local disks.

FIGURE 5.17. CORRELOGRAM FOR THE SERIES OF LENGTHS OF SHORT PACKET SEQUENCES – *STATION 11*'S SEND QUEUE

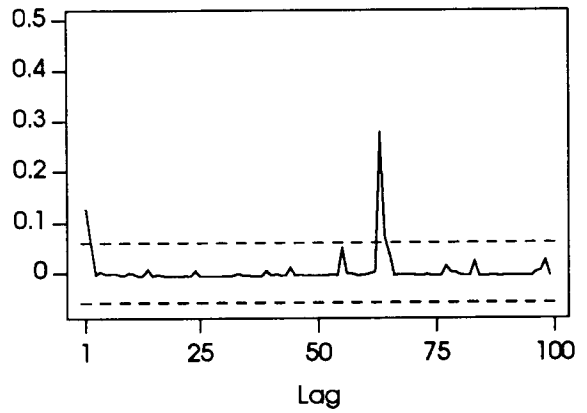
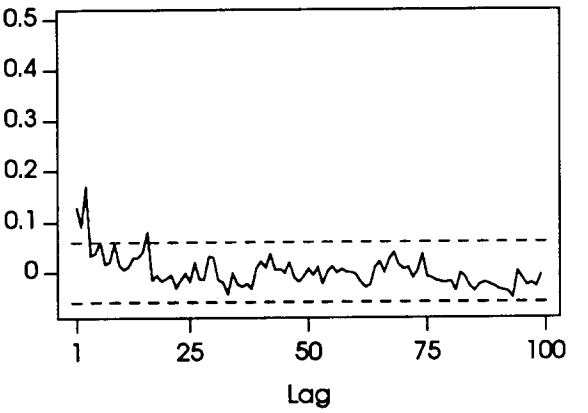


FIGURE 5.18. CORRELOGRAM FOR THE SERIES OF LENGTHS OF LONG PACKET SEQUENCES – *STATION 11*'S SEND QUEUE



Given that sequences of short packets alternate with sequences of long packets, that the two types of sequences are independent of each other, and that both can be approximated by geometric distribution functions, the most natural choice for a mathematical model of these sequences is a two-state discrete-time Markov chain. We have estimated the transition probabilities for the processes generated by the send and receive queues of a number of workstations. These probabilities are shown in Tables 5.1 and 5.2.

TABLE 5.1. CONDITIONAL TRANSITION PROBABILITIES – SEND QUEUES

WORKSTATION	MATRIX ELEMENTS			
	p_{11}	p_{12}	p_{21}	p_{22}
station 2	0.865	0.135	0.076	0.924
station 3	0.868	0.132	0.079	0.921
station 7 (s)	0.910	0.090	0.075	0.925
station 8	0.887	0.113	0.063	0.937
station 11	0.993	0.007	0.242	0.758
station 13	0.910	0.090	0.082	0.918
station 14	0.828	0.172	0.091	0.909
station 15	0.913	0.087	0.048	0.952
station 16	0.946	0.054	0.072	0.928
station 20	0.901	0.099	0.184	0.816
station 21 (s)	0.840	0.160	0.136	0.864
station 22	0.927	0.073	0.019	0.981
station 23 (s)	0.753	0.247	0.071	0.929
station 25	0.838	0.162	0.085	0.915
station 26	0.966	0.034	0.020	0.980
station 27	0.984	0.016	0.004	0.996

We will use these transition probability matrices as the embedded two-state Markov chains of our semi-Markov models. One state corresponds to the generation of sequences of short packets, the other to the generation of sequences of long packets.

TABLE 5.2. CONDITIONAL TRANSITION PROBABILITIES – RECEIVE QUEUES

WORKSTATION	MATRIX ELEMENTS			
	p_{11}	p_{12}	p_{21}	p_{22}
station 2	0.789	0.211	0.065	0.935
station 3	0.793	0.207	0.058	0.942
station 7 (s)	0.968	0.032	0.077	0.923
station 8	0.954	0.046	0.038	0.962
station 11	0.936	0.064	0.256	0.744
station 13	0.863	0.137	0.073	0.927
station 14	0.674	0.326	0.063	0.937
station 15	0.938	0.062	0.028	0.972
station 16	0.673	0.327	0.019	0.981
station 20	0.971	0.029	0.117	0.883
station 21 (s)	0.916	0.084	0.115	0.885
station 22	0.883	0.117	0.041	0.959
station 23 (s)	0.848	0.152	0.095	0.905
station 25	0.691	0.309	0.063	0.937
station 26	0.890	0.110	0.027	0.973
station 27	0.978	0.022	0.014	0.986

In each state we will use two probability density functions to generate interarrival times and packet lengths. Our desire to define the models based on properties of the data has led us to modify the traditional semi-Markov model. This is discussed in the next section.

III. A Generalized Semi-Markov Process

The semi-Markov process was studied in the late fifties and early sixties by Smith [82] and Pyke [72,73], among others. A description of its main properties can be found in those studies. A semi-Markov process contains an embedded Markov chain that determines transitions between states. In each state intervals are chosen independently and identically distributed according to a probability density function associated with that state. The independence of successive intervals in each state makes the derivation of the properties of a semi-Markov process particularly simple. However, it also limits its suitability as a model for our arrival processes. Were we to use a two-state semi-Markov model, one state would represent the generation of short packets and the other the generation of long packets. However, as shown in Chapter 3, contrary to the assumptions on which the semi-Markov model is based, in each of these states interarrival times would be correlated.

Our solution is to use a modified two-state semi-Markov model, which we will term a *generalized semi-Markov* model, that does not assume independence of intervals in each state. The generalized semi-Markov model instead assumes that intervals in each state have a non-zero covariance. It consists of a two-state Markov chain with state-transition matrix

$$\mathbf{P} = \begin{bmatrix} \alpha_1 & 1-\alpha_1 \\ 1-\alpha_2 & \alpha_2 \end{bmatrix}, \tag{5.1}$$

and with equilibrium state probability vector $\boldsymbol{\pi} = (\pi_1, \pi_2)$

$$\pi_1 = \frac{1-\alpha_2}{2-\alpha_1-\alpha_2}, \qquad \pi_2 = \frac{1-\alpha_1}{2-\alpha_1-\alpha_2}. \tag{5.2}$$

In state 1, interarrival times are sampled from probability density function $f_1(x)$; in state 2, from probability density function $f_2(x)$. Each of the two series of intervals will be correlated; however, we continue to assume that intervals in one state are independent of intervals in the other state. In addition, we assume that the process generated in each state is only suspended, not disrupted, by a transition to the other state; upon a transition back to the original state, the process resumes at the point at which it was interrupted.

Under the assumptions stated above, the probability density function of interarrival times produced by the generalized semi-Markov process is

$$f_X(x) = \pi_1 f_1(x) + \pi_2 f_2(x) ,$$

and the mean interarrival time and the interarrival time variance are

$$E(X_i) = \pi_1\mu_1 + \pi_2\mu_2 ,$$

$$\text{var}(X_i) = \pi_1\sigma_1^2 + \pi_2\sigma_2^2 + \pi_1\pi_2(\mu_1 - \mu_2)^2 , \tag{5.3}$$

in which μ_1 and μ_2 are the means of the intervals generated in states 1 and 2, and σ_1^2 and σ_2^2 are the interval variances of the two states. We will next derive formulas for the covariance of intervals, correlation coefficients, and index of dispersion for intervals.

Let us use u to denote one state and $\bar{u}$ to denote the other state. If we consider two intervals X_i and X_{i+k} and denote with (v, w), where v and w are states, the fact that X_i is sampled from $f_v(x)$ and that X_{i+k} is sampled from $f_w(x)$, we have

$$E(X_iX_{i+k}\,|\,(u, u)) = \mu_u^2 + C_u^k ,$$

where C_u^k is a function, dependent on k, of the covariance of the intervals produced in state u. We also have

$$E(X_iX_{i+k}\,|\,(u, \bar{u})) = \mu_u\mu_{\bar{u}}$$

because intervals in one state are independent of the intervals in the other state.

We can express C_1^k and C_2^k in terms of the covariances of functions $f_1(x)$ and $f_2(x)$, which we will indicate with $\text{cov}_1(X_i, X_j)$ and $\text{cov}_2(X_i, X_j)$. Let us consider a sequence of $k+1$ Markov-chain states, $s_0s_1 \cdots s_k$, beginning and ending in the same state u, that is with $s_0 = u$ and $s_k = u$. Since we have assumed that $f_1(x)$ and $f_2(x)$ each evolve independently of any transitions to the other state, C_u^k will equal $\text{cov}_u(X_i, X_{i+m})$ with m equal to the number of occurrences of state u, in the Markov-chain sequence, minus 1. Hence, the range of m will be between 1 and k: 1 when the initial and the last states are the only occurrences of state u, and k when all the states in the sequence are u.

We will indicate with $P_{m,n}^{v,w}$ the probability that a sequence of length m, beginning in state v and ending in state w, contains n occurrences of state w. These probabilities can be easily expressed recursively:

$$P_{m,n}^{u,u} = \begin{cases} \alpha_u P_{m-1,n-1}^{u,u} & (n = m) \\ (1-\alpha_u)P_{m-1,n-1}^{\bar{u},u} + \alpha_u P_{m-1,n-1}^{u,u} & (2 < n < m) \\ (1-\alpha_u)P_{m-1,n-1}^{\bar{u},u} & (n = 2) \end{cases}$$

$$P_{m,n}^{\bar{u},u} = \begin{cases} (1-\alpha_{\bar{u}})P_{m-1,n}^{u,u} & (n = m-1) \\ \alpha_{\bar{u}}P_{m-1,n}^{\bar{u},u} + (1-\alpha_{\bar{u}})P_{m-1,n}^{u,u} & (1 < n < m-1) \\ \alpha_{\bar{u}}P_{m-1,n}^{\bar{u},u} & (n = 1) \end{cases}$$

with the following set of initial conditions

$$P_{2,2}^{1,1} = \alpha_1 , \quad P_{2,1}^{1,2} = 1-\alpha_1 , \quad P_{2,1}^{2,1} = 1-\alpha_2 , \quad P_{2,2}^{2,2} = \alpha_2 .$$

We can now express C_u^k in terms of the covariance of function $f_u(x)$

$$C_u^k = \sum_{q=1}^{k} P_{k+1,q+1}^{u,u} \operatorname{cov}_u(X_l, X_{l+q}) .$$

The unconditional probability that beginning in state v the Markov chain will be in state w after k steps is

$$\Pr(s_k = w, s_0 = v) = \Pr(s_k = w \mid s_0 = v)\Pr(s_0 = v) = p_{v,w}^k \pi_v .$$

We have indicated with $p_{v,w}^k$ the probabilities $\Pr(s_k = w \mid s_0 = v)$, that is, the elements of $\mathbf{P}^k$:

$$\mathbf{P}^k = \begin{bmatrix} \pi_1 + \pi_2\beta^k & \pi_2 - \pi_2\beta^k \\ \pi_1 - \pi_1\beta^k & \pi_2 + \pi_1\beta^k \end{bmatrix},$$

where $\beta = \alpha_1 + \alpha_2 - 1$. Then, the expectation $E(X_lX_{l+k})$ can be computed as

$$E(X_lX_{l+k}) = \pi_1 p_{1,1}^k (\mu_1^2 + C_1^k) + \pi_1 p_{1,2}^k \mu_1\mu_2 + \pi_2 p_{2,1}^k \mu_1\mu_2 + \pi_2 p_{2,2}^k (\mu_2^2 + C_2^k) ,$$

which can be expanded into

$$E(X_lX_{l+k}) = \pi_1(\pi_1 + \pi_2\beta^k)(\mu_1^2 + \sum_{q=1}^{k} P_{k+1,q+1}^{1,1} \operatorname{cov}_1(X_l, X_{l+q})) +$$

$$+ \pi_1(\pi_2 - \pi_2\beta^k)\mu_1\mu_2 + \pi_2(\pi_1 - \pi_1\beta^k)\mu_1\mu_2 +$$

$$+ \pi_2(\pi_2 + \pi_1\beta^k)(\mu_2^2 + \sum_{q=1}^{k} P_{k+1,q+1}^{2,2} \operatorname{cov}_2(X_l, X_{l+q})) .$$

We can now evaluate the covariance of the generalized semi-Markov process

$$\operatorname{cov}(X_l, X_{l+k}) = E(X_lX_{l+k}) - E^2(X_l) =$$

$$= (\mu_1 - \mu_2)^2\pi_1\pi_2\beta^k +$$

$$+ \pi_1(\pi_1 + \pi_2\beta^k) \sum_{q=1}^{k} P_{k+1,q+1}^{1,1} \operatorname{cov}_1(X_l, X_{l+q}) +$$

$$+ \pi_2(\pi_2 + \pi_1\beta^k) \sum_{q=1}^{k} P_{k+1,q+1}^{2,2} \operatorname{cov}_2(X_l, X_{l+q}) . \qquad (5.4)$$

Observe that the covariance depends not only on the covariances of the two component processes, but also on the difference between the means μ_1 and μ_2.

The series of correlation coefficients is obtained using the relationship $\rho_k = \operatorname{cov}(X_l, X_{l+k}) / \operatorname{var}(X)$, and the index of dispersion for intervals using Equation (4.3):

$$J_n = \frac{1}{\pi_1\mu_1 + \pi_2\mu_2}\Big[\pi_1\sigma_1^2 + \pi_2\sigma_2^2 + \pi_1\pi_2(\mu_1 - \mu_2)^2 + 2\sum_{k=1}^{n-1}(1 - \frac{k}{n})\Big[(\mu_1 - \mu_2)^2\pi_1\pi_2\beta^k + \pi_1(\pi_1 + \pi_2\beta^k)\sum_{q=1}^{k} P_{k+1,q+1}^{1,1}\,\mathrm{cov}_1(X_l, X_{l+q}) + \pi_2(\pi_2 + \pi_1\beta^k)\sum_{q=1}^{k} P_{k+1,q+1}^{2,2}\,\mathrm{cov}_2(X_l, X_{l+q})\Big]\Big].$$

We will next attempt to verify the ability of the generalized semi-Markov model to capture the variability of our packet arrival processes.

IV. Data Fitting and Analysis

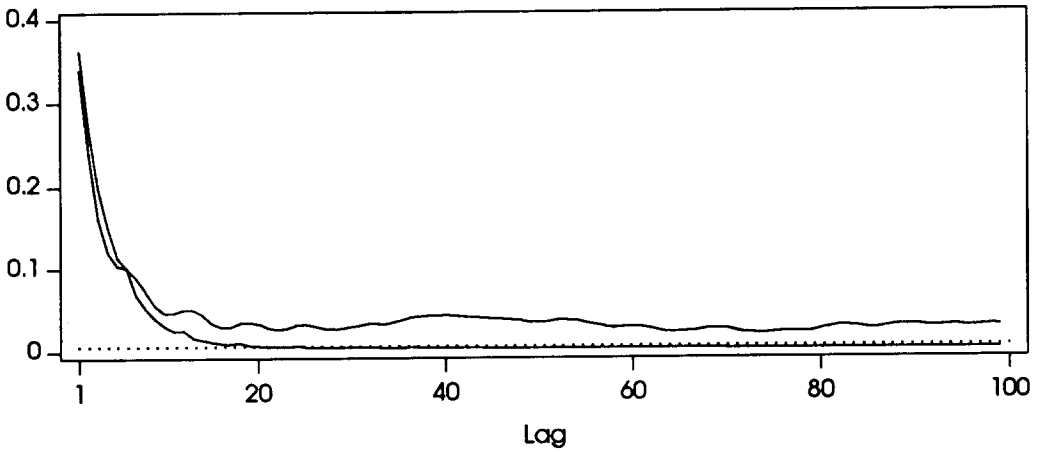
In Figure 5.19 we plot the first 100 autocorrelation coefficients of the interarrival-time series of *station 25*'s send queue and we also plot the autocorrelation coefficients produced by the generalized semi-Markov process using the covariances of *station 25*'s series. To compute the model's correlogram, we have separated *station 25*'s interarrival times into two series: one of interarrival times associated with short packets, the other of the interarrival times associated with long packets. (As mentioned earlier in Section I, we associate an interarrival time to the first of the two packets that produced it.) We have then computed the autocovariances of each series separately, and fed the coefficients into Equation (5.4). Although the two curves are close to each other for small values of the lag, there is a clear divergence between the model and the data points for lag values larger than 20: the data coefficients are above the 99 percent confidence band (indicated by the dotted line), whereas the model's coefficients are below it. Although the difference may seem negligible, in the calculation of the index of dispersion for intervals the effect of tiny coefficients is quite significant as a point at lag j on the IDI curve is constructed by adding together all of the correlation coefficients for lags smaller than j. To understand why data and model diverge, we will generate artificial component processes with controllable parameters, and compare the estimated autocorrelation coefficients of the process obtained combining the two component processes with the autocorrelation coefficients predicted by the generalized semi-Markov model.

We will construct the point processes from two component time series, which we assume are generated in state 1 and state 2 of the embedded Markov chain. We will produce autocorrelated interarrival times for the component time series using autoregressive functions of the form

$$Y_{l+1} = (1 - \gamma)g(\bar{\mu}) + \gamma Y_l\,, \tag{5.5}$$

where $g(\bar{\mu})$ is an exponential random variable with mean $\bar{\mu}$ and γ is between 0 and 1 [10]. In all of the examples in the remainder of this section we will compare the correlation coefficients of the simulated data with the correlation coefficients produced by the model. In order to focus on the effects of the correlations in the two component processes, we will generate components with the same mean so that the first term in Equation (5.4) will be zero.

FIGURE 5.19. AUTOCORRELATION COEFFICIENTS OF *STATION 25*'S SEND QUEUE AND MODEL



In Figure 5.20 we present a process that satisfies exactly all the model's requirements: we have two series that are mutually independent, each of the series is autocorrelated, the two series are mixed using a two-state Markov chain that determines the lengths of the sequences of interarrival times in each state, and transitions out of a state do not lose state memory. As we can see, there is a good match between the correlogram of the constructed process and that of the model evaluated using parameters from the component processes. (The small oscillations are caused by stochastic variations in the random number generator that produced the time series.)

To see whether correlations between the two components generate the difference seen in Figure 5.19, we evaluate the model using a process with two component time series that are mutually correlated. We generate one time series Y_i according to the autoregressive model shown above, and the second using the relationship

$$W_{i+1} = (1 - \theta)Y_i + \theta W_i .$$

We set the various coefficients so that the amount of cross correlations between the two component variables is significant. We show the first 100 correlation coefficients in Figure 5.22, where we also indicate the confidence band in which, with probability 0.95, the correlation coefficients would lie were the processes independent. In Figure 5.21 we graph the correlogram of the constructed process and that of the model evaluated using parameters from the component processes. Because the two curves are so close to each other, we conclude that the model is relatively independent of cross correlation between the two component functions.

We next test whether the clustering displayed in Figure 5.2 and 5.4 has an effect on the model's correlation structure. To do this, we generate a stream of interarrival times in which sequences of short packets are geometrically distributed and sequences of long packets are exactly six packets long. Within each sequence, the interarrival times are of the form (5.5). No major effect is discernible in the correlogram, shown in Figure 5.23; we, hence, conclude that packet clustering is not causing the divergence between the data and the generalized semi-Markov model in Figure 5.19.

FIGURE 5.20. DATA PRODUCED BY TWO FUNCTIONS SATISFYING ALL MODEL'S ASSUMPTIONS

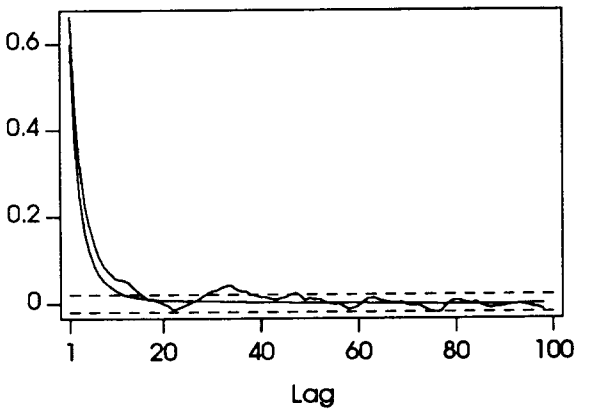


FIGURE 5.21. DATA PRODUCED BY TWO CROSS-CORRELATED FUNCTIONS

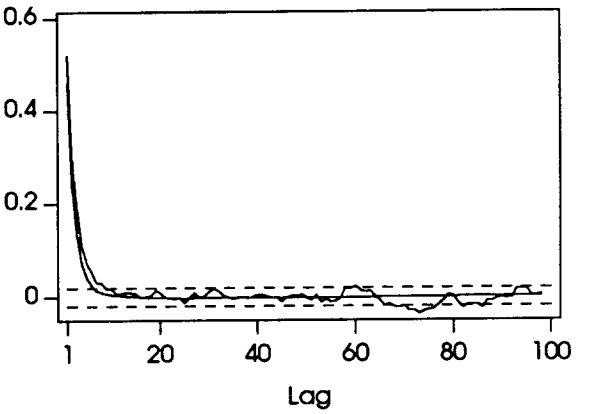
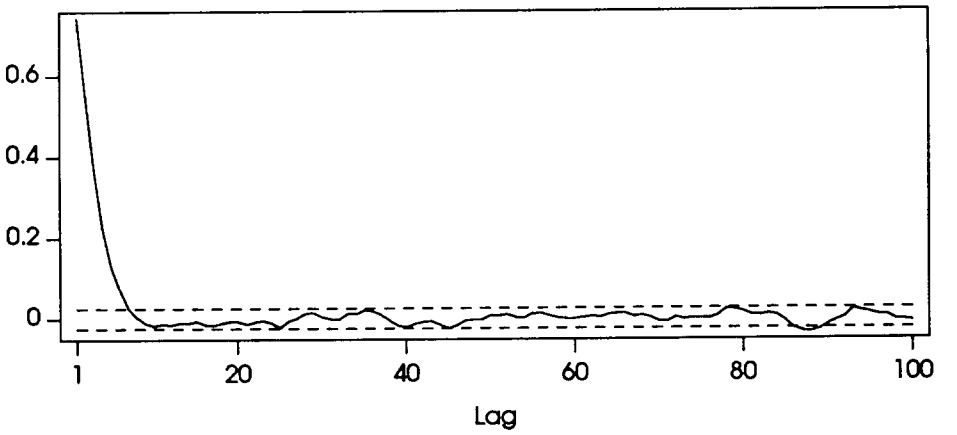


FIGURE 5.22. CROSS CORRELATION COEFFICIENTS OF COMPONENT PROCESSES



We, finally, experiment with nonstationary data in Figure 5.24. We produce two independent, moderately nonstationary series by slowly increasing the mean $\bar{\mu}$ in the autoregressive model (5.5). Now we see that the correlogram of the data decreases very slowly, and, indeed, lies outside the 95-percent confidence interval. As one could have guessed, the model's output is instead more regular and quickly positions itself inside the confidence band. (Recall that the confidence interval delimits the region where, in this case with probability 0.95, all the coefficients would be were not the process significantly autocorrelated.) As in Figure 5.19, here too what appears a small difference between the correlation coefficients is greatly amplified during the computation of the index of dispersion for intervals. Although we have produced these data with simple, linearly increasing trends, while, as discussed in Chapter 3, nonstationarities in our arrival processes cannot be classified as trends, the effects of packet arrival nonstationarities on the generalized semi-Markov model follow the same patterns.

FIGURE 5.23. DATA PRODUCED WHEN THE PROBABILITY DISTRIBUTION IN ONE OF THE STATES IS DETERMINISTIC

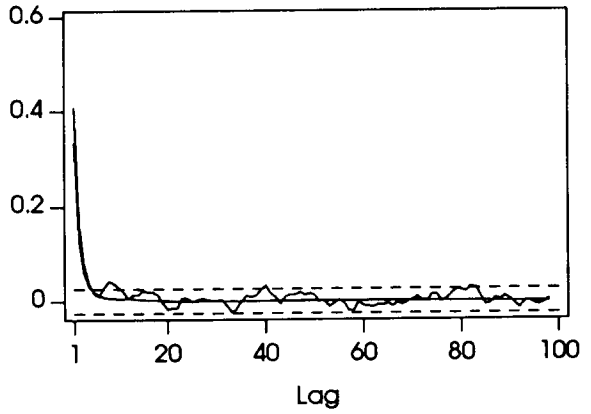
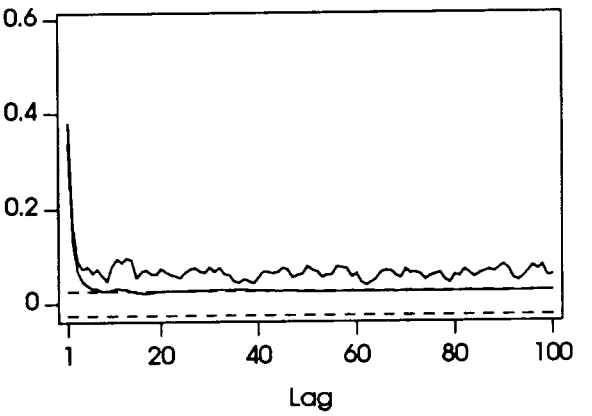


FIGURE 5.24. DATA PRODUCED BY NONSTATIONARY COMPONENT PROCESSES



We have come to realize that much of the medium- and long-term variability of packet arrival processes, as revealed by the index of dispersion for intervals, is a direct effect of the processes' slowly decreasing autocorrelation coefficients, and is caused by nonstationary process components. In Section 3.V, we argued that is it rather difficult to isolate and remove these components from our arrival processes. As a result, we are forced to reject the generalized semi-Markov model as a model appropriate for the representation of packet arrival processes that are nonstationary over long time scales. Nevertheless, in the next section we will provide approximations, based on our extended semi-Markov process, that are valid for short-term variability.

V. Short-Interval Approximations

It is unfortunate, though not quite unexpected, that our generalization of the semi-Markov process performs poorly with nonstationary processes. However, in interactive systems such as the queuing systems found in workstation networks, the system behavior over short intervals is perhaps the key performance indicator. These systems are designed to be highly responsive, and queue lengths must be kept short. Hence, the queuing behavior is associated to the statistical interactions among packet arrivals for small lag values. Although the high values of the asymptotes in the index of dispersion curves shown in Chapter 4 indicate that interval and count processes are affected by considerable long-term variability, one could argue that long-term arrival-process fluctuations have no effects on queues if the system manages to keep queue lengths short. In addition, as shown in Figure 5.19, the model fit over small lags is acceptable (and more evident if we smooth the data autocorrelation coefficients). Therefore, we propose to use the generalized semi-Markov model to represent the variability of the arrival processes over short intervals. (Notice that a simpler semi-Markov process would not suffice as a suitable model since its autocorrelation coefficients are of the form Kb^n, with both K and b less than 1, which hence decrease too fast [18].) In order to fit the model to our arrival processes, in addition to the transition probabilities α_1 and α_2 of matrix (5.1) (that is p_{11} and p_{22} in Tables 5.1 and 5.2), we need to provide the mean and variance of the two component processes, as well as their autocovariances.

It is infeasible to express the correlation structure of the two component processes in terms of their autocovariance coefficients, as we have done in the development of the theory in Section III. For ease of use, here we will instead provide algebraic approximations based on correlation coefficients. (We chose the correlation coefficients for it is easier to work with their normalized scale; for each process, the autocovariances can be obtained by multiplying the autocorrelation coefficients by the variance of the process found in Table 5.3 below.) It turns out that the autocorrelation functions of both component processes are well approximated by straight lines, as the examples in Figures 5.25 and 5.26 show. The approximation holds for all types of workstations: diskless, with local disks, file servers. Figure 5.25 shows the autocorrelation coefficients of interarrival times between short packets and, as an almost horizontal line near 0, the autocorrelation coefficients of interarrival times between long packets for the send queue of client *station 3*. Figure 5.26 presents the same data for the two component processes produced by client *station 27*, a workstation with a local disk.

FIGURE 5.25. AUTOCORRELATION COEFFICIENTS OF PROCESS COMPONENTS OF *STATION 3*'S SEND QUEUE

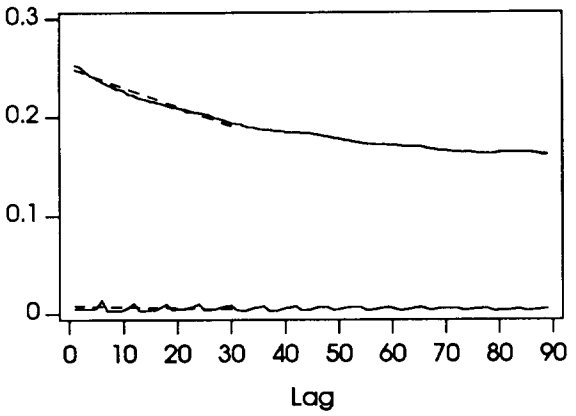
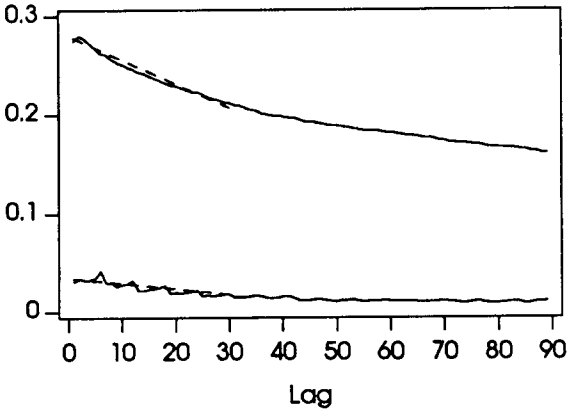


FIGURE 5.26. AUTOCORRELATION COEFFICIENTS OF PROCESS COMPONENTS OF *STATION 27*'S RECEIVE QUEUE



The linear approximations that we will tabulate for the autocorrelation coefficients of interarrival times in Tables 5.1 and 5.2 are valid only between lags from 1 to 30. (Notice that we have excluded the lag-0 coefficient, which is always 1.) The time spans of the linear models range from about 218 ms to 517 ms depending on the mean interarrival times of the various workstations. These intervals, outside which we suggest not to use our approximation models, may seem small. However, only very rarely do we observe queue lengths of size 30 or more. Hence, the dynamics of the queues of workstation systems should be captured by the generalized semi-Markov model truncated to these time scales.

Table 5.3 lists estimated values of the means and variances of the processes produced by the send queues of a representative set of workstations. As expected, in general faster architectures produce shorter mean interarrival times. The table also shows the estimated values of the mean and variance for each of the two component processes: the short-packet interarrival-time process, indicated by subscript 1, and the long-packet one, with subscript 2. Observe how much smaller the mean interarrival times of long packets are than those of short packets. However, their variances display a

much wider range than the variances of short-packet interarrival times. This is related to the long interarrival times that we have classified as "LastFragment/1500". As we can see, the variance is higher in machines with local disks, an indication that these workstations transfer longer data sets, and thus produce more "LastFragment/1500" interarrival times.

TABLE 5.3. MEANS AND VARIANCES OF MEASURED PROCESSES – SEND QUEUES

Means in ms Variances in ms^2 Errors in percentage

WORKSTATION	AGGREGATE PROCESS				COMPONENT PROCESSES			
	mean	mean error	var	var error	$mean_1$	var_1	$mean_2$	var_2
station 2	12.391	–0.044	325.983	–0.039	26.558	415.926	4.407	98.417
station 3	15.909	–0.028	414.982	–0.018	34.028	431.696	5.058	90.632
station 7	11.004	–5.878	231.408	–3.693	16.853	270.671	4.944	118.554
station 8	13.053	–0.297	347.386	–0.241	26.024	459.565	5.761	136.567
station 11	16.168	–0.373	323.939	0.087	16.348	322.958	7.799	296.829
station 13	14.683	–6.790	334.699	–4.438	24.069	377.069	4.226	79.991
station 14	12.598	0.237	339.101	0.209	28.599	457.337	4.178	71.276
station 15	12.406	0.281	408.470	0.038	22.112	453.597	7.105	303.740
station 16	17.251	–0.691	375.375	–0.302	27.420	368.034	3.414	53.207
station 20	14.312	–0.043	362.179	–0.045	19.039	396.539	5.509	178.828
station 21	11.816	9.165	244.905	5.694	20.366	291.204	6.552	143.674
station 22	9.005	0.131	313.545	0.161	31.185	729.676	3.247	44.675
station 23	8.552	15.094	229.607	12.800	25.264	352.720	5.410	144.048
station 25	12.559	0.778	346.452	0.678	29.262	468.933	3.944	65.182
station 26	10.152	–0.775	254.886	–0.669	20.852	372.369	3.733	74.529
station 27	7.278	–4.771	140.489	–3.363	14.778	183.706	4.969	104.536

Using the transition probabilities of Table 5.1 we can compute the equilibrium state probabilities (5.2), and with them the model's mean and variance from Equations (5.3). In the columns "mean error" and "var error" in Table 5.3, we list the percent errors on the model's predictions. Except for one workstation, the percent errors are in absolute value smaller than 10, proving again that the Markov property is satisfied in our data.

In order to evaluate the generalized semi-Markov models, we also need the autocovariances of the component processes. As mentioned earlier, we will provide first-order polynomial formulas (i.e. equations of straight lines) that interpolate accurately the first 30 autocorrelation coefficients. In order to obtain the covariances, one will have to multiply the correlation equations by the values of the (aggregate) process variances found in Table 5.3. In Table 5.4, we list the coefficients of interpolating linear equations of the form $y = mx + q$. (Again, with subscript 1 we label quantities that refer to short-packet processes.) If both coefficients are very small, a situation that indicates statistical independence between subsequent arrivals, we enter zeros in the table.

VI. Models of Packet Lengths

In order to complete our data characterization, which, as mentioned in the Introduction, is based on a two-dimensional model, we now describe model approximations for lengths of small and large packets. Several observations are in order. First, large NFS packets have a very simple structure: they come in groups of six, the first five fragments in a group are 1500 bytes long and the last fragment is somewhat shorter, for instance 920 bytes. Groups can follow each other, but as shown in Figures 5.2 and 5.4 and as discussed in Section II, the probability that k groups occur consecutively decreases

TABLE 5.4. CORRELATION COEFFICIENT INTERPOLATION LINES – SEND QUEUES

WORKSTATION	EQUATION COEFFICIENTS			
	m_1	q_1	m_2	q_2
station 2	-0.00218	0.193	0	0
station 3	-0.00204	0.255	0	0
station 7	-0.00229	0.19	-0.000179	0.0154
station 8	-0.00368	0.159	-0.000179	0.0214
station 11	-0.00025	0.235	-0.000107	0.0242
station 13	-0.00279	0.227	0	0
station 14	-0.00211	0.141	-0.000107	0.0202
station 15	-0.00164	0.342	-0.00196	0.187
station 16	-0.00114	0.121	-0.000393	0.0188
station 20	-0.000357	0.0907	-0.00132	0.144
station 21	-0.00261	0.169	0	0
station 22	-0.00129	0.548	-0.000214	0.0214
station 23	-0.00129	0.0936	0	0
station 25	-0.00225	0.117	-0.000536	0.0221
station 26	-0.00179	0.175	-0.000429	0.0459
station 27	-0.00457	0.328	0	0

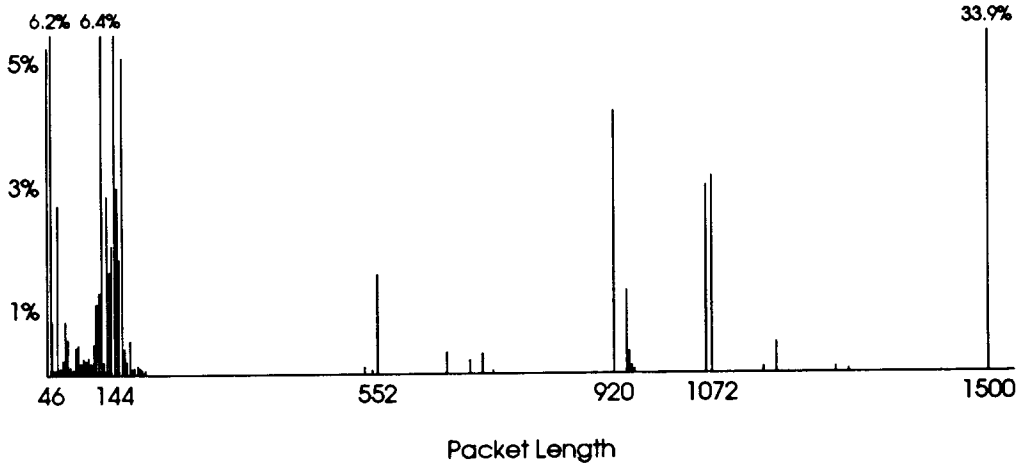
geometrically as k increases. This structure produces correlation at values that are multiples of six.

Second, the other components in the distribution of large packets, TCP and ND packets, which are shown in Figure 2.15, at 1064 and 1072 bytes respectively, are irrelevant. ND, the Network Disk protocol, is being phased out and, although it produces about 4 percent of the packet count (including also small packets), among the workstations that we have considered only *station 1* uses it. Long TCP packets account for about 3 percent of the total, but they also occur in groups and hence, rarely mix with NFS packets. We will ignore both TCP and ND packets.

Third, some complications occur with file servers as their traffic is the superposition of traffic simultaneously directed to or coming from several client workstations. This will have the effect of making the lengths of sequences of 1500-byte packets variable, as discussed in Section II with reference to Figures 5.10 and 5.12. As a result of these considerations, we propose three possible models, with varying degrees of approximation, for long packet sizes: the first model would produce one 920-byte packet every five 1500-byte packets; the second model is a mean/standard-variation model with mean 1373 bytes and standard variation 247 bytes; and the third model, a special case of the second, would produce a constant packet of size 1373.

Fourth, the structure of the short-packet distribution is rather complicated. The sequence of short-packet lengths is strongly autocorrelated, as contiguous packets tend to have similar lengths. However, these correlations lose importance if they are put in perspective. In Figure 5.27 we reproduce the aggregate packet length distribution on a linear scale. From this graph we can see that the autocorrelation within the series of short packets, which are all concentrated at one end of the figure, amounts to small variations in packet sizes when viewed from the other end. Thus, we will ignore the fine structure produced by occurrences of groups of short packets of similar lengths, and describe short packets simply by their means and variances. In most cases there are no significant differences among the various workstations, hence we will simply provide the mean short packet and the short-packet standard variation based on the aggregate traffic. They

FIGURE 5.27. PERCENTAGE OF PACKETS VS. PACKET LENGTH



are 119 bytes and 50 bytes, respectively.

VII. Summary

In this chapter we have constructed models of packet arrival processes using properties of the data. We focused on interarrival times, since we felt that packet lengths needed to be represented explicitly, which is not possible packet count models. We found that sequences of short and long packets are modeled accurately by two-state discrete-time Markov chains. In order to represent correlations among arrivals in each of the states, we have extended the semi-Markov model and derived formulas for the autocovariances, and index of dispersion of intervals.

Unfortunately, the generalized semi-Markov model does not reproduce accurately the autocorrelation coefficients for medium and large lag values. Nonetheless, we provided model approximations based on linear interpolation of autocorrelation coefficients of the component processes that are valid and accurate for small lag values.

6 Conclusions

In this dissertation we have investigated the variability of packet-arrival processes produced by individual engineering workstations on an Ethernet local-area network. Chapter 2 presented the measurement methodology, the measurement error analysis, and a simple description of the network's aggregate traffic in terms of network load, packet length distribution, and interarrival time distribution. Chapter 3 dealt with the stochastic analysis of interarrival time series. We focused on the second-order properties as they provide the basis of our definition of variability. In Chapter 4 we defined variability in terms of indices of dispersion for intervals and counts. We estimated these indices for a number of workstations, and discussed their properties. In addition, with an example based on the rather artificial Markov-modulated Poisson process, we illustrated a way to introduce variability into analytical models. Finally, in Chapter 5 we derived a model of variability whose structure follows the structure of the data: short and long packets produce disjoint sets of interarrival times; lengths of sequences of short and long packets form a discrete-time Markov chain, a generalized semi-Markov process provides a good approximation to the autocorrelation coefficients of interarrival times for short time scales. The generalized semi-Markov process approximation requires only estimates of the first- and second-order moments of interarrival times. To complete the model, we also provided simple characterizations of short and long packet lengths. Because of the recursive nature of the model's equations, the model is suited for in simulation studies.

The relevance of the dissertation lies in the prominence of the type of computing system that we have analyzed: personal engineering workstations represent one of the most successful products developed by the computer industry over the last decade. Because of this prominence, researchers and computer analysts alike are always looking for data about system usage and modeling examples. Our analysis is quite specific, as we developed the notion of variability and tailored traffic models to reproduce it. Computer communication research is proceeding apace toward future Gigabit networks. These networks will require knowledge of the statistical behavior of data streams if they are to switch data efficiently. Analyzing the variability of arrival processes with indices of dispersion can provide some of that knowledge. Models of variability can become useful as input generators to gigabit network simulation studies.

II. Future Work

In spite of the wide range of material presented in the previous chapters, we have just scratched the surface of the possible research topics that can be identified around the issues of stochastic analysis and modeling of packet arrival processes. Here, we address briefly, in the form of research questions, some of the work that remains to be done.

By far the most interesting and pressing question is the relationship between our models of variability and queuing behavior. A key notion is whether the two-moment approximation of the generalized semi-Markov process is sufficient to guarantee an accurate queuing performance. What is the relationship between index of dispersion curves and queuing? Are there simple rules that relate the initial slope, curve knee, and asymptote of the indices to, say, waiting times? The work of Whitt et al. [26,84] provides some initial answers, but we feel that more research is needed to understand these questions completely.

Many researchers are working on future packet-switching gigabit-speed networks. These networks might provide performance guarantees. Can indices of dispersion characterize the traffic sources that can be used by algorithms that guarantee the performance of communication channels in terms of delay, jitter, and/or throughput [27]?

We discussed the difficulties associated with removing nonstationary components from packet arrival processes. Can a nonstationary version of the generalized semi-Markov process be derived that models nonstationary packet arrival processes?

How general are the models of variability that we have developed? Can they be adapted to handle, say, compressed audio and compressed video data?

Cluster models are important in applications since clusters can be treated as units by the operating system for the purpose of scheduling or processing. What is the relationship between cluster models and models of variability? Can one type of models be expressed in terms of the other?

We have expressed file server queues, composed of the superposition of several client workstation queues, in terms of the same models. Can the generalized semi-Markov model be used to represent the entire aggregate traffic?

III. Epilogue

The analysis of a massive amount of data such as that described in this work has required time and resources out of the ordinary. Real data is almost never linear and immediately suitable for simple models. We found no easy recipes for giving structure to what at first (and for quite some time) looked like an orderless aggregate of arrival times. Trial and error, experimentation, and exploratory-data-analysis techniques have slowly shaped our final solution. Unfortunately, modeling is still an art, one in which the form of the final product depends as much on the raw material as it depends on the inclination of the artist. But as artistic invention relies on techniques and tools, so our modeling of packet arrival processes has benefited from the framework and the methodology of point processes. We hope that the analyses and the models that we have developed can be the base upon which others can build.

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)

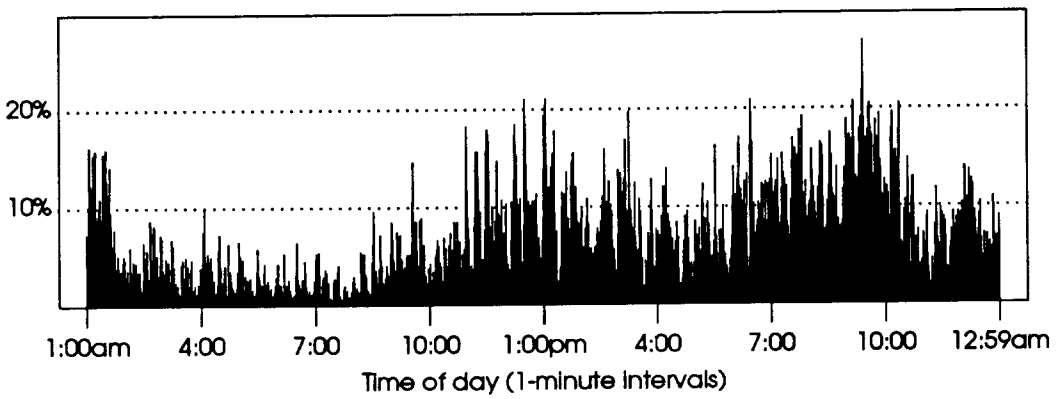
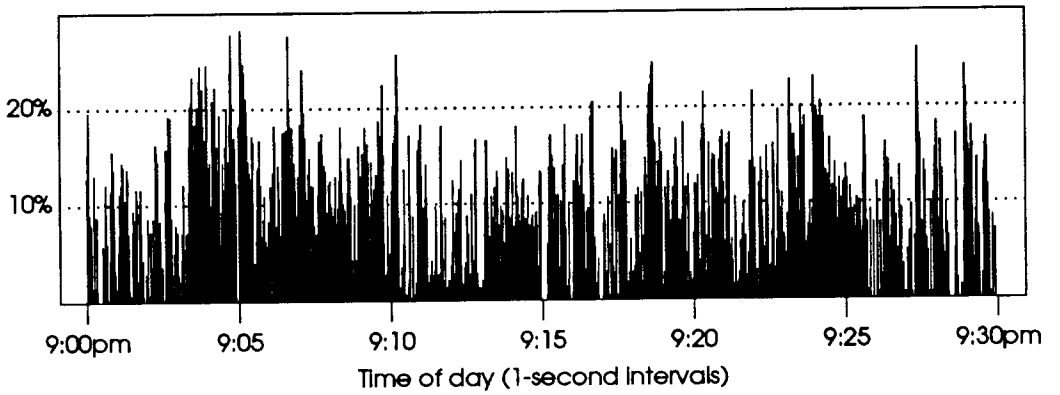


Figure A.1 shows that the utilization, measured over 1-minute intervals, rarely exceeded 20 percent. If, however, we examine the utilization more closely over smaller intervals, we see that in fact the Ethernet behavior was quite bursty, frequently reaching levels over 30 percent. Such traffic would 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)

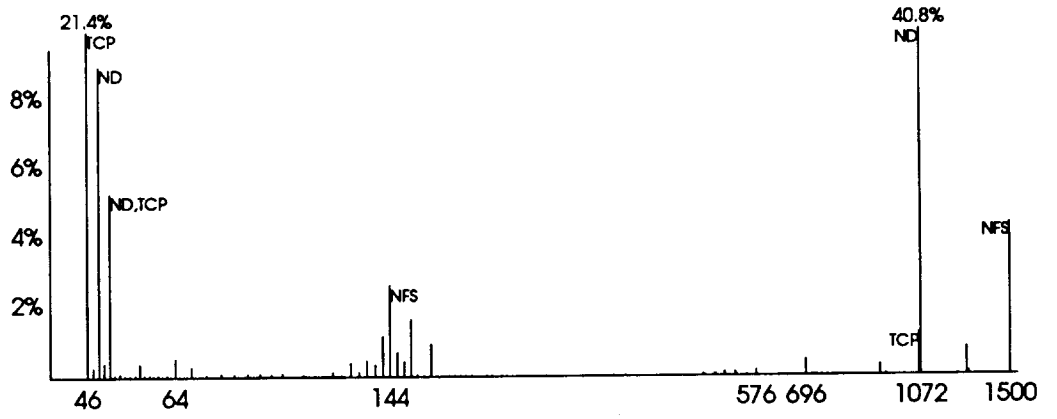


a real-world environment compels us to re-evaluate the notion that under normal traffic an Ethernet is lightly loaded [41, 80].

B. Packet Length

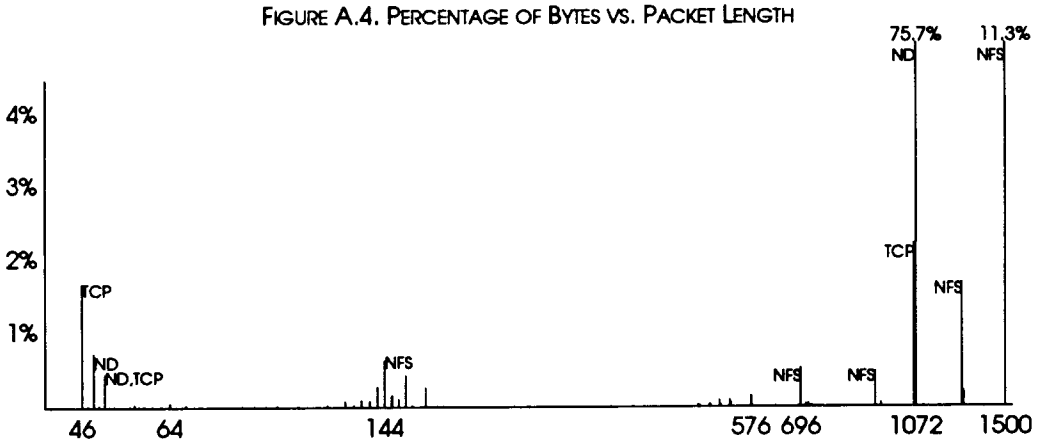
Figure A.3 and Figure A.4 display respectively the number of packets and number of bytes transmitted as a function of the packet data field length.

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



Length of packet data field

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.

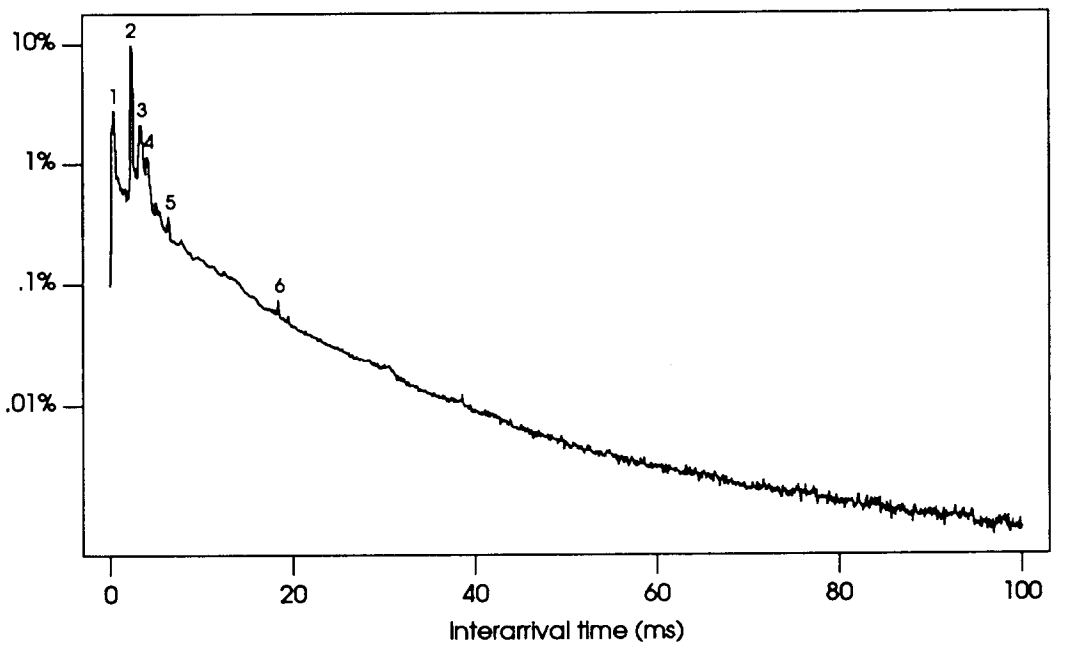
TABLE A.2. SOURCES AND DESTINATIONS OF TCP TRAFFIC (PERCENTAGES)

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

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)



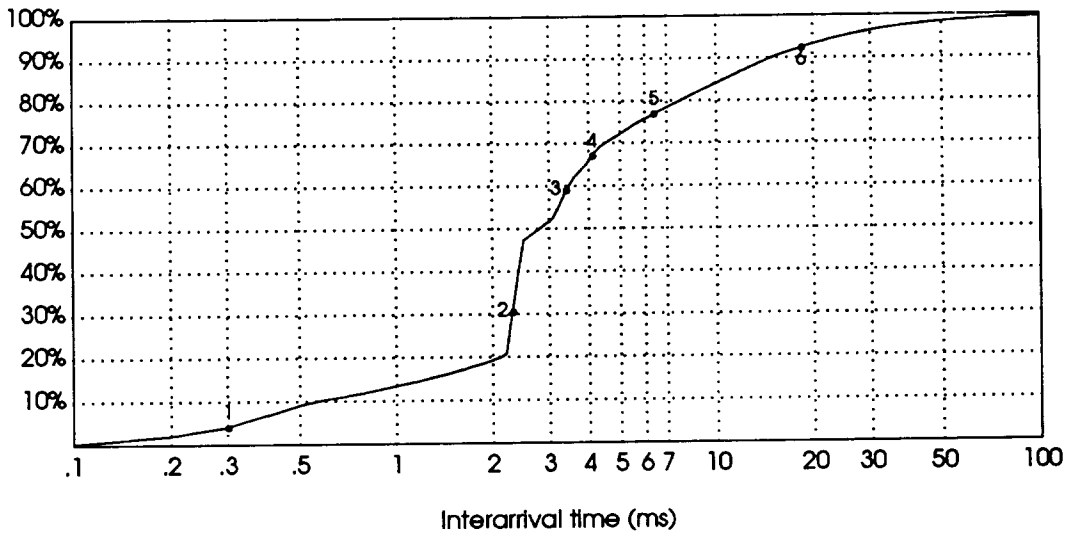
The Poisson model, very often used in analytical and simulation studies, is inappropriate for modeling our data because it assumes that there is no correlation between arrivals. One of our investigations shows that the arrival process is highly bursty and that the independence assumption is not justified [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.

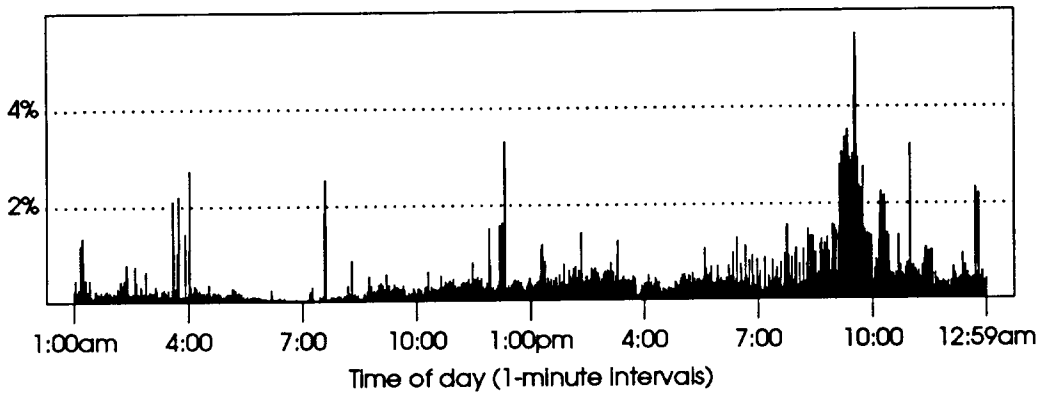
Analyzing the individual protocols will give us a more precise picture than the one obtained so far by looking at the entire packet stream. In the following three sections we shall describe the network protocols by focusing on TCP, ND, and NFS. We shall limit the analysis to those features of the protocols 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

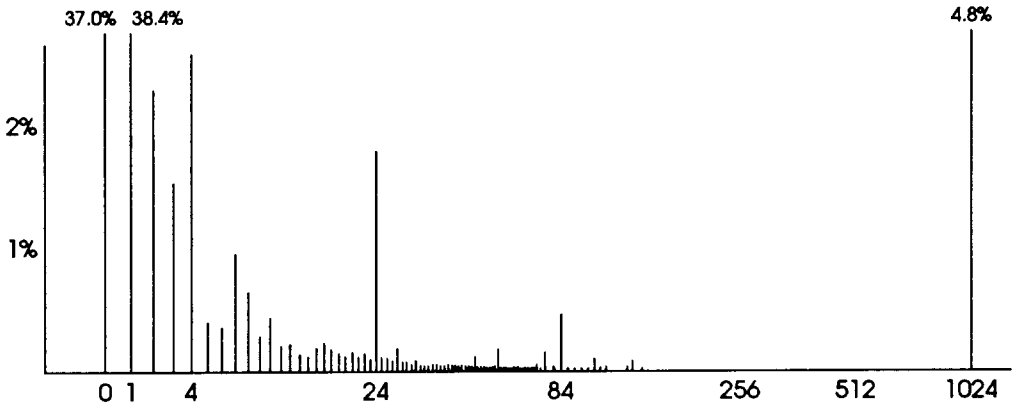
Many TCP packets carry character traffic (characters of UNIX shell command lines) from diskless workstations on the XCS network to VAXs on other networks. Therefore, many packets are short. This is displayed in Figure A.4, which also shows that TCP traffic is essentially bimodal. Figure A.8 shows the 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.

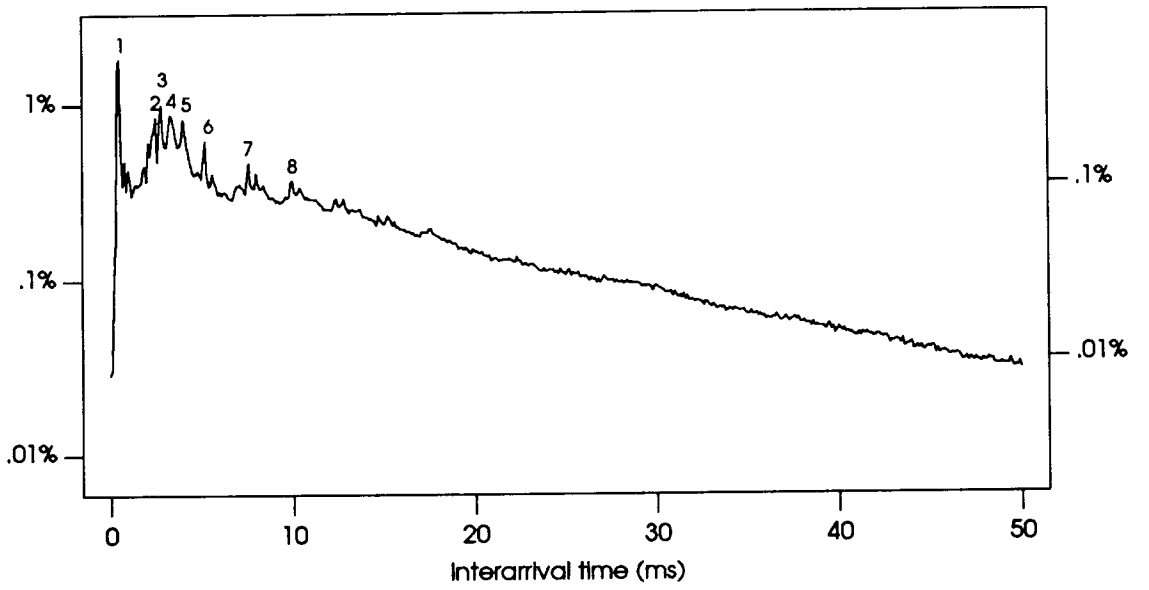
D. Source-Destination Patterns

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.)

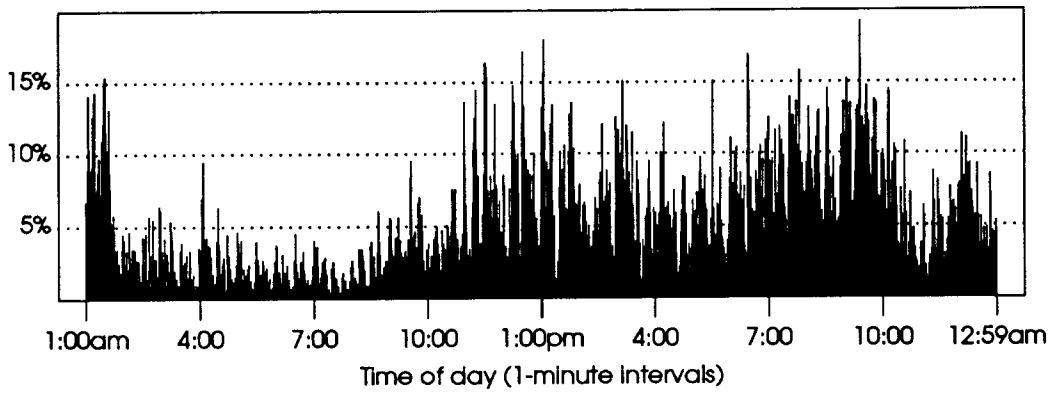
A. Utilization

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].

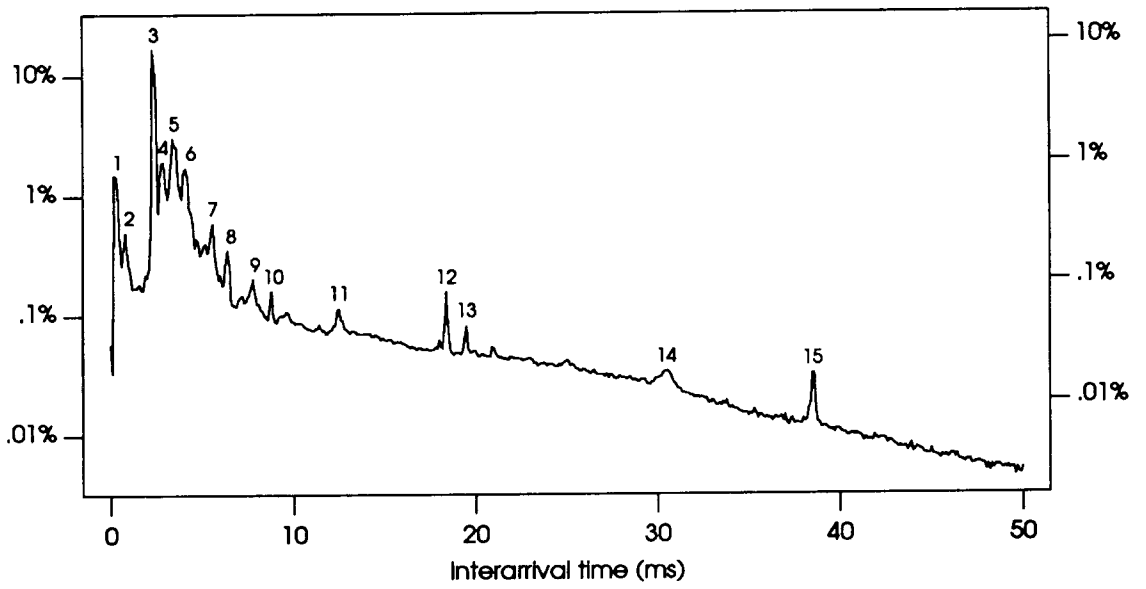
One might find it surprising to discover that the ND protocol accounts for most of the network traffic. The relative importance of the ND protocol components can be explained by 1) small physical memories on most client workstations, which cause excessive paging activity; 2) the tendency of the virtual-memory 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)



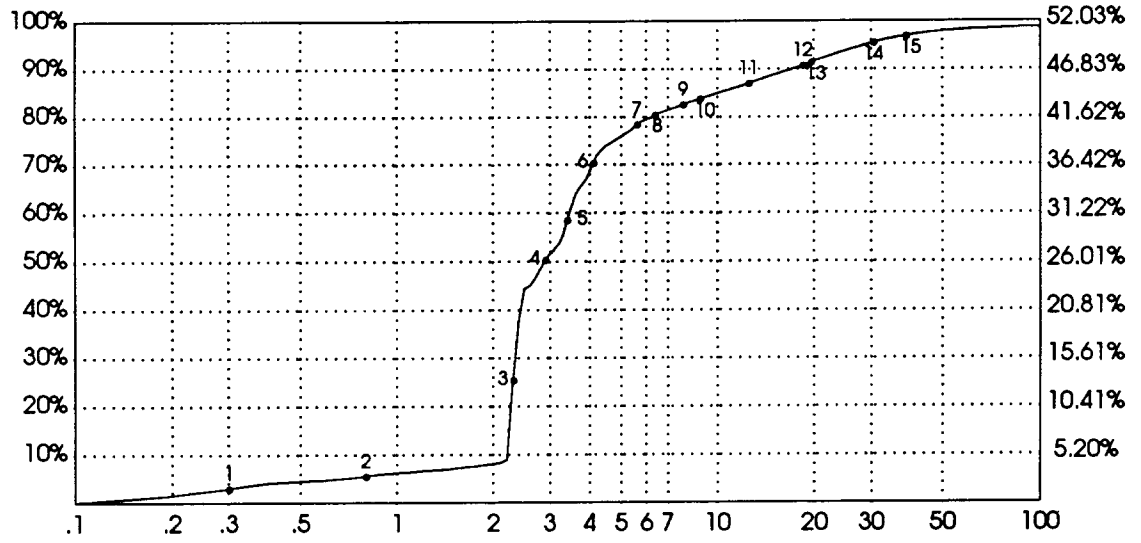
The ND protocol is very fast: 90 percent of all ND packets are followed by the next packet within 18 ms. Three factors contribute to this remarkable speed. First, the protocol is extremely simple, since it uses the IP transport mechanisms to bypass much of the kernel's code for higher-level protocols; 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)

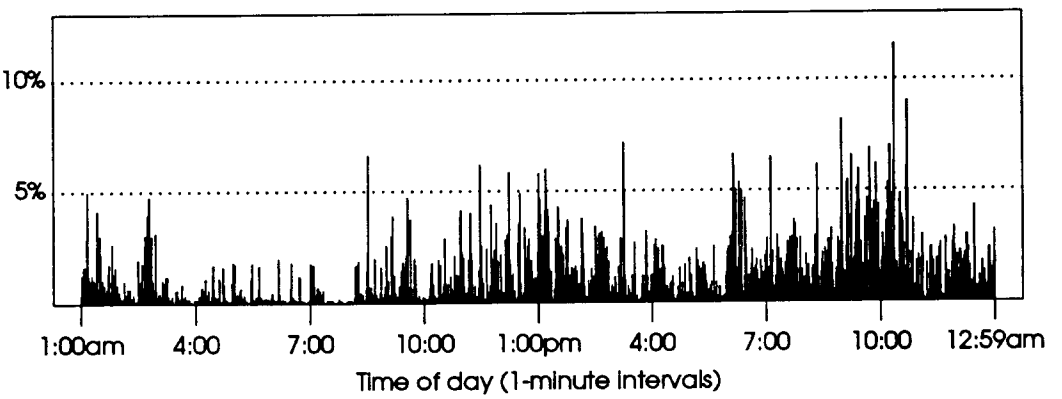
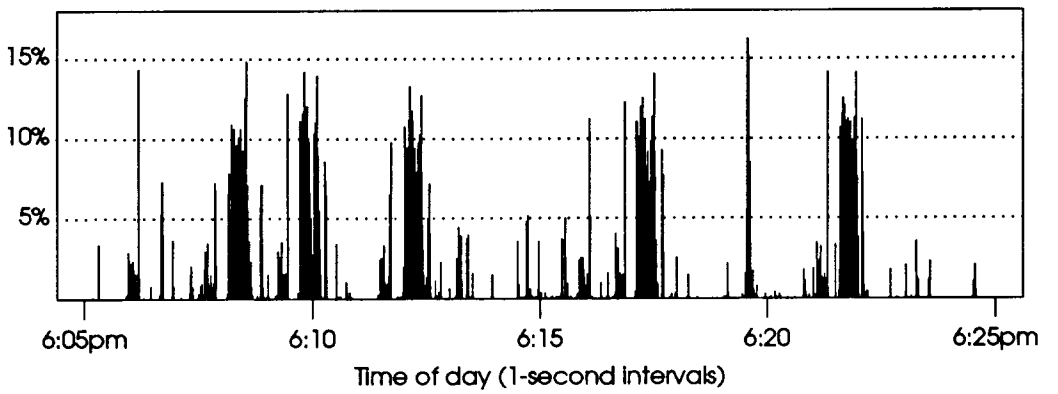


Figure A.14 displays the network utilization, over 1-second intervals, generated by NFS traffic between two Sun-3 workstations: a client and a heavily used server. The network activity is intermittent, following the usual pauses in user behavior. In addition to occasional 1-second peaks at 15 percent, 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.

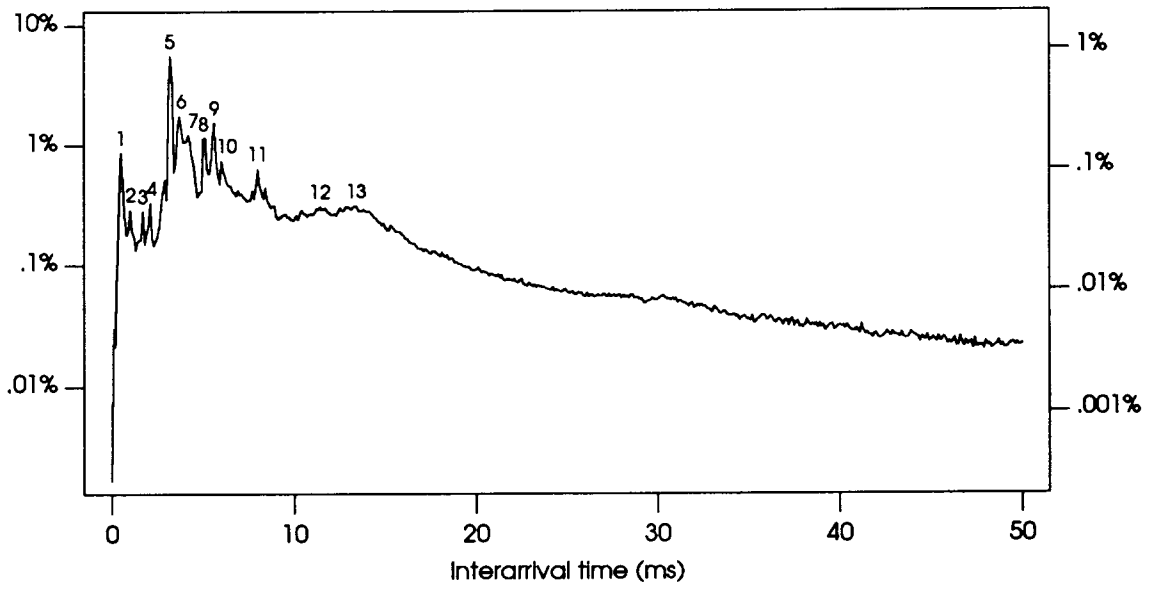
More than six percent of the write calls and more than one percent of the read calls were retransmitted. Most of these retransmissions occurred 0.7 seconds after the original transmission. The proportion of write retransmissions is higher than the proportion of read retransmissions because reads are 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

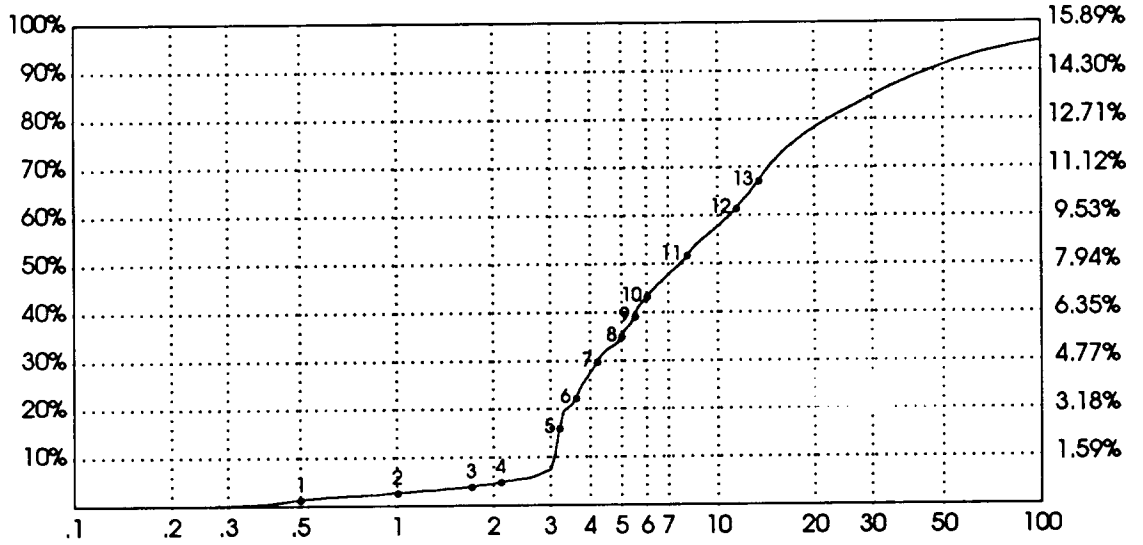
Figure A.15 displays the NFS packet interarrival-time distribution, and Figure A.16 the cumulative interarrival-time distribution. In the two figures, the interarrival time is the difference between the beginnings of the transmissions of two subsequent NFS packets. In both figures, the vertical axis 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.

References

1. Albin, S. L., Approximating Queues with Superposition Arrival Processes, *Doctoral dissertation*, Columbia University, 1981.

2. Aldous, D., Ultimate Instability of Exponential Back-Off Protocol for Acknowledgment-Based Transmission Control of Random Access Communication Channels, *IEEE Trans. on Information Theory 33*, 2 (March 1987), 219-223.

3. Anderson, D. P., R. Govindan, G. Homsy and R. Wahbe, Integrated Digital Continuous Media: a Framework Based on Mach, X11, and TCP/IP, *International Computer Science Institute*, Feb. 1990.

4. Babaoglu, O. and W. N. Joy, Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Referenced Bits, *Proc. of the 8th Symposium on Operating System Principles*, 1981, 78-86.

5. Babaoglu, O., *Virtual Storage Management in the Absence of Reference Bits*, Ph.D. dissertation, Computer Science Division, University of California, Berkeley, Nov. 1981.

6. Barlett, M. S., The Spectral Analysis of Point Processes, *J. R. statist. Soc. B 25* (1963), 264-296.

7. Barnett, L. and M. K. Molloy, ILMON: A UNIX Network Monitoring Facility, *Usenix Conference Proceedings*, Washington, DC, Jan. 1987, 133-144.

8. Bartlett, M. S., *An Introduction to Stochastic Processes (2nd edition)*, Cambridge University Press, 1966.

9. Boggs, D. R., J. C. Mogul and C. A. Kent, Measured Capacity of an Ethernet: Myths and Reality, *Proceedings of SIGCOMM 18*, 4 (August 1988), 222-234.

10. Box, G. E. P. and G. M. Jenkins, *Time Series Analysis, Forecasting, and Control*, Holden-Day, San Francisco, 1976.

11. Brillinger, D. D., Comparative Aspects of the Study of Ordinary Time Series and of Point Processes, in Developments in Statistics, P.R. Krishnaiah, Ed., *Academic Press, vol. 1*, New York, 1978, 33-133.

12. Caceres, R., Measurements of Wide-Area Internet Traffic, Report UCB/CSD 89/550, Computer Science Division, University of California, Berkeley, December 1989.

13. Charnock, D. M., Continuous Time Spectral Analysis of Intervals between Events in Stationary Point Processes, *Journal of the Royal Statist. Soc. B*, 39 (1977), 48-55.

14. Cheriton, D. R., VMTP: a Transport Protocol for the Next Generation of Communication Systems, *Proc. of SIGCOMM '86*, Aug. 1986, 406-415.

15. Cheriton, D. R. and C. L. Williamson, Network Measurement of the VMTP Request-Response Protocol in the V Distributed System, *Proceedings of ACM Sigmetrics*, 1987, 216-225.

16. Cheriton, D. R., The V Distributed System, *Communications of the ACM 31*, 3 (Mar. 1988), 314-333.

17. Cheriton, D. R. and T. Mann, Decentralizing a Global Naming Service for Improved Performance and Fault Tolerance, *ACM TOCS 7*, 2 (May 1989), 147-183.

18. Cox, D. R. and H. D. Miller, *The Theory of Stochastic Processes*, Methuen, London, 1965.

19. Cox, D. R. and P. A. W. Lewis, *The Statistical Analysis of Series of Events*, Chapman and Hall, London, 1966.

20. Cox, D. R. and V. Isham, *Point Processes*, Chapman and Hall, New York, 1980.

21. Crane, R. C. and E. A. Taft, Practical Considerations in Ethernet Local Network Design, *Proc. of the 13th Hawaii Intl. Conf. on Systems Science*, Jan. 1980, 166-174.

22. Cruz, R. L., A Calculus for Network Delay and a Note on Topologies of Interconnection Networks, Ph.D. Dissertation, Report no. UILU-ENG-87-2246, University of Illinois, July 1987.

23. Digital Equipment Corporation, Intel Corporation, and Xerox Corporation, *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specifications*, Version 2.0, July 15, 1982.

24. Feldmeier, D. C., *Empirical Analysis of a Token Ring Network*, Tech. Memo. MIT/LCS/TM-254, M.I.T. Laboratory Computing Science, Jan. 1984.

25. Fendick, K. W., V. R. Saksena and W. Whitt, *Dependence in Packet Queues, I: A Multi-Class Batch-Poisson Model*, unpublished paper, AT&T Bell Laboratories, 1987.

26. Fendick, K. W. and W. Whitt, Measurements and Approximations to Describe the Offered Traffic and Predict the Average Workload in a Single-Server Queue, *Proceedings of the IEEE 77*, 1 (January 1989), 171-194.

27. Ferrari, D. and D. C. Verma, A Scheme for Real-Time Channel Establishment in Wide-Area Networks, Report Technical Report-89-036, ICSI, May 1989.

28. Fisher, W. and K. S. Meier-Hellstern, *The MMPP Cookbook*, INRS-Telecommunications, Verdun, Quebec, and AT&T Bell Laboratories, Holmdel, NJ, in preparation, 1990.

29. Fontana, B. and A. Guerrero, Packet Traffic Characterization, Arrival Laws, and Waiting Times, *12th Int. Teletraffic Congress*, Turin, June 1988.

30. Gonsalves, T. A., Performance Characteristics of two Ethernets: an Experimental Study, *Proceedings of the ACM SIGCOMM Conference*, May 1985, 78-86.

31. Goodman, J., A. G. Greenberg, N. Madras and P. March, Stability of Binary Exponential Backoff, *Journal of the ACM 35*, 3 (July 1988), 579-602.

32. Granger, C. W. J., The Typical Shape of an Econometric Variable, *Econometrica 34* (1966), 150-161.

33. Gusella, R., A Measurement Study of Diskless Workstation Traffic on an Ethernet, *IEEE Transactions on Communications 38*, 9 (September 1990), 1557-1568.

34. Gusella, R., Characterizing the Variability of Arrival Processes with Indices of Dispersion, *IEEE Journal on Selected Areas in Communications*, (to appear) February 1991.

35. Heffes, H. and D. M. Lucantoni, A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance, *IEEE Journal on Selected Areas in Communications SAC-4*, 6 (September 1986), 856-868.

36. Heimlich, S. A., Traffic Characterization of the NFSNET National Backbone, *Proceedings of the 1990 Usenix Conference*, January 1990.

37. Intel, 82586 Local Area Network Coprocessor, Technical Information Sheets, December 1986.

38. First International Workshop on Network and Operating System Support for Digital Audio and Video, ICSI Report Technical Report-90-062, International Computer Science Institute, Berkeley, California, November 8-9, 1990.

39. Jacobson, V., private communication, July 1988.

40. Jacobson, V., Some Interim Notes on the BDS Network Speedups, message posted on the 'comp.protocols.tcp-ip' news group, July 20, 1988.

41. Jain, R. and W. R. Hawe, Performance Analysis and Modelling of Digital's Networking Architecture, *Digital Technical Journal*, Sept. 1986, 25-34.

42. Jain, R. and S. A. Routhier, Packet Trains – Measurements and a New Model for Computer Network Traffic, *IEEE Journal on Selected Areas in Communications SAC-4*, 6 (September 1986), 986-995.

43. Karels, M. J., *Another Internet Subnet Addressing Scheme*, University of California, Berkeley, Feb. 1985.

44. Karels, M. J., private communication, Feb. 1987.

45. Kendall, M. G. and A. Stuart, *The Advanced Theory of Statistics*, Griffin, London, 1983.

46. Kleinrock, L., *Queueing Systems, vol. 1: Theory*, Wiley and Sons, 1975.

47. Kleinrock, L., Computer-Communication Networks: Measurements, Flow Control, and ARPANET Traps, in *Queueing Systems*, vol. 2: Computer Applications , Wiley and Sons, 1976.

48. Lam, S. S., A Carrier Sense Multiple Access Protocol for Local Networks, *Computer Networks 4*, 1 (February 1980), 21-32.

49. Lazowska, E. D., J. Zahorjan, D. R. Cheriton and W. Zwaenepoel, File Access Performance of Diskless Workstations, *ACM Transactions on Computer Systems 4*, 3 (Aug. 1986), 238-270.

50. Leach, P. J., P. H. Levine, B. P. Douros, J. A. Hamilton, D. L. Nelson and B. L. Stumpf, The Architecture of an Integrated Local Network, *IEEE Journal on Selected Areas in Communications 1*, 5 (Nov. 1983), 842-857.

51. Leland, W. E., LAN Traffic Behavior from Milliseconds to Days, *Broadband Technologies: Architectures, Applications, Control and Performance*, Morristown, New Jersey, October 1990.

52. Lewis, P. A. W., A Branching Poisson Process Model for the Analysis of Computer Failure Patterns, *J. R. statist. Soc. B 26* (1964), 398-456.

53. Lyon, B., Sun External Data Representation Specification, Technical Report, Sun Microsystems, Inc., 1984.

54. Marshall, W. T. and S. P. Morgan, Statistics of Mixed Data Traffic on a Local Area Network, *Teletraffic Issues in an Advanced Information Society*, Kyoto, Japan, 1985,

569-575.

55. McKusick, M. K., W. N. Joy, S. J. Leffler and S. J. Fabry, A Fast File System for UNIX, *ACM Transactions on Computer Systems 2*, 3 (Aug. 1984), 181-197.

56. Meier, K. S., A Statistical Procedure for Fitting Markov-Modulated Poisson Processes, Ph.D. dissertation, University of Delaware, 1984.

57. Metcalfe, R. M. and D. R. Boggs, Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM 19*, 7 (July 1976), 395-404.

58. Mogul, J., Internet Subnets, Internet Network Working Group, RFC 917, Stanford University, Oct. 1984.

59. Nelson, M., B. Welch and J. Ousterhout, Caching in the Sprite Network File System, *ACM TOCS 6*, 1 (Feb. 1988), 134-154.

60. Neuts, M. F., *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, M. Dekker Inc., New York, 1989.

61. Neuts, M. F. and 1986, The Caudal Characteristic Curve of Queues, *Adv. Appl. Prob. 18*, 221-254.

62. Nomura, M., T. Fujii and N. Ohta, Basic Characteristics of Variable Rate Video Coding in ATM Environment, *IEEE Journal on Selected Areas in Communications 7*, 5 (June 1989), 752-760.

63. Nowicki, B., NFS, Network File System Protocol, Internet Network Working Group, RFC 1094, Sun Microsystems, Inc., March 1989.

64. Ousterhout, J., H. Da Costa, D. Harrison, J. Kunze, M. Kupfer and J. Thompson, A Trace-Driven Analysis of the UNIX 4.2BSD File System, *Proceedings of 10th ACM Symposium on Operating System Principles*, Orcas Island, WA, December 1985.

65. Pawlita, P. F., Two Decades of Data Traffic Measurements: a Survey of Published Results, Experiences and Applicability, *Proceedings of the 12th International Telecommunication Conference*, Torino, June 1988.

66. Plummer, D. C., An Address Resolution Protocol, Internet Network Working Group, RFC 826, Massachussets Institute of Technology, Nov. 1982.

67. Postel, J., Simple Mail Transfer Protocol, Internet Network Working Group, RFC 821, USC Information Sciences Institute, Aug. 1982.

68. Postel J., Ed., User Datagram Protocol, Internet Network Working Group, RFC 768, USC Information Sciences Institute, Aug. 1980.

69. Postel J., Ed., Internet Protocol – DARPA Internet Program Protocol Specification, Internet Network Working Group, RFC 791, USC Information Sciences Institute, Sept. 1981.

70. Postel J., Ed., Transmission Control Protocol, Internet Network Working Group, RFC 793, USC Information Sciences Institute, Sept. 1981.

71. Presotto, D., private communication, July 1987.

72. Pyke, R., Markov Renewal Processes: Definitions and Preliminary Properties, *Ann. math Statist. 32* (1961), 1231-1242.

73. Pyke, R., Markov Renewal Processes with Finitely Many States, *Ann. math Statist. 32* (1961), 1243-1259.

74. Ramaswami, V. and G. Latouche, Modeling Packet Arrivals from Asynchronous Input Lines, *12th Int. Teletraffic Congress*, Turin, June 1988.

75. Reynolds, J. and J. Postel, Official Internet Protocols, Internet Network Working Group, RFC 1011, USC Information Sciences Institute, May 1987.

76. Rifkin, A. P., M. P. Forbes, R. L. Hamilton, M. Sabrio, S.Shah and K. Yueh, RFS Architectural Overview, *Usenix Conference Proceedings*, Atlanta, June 1986, 248-259.

77. Sandberg, R., D. Goldberg, S. Kleinman, D. Walsh and B. Lyon, Design and Implementation of the Sun Network Filesystem, *Usenix Conference Proceedings*, June 1985, 119-130.

78. Scheifler, R. W. and J. Gettys, The X Window System, *ACM Transactions on Graphics 5*, 2 (Apr. 1986), 79-109.

79. Shenker, S., Some Conjectures on the Behavior of Acknowledgement-Based Transmission Control of Random Access Communication Channels, *Proceedings of ACM Sigmetrics*, 1987, 245-255.

80. Shoch, J. F. and J. F. Hupp, Measured Performance of an Ethernet Local Network, *Communications of the ACM 23*, 12 (December 1980), 711-721.

81. Shoch, J. F., Y. K. Dalal, D. D. Redell and R. C. Crane, The Ethernet, in *Local Area Networks: An Advanced Course*, Hutchinson, D., J. Mariani and D. Shepherd (editor), Springer-Verlag, 1982, 549-572.

82. Smith, W. L., Regenerative Stochastic Processes, *Trans. Amer. math. Soc. 104* (1955), 79-100.

83. Song, C. and L. H. Landweber, Optimizing Bulk Data Transfer Performance: a Packet Train Approach, *Proceedings of ACM SIGCOMM*, 1988, 134-145.

84. Sriram, K. and W. Whitt, Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data, *IEEE Journal on Selected Areas in Communications SAC-4*, 7 (September 1986), 833-846.

85. Sun Microsystems, *ND(8)*, UNIX User's Manual, Section 8, 1983.

86. Sun Microsystems, SunOS Release 4.0 Reference Manual, 1988.

87. Sun Microsystems, RPC: Remote Procedure Call Protocol Specification, Internet Network Working Group, RFC 1050, April 1988.

88. Sventek, J., W. Greiman, M. O'Dell and A. Jansen, A Comparison of Experimental and Theoretical Performance, *Computer Networks*, August 1984, 301-309.

89. Swinehart, D., G. McDaniel and D. Boggs, WFS: A Simple Shared File System for a Distributed Environment, *Proc. of the 7th ACM Symposium on Operating System Principles*, Dec. 1979, 9-17.

90. Thacker, C. P., E. M. McCreight, B. W. Lampson and D. R. Boggs, Alto: A Personal Computer, in *Computer Structures: Principles and Examples*, Siewiorek, D. P., C. G. Bell and A. Newell (editor), McGraw-Hill Book Co., 1982, 549-572.

91. Thompson, K., UNIX Time-Sharing System: UNIX Implementation, *Bell System Technical Journal 57*, 6 (July-Aug. 1978), 1931-1946.

92. Tzou, S., Software Mechanisms for Multiprocessor TLB Consistency, Ph.D. dissertation, UCB Technical Report 89/551, University of California, Berkeley, December 1989.

93. Wilson, D. V. and W. E. Leland, High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for BISDN Networks, *Proceedings of the 1990 Bellcore Symposium on Performance Modeling*, Livingston, NJ, May 1990.

94. Young, M., A. Tevanian, R. Rashid, D. Golub, J. Eppinger, J. Chew, W. Bolosky, D. Black and R. Baron, The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System, *Proc. of the 11th ACM Symposium on Operating System Principles*, 1987, 63-76.

95. Zhou, S., H. D. Costa and A. J. Smith, A File System Tracing Package for Berkeley UNIX, *Report No. UCB/CSD 85/235*, University of California, Berkeley, May 1985.