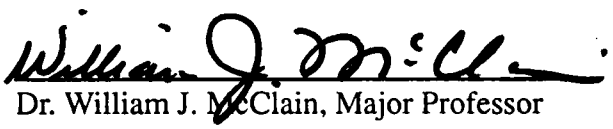
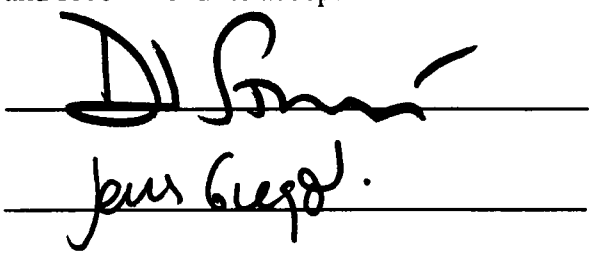
To the Graduate Council:

I am submitting herewith a thesis written by Timothy William Hickerson entitled "The Impact of the X Window System Upon Ethernet Performance." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.



Dr. William J. McClain, Major Professor

We have read this thesis
and recommend its acceptance:



Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of the Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Timothy Wm. Hickerson

Date 11/13/92

The Impact of the X Window System Upon Ethernet Performance

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Timothy William Hickerson
December 1992

Acknowledgments

I would like to express my thanks and appreciation to Dr. Bill McClain for his concern, guidance, and editorial abilities demonstrated during the preparation of this thesis. Thanks is also extended to Dr. David Straight and Dr. Jens Gregor for their comments and for serving on my committee.

Gratitude is offered to Martin Marietta Energy Systems for permission to utilize their equipment during development of this thesis, and for their general support of continued education. Specific thanks is owed to Harold Cromwell for his motivation and support.

Finally, a special appreciation and thanks to my family -- to my wife, Cindy, whose patience and support were, and continue to be priceless, and, to my parents for instilling in me the desire to strive for excellence in all aspects of life.

Abstract

The purpose of this thesis was to investigate the demands placed upon an Ethernet by remote clients of the X Window System. Three methods of investigation were applied - queuing analysis, simulations, and an after-the-fact analysis based upon utilization statistics gathered while monitoring network traffic. Each of these methods was applied to profiles which characterize the primary transactions executed by a user -- the display of textual or graphical information and the input of information either by keyboard or mouse.

Accuracy of the data obtained from simulations was confirmed by both the theoretical analysis and the experimental results taken from an actual network. Given the reliability of the output, and considering the flexibility of the simulation tool, simulations were used as the primary instrument for the evaluation.

Results from this study reveal that the display of text and the input of graphical information provide insignificant network loads, while the display of graphics and the input of text can produce appreciable loads yet maintain acceptable user response times. In contrast, applications with a high degree of user interaction (i.e., those

applications which make extensive use of drag operations) can saturate the network and provide very poor response times with relatively few users.

A couple of network configurations are given whereby remote X traffic may be localized to avoid potential network saturation.

Table of Contents

List of Tables

List of Figures

CHAPTER 1 Introduction

The field of distributed computing is a rapidly growing and immensely important technology destined to change the nature of data processing. In general terms, distributed computing is the cooperation of physically separated computers communicating through networks to solve problems. The accomplishment of distributed computing in the 1990's is largely an outgrowth of two underlying technologies: *Very Large Scale Integration (VLSI)* and *computer networks*.

Before 1970, a generally accepted rule was Grosch's law which stipulated that "the cost per machine instruction was inversely proportional to the square of the size of the machine" [Mart81]. This line of reasoning nourished the computer facility centralization of the period.

VLSI technology began to invalidate Grosch's law in the 1970's with the development of minicomputers and microcomputers. Researchers realized that powerful and inexpensive computers would soon become available and that, through networking, these computers could cooperatively solve tasks [Witt91]. By the 1980's Reduced Instruction Set Computing (RISC) technology had completely reversed Grosch's Law.

Concurrent with these VLSI technology developments were major advances in the field of computer networks. The national research network ARPAnet, created in 1969 by the US Department of Defense, grew from four sites in 1969 to an Internet spanning 26 countries, 5,000 networks, and more than 300,000 computers in 1991 [Cerf91].

The need for sharing expensive printers led Xerox Corporation researchers to develop and build Ethernet [Metc76] in 1975; IBM began selling token-ring interfaces in 1985. By 1989, the cost of an Ethernet interface chip had fallen to $50, and variations of Ethernet and token-ring had become international Local Area Network (LAN) standards as defined in IEEE 802.3 and 802.5 [Witt91].

These technological developments, their falling costs, and the desire and need to share physically-distributed resources have brought distributed computing to the technological forefront in the 1990's.

The distributed computing model offers:

- users a means for marrying the powerful and costly mainframes with the cheaper desktop microcomputers,
- the possibility of accessing a variety of applications and/or data on different host computers from one device,
- higher utilization of otherwise wasted computing power,
- and possible financial savings through the elimination of duplicate resources otherwise required by physically separated facilities.

Whereas the potential of distributed computing is great, so is the dilemma facing systems engineers, analysts, and managers who make capacity planning decisions. The decisions regarding implementations and their effect upon system and corporate resources are complex and can have long-lasting financial consequences. One such resource that may be adversely affected by the distributed computing model is an intrinsic one, the computer network.

This likelihood is inherent to the basic characteristics of distributed computing. Two common characteristics of distributed computing, generally agreed upon by users, analysts, and vendors, are [Parr91] [Zamb91]:

- the use of the client/server model of computing, and
- the concept of transparency.

In the client/server model of computing, the work is split between clients and servers. Clients are the stations which ask for a work or service to be done while servers are those stations which perform the work or service requested by the clients. The client and server processes are independent, may reside on the same or different hosts, and communicate via a message protocol. The client and server hosts may differ in either the type or brand of device.

The second characteristic, transparency, provides the same view of the system resources regardless of the machine that one is using. This viewpoint may be that of the human user, or, the procedural interface to the message protocol, database queries, files, etc.

The increased message communications (overhead) resulting from this separation of the client and server, coupled with the overhead in maintaining transparency have created demands for higher bandwidth LANs such as fiber distributed data interface (FDDI) [Vere90] and

have raised concerns regarding the longevity of current LAN technologies [Metc91].

One system in common use today which possesses the characteristics of transparency and the client/server model is the X Window System. Concerns regarding the impact on a network have permeated the X Window System user community with conflicting appraisals. Some persons claim little or no network impact [Coul91], and others claim significant impacts [Pado91] [Zach91a] [Zach91b]. Little documentation or recommendations appear in current literature [Thar91] to assist system and corporate designers in their capacity planning endeavors.

Yet the popularity and proliferation of the X Window System as a network-based windowing architecture is growing at an exponential rate [Vere91]. Hence, an examination to determine the effects of the X Window System upon network (specifically Ethernet[1]) performance is warranted.

Description and Objectives of the Research

The principal aim of this investigation was to examine the impact of the X Window System upon Ethernet performance as a function of

1. Unless otherwise stipulated, henceforth in this work the term "network" is used in reference to the specific network under consideration -- an IEEE 802.3 10Base5 Ethernet.

the number and types of users (applications), and to provide guidelines and recommendations for system designers through the presentation of the findings. In accomplishing this aim, the following questions were considered:

- What roles do the number and types of users play?
- How many users of a given type can the network support?
- Are the response times within acceptable limits?
- How does varying the levels of background network utilization affect the results?

The procedure for this investigation was to look initially into the X Window System, to relate its operation to network activity, and to characterize and classify users and applications based upon their contributions to the network traffic.

A model was developed for each user or application class. Each class was examined analytically and by simulations while varying the number of users and the level of background network traffic.

Analytic investigations were performed by the means of queueing models, while a commercial LAN performance analysis and simulation tool (LANNET II.5 by CACI Products Company) was used as the engine for the simulations. Experimental data collected by

recording and playing back user sessions were compared with the analytic and simulation results for completeness.

Synopsis of the Remaining Chapters

Chapter 2 begins by briefly describing Ethernet, then moves to a description of the X Window System, its major components, and its interaction with the underlying network. Discussion is also given to the philosophy applied in selecting the models used in the performance evaluation.

Descriptions of the methods used in the impact evaluations are contained in Chapter 3. These methods include analytic, simulation, and experimental approaches.

Chapter 4 implements the methods outlined in Chapter 3 and presents the results.

A summary of the overall thesis and some remarks by the author are given in Chapter 5.

CHAPTER 2

Formation of the System Models

The certainty of any model-based analysis can be no better than the exactness of the model that portrays the real world system. Thus the purpose of this chapter is to explore and extract facts regarding network performance and the X Window System and to incorporate this knowledge into models for investigating network behavior. Specifically, this chapter discusses:

- the Ethernet local area network,
- the X Window System,
- the correlation between the X Window System and network activity, and
- the characterization and development of the system models.

This chapter concludes with the specifications of the models used in evaluating the impact of the X Window System upon Ethernet performance.

Ethernet

Ethernet is a packet-switching, baseband bus system for local data communications among computing stations. Ethernet [Metc76] originated from the radio-based, packet collision and retransmission concepts of the ALOHA network developed at the University of Hawaii.

The shared communication facility of Ethernet is the branching ether, which, in most cases, is a passive coaxial cable with no central control. This cable is bidirectional and properly terminated so that no reflections occur at its ends. A station connects bit-serially to a transceiver, which taps into the passive ether. A packet is broadcast onto the ether, is heard by all stations, and is copied from the ether by all destination stations whose addresses match the destination address of the packet.

Ethernet employs a random-access procedure referred to as carrier-sense multiple access with collision detection (CSMA/CD). When a station wishes to transmit a packet, a carrier sense mechanism first permits the station to "listen" to the ether; the station defers

transmission if a transmission is in progress. (Listening to the ether is made possible by the phased-encoding of a packet, thus assuring that there is at least one transition on the carrier during each bit-time.) If no station is transmitting, the sender begins transmitting immediately; otherwise, the sender waits until the packet already on the ether has passed before transmitting.

A collision occurs when two or more stations sense that the ether is idle and begin transmitting simultaneously. A station continues to listen to the ether while transmitting and detects a collision when the signal on the ether does not match its own output. If a collision is detected, transmission of the packet is terminated, and a jamming signal (32 bits of arbitrary data) is transmitted to ensure that all other stations are aware of its occurrence.

After a station has detected a collision and aborted, the station waits a random retransmission delay, then begins the access/transmission cycle over again. Retransmission delays are computed using a binary exponential backoff algorithm, with an intent to equitably resolve contention among the connected stations. The algorithm uses a random number generator to determine how long to wait before attempting the retransmission. The mean wait is initially equal to the end-to-end round-trip delay on the cable. If the retransmission

attempt also results in a collision, the mean delay time is doubled, and continues doubling with every failed attempt at retransmission.

An Ethernet frame, as depicted in Figure 1, consists of seven fields: the preamble, the start frame delimiter, the destination address, the source address, the type field, the data field, and the frame check sequence.

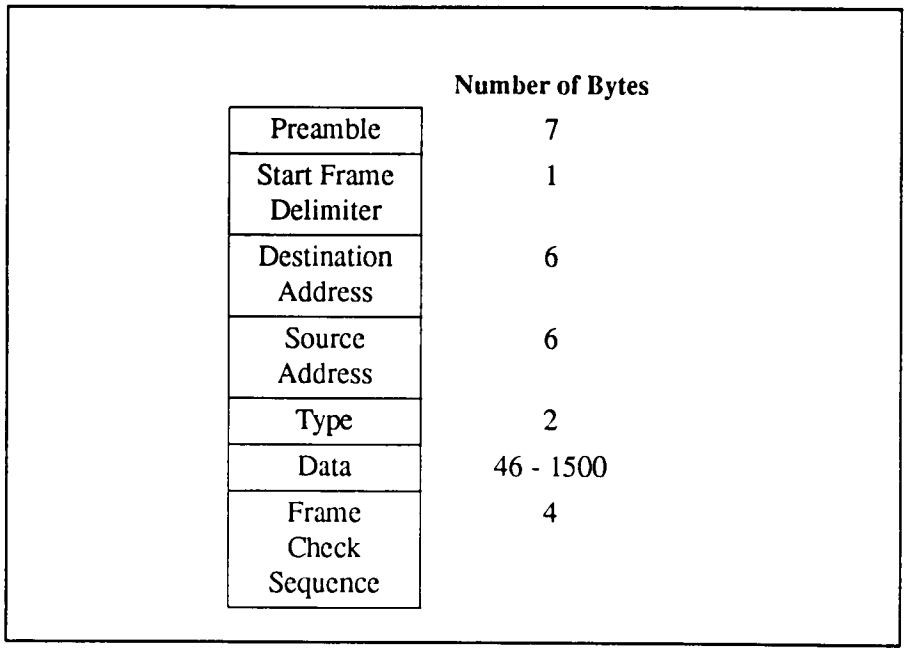


FIGURE 1 Ethernet Frame Format

The preamble of an Ethernet frame is a seven-byte field that is used to allow receivers to reach a steady-state synchronization with the frame

time [IEEE85]. The seven-byte field is a series of 1's and 0's, i.e. 10101010 ...

The start frame delimiter field is a one-byte sequence of 10101011. It immediately follows the preamble and indicates the start of a frame.

The first bit of the destination address is used to identify the type of address - either an individual or group address. If this bit is a 0, then the field contains the unique address of the destination. If the bit is a 1, the field specifies a group or multi-destination address; a special case is the broadcast address which is all 1's. In the source address, the first bit is a reserved bit and is always 0.

The type field is a 16-bit field that defines how the data field is to be interpreted by the receiver. The data field ranges from 46 to 1500 bytes. A minimum size data field is required to distinguish valid data packets from collision fragments. The frame-check sequence is a 32-bit cyclic redundancy check (CRC) generated as a function of contents of the preamble, start frame delimiter, source address, destination address, and data fields.

A 10Base5 LAN denotes a 10 Mb/second baseband specification whose maximum segment length is 500 meters [IEEE85]. Multiple segments may be interconnected by repeaters with an upper limit of five segments; however, three interconnected segments is considered

a practical limit [Chor89]. The maximum number of transceivers on any segment is 100, with a minimal spacing of 2.5 meters between any two transceivers.

Overview of the X Window System

The X Window System, commonly referred to simply as X, is a device-independent, network-based software system which allows programs to display windows containing text and graphics on any hardware architecture which supports the X protocol [Youn90].

X was developed at Massachusetts Institute of Technology (MIT) in 1984 under the oversight of Project Athena, which was sponsored by Digital Equipment Corporation (DEC) and International Business Machines (IBM). The initial implementation resulted from the porting of the W prototype window system developed at Stanford University, and hence the name X followed. Early versions of X saw primary use within DEC and MIT. In 1985, Version 9 became publicly available, and, with the release of Version 10 in 1986, other manufacturers, such as Hewlett-Packard, were contributing and developing products based upon X. Version 11 of X saw the development and the passing of control and support to a consortium of hardware and software vendors -- the X Consortium. X Version 11

Release 5 is the product currently available from the X Consortium [Sche90].

X provides a hierarchy of resizable rectangular windows and facilities for creating text and graphics in these windows. Windows can be moved, resized, and restacked dynamically. Windows can also overlap one another. Applications may contain thousands of these resource-cheap windows.

In X, the roles of client and server seem opposite from their use in other computing environments. To X, the client is the application program that the user runs, while the server is the local system software that manages a single display, keyboard, and mouse. The server acts as the mediator between the user and the application program. A client application program sends drawing and information requests to the server; the server responds to these requests and also sends back user input and error reports. The client may be running on the same machine as the server or on a different computer in the network [ORei90]. Multiple clients can have simultaneous open connections to a single server, and, similarly, a single client can have open connections to multiple servers simultaneously [Sche87]. Figure 2 shows a server, three clients, and their relationship to the network.

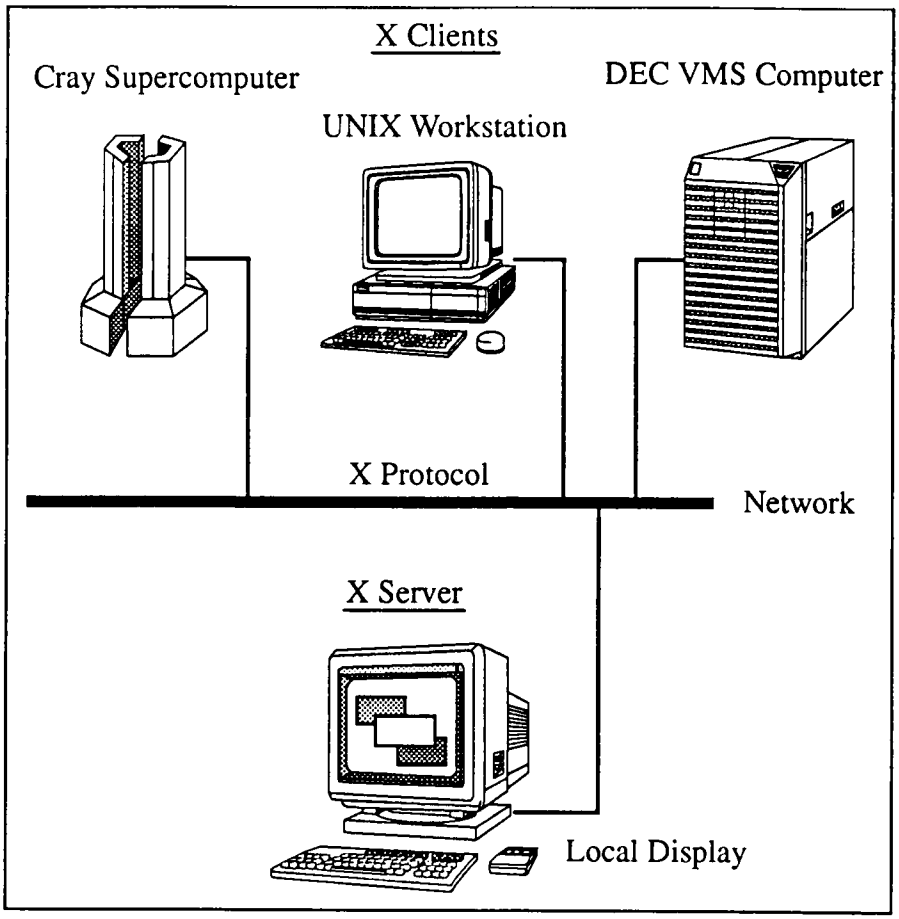


FIGURE 2 X Window System Client-Server Model

Clients and servers use the X protocol to exchange information. When the client and server reside on the same machine, communication is often implemented using shared memory or local interprocess communications channels. When the client and server reside on different machines, any network transport layer which provides reliable, bidirectional, and ordered delivery of the data may be used. Two commonly used transport layers are TCP/IP and

DECnet. The X protocol is designed to operate primarily in an asynchronous fashion eliminating the need for round-trip communication, thereby improving performance [Sche90].

The X protocol specifies four formats of messages that may be transferred between the client and server: *requests*, *replies*, *events*, and *errors*. Requests are sent from the client to the server while replies, events, and errors are sent from the server to the client [ORei90] [Sche87] [Sche90] [Youn90]. All X protocol messages are multiples of four bytes in length to allow for easy alignment on 16 and 32 bit boundaries. The description and encoding for the requests, replies, events, and error messages referenced in this work are given in APPENDIX A.

When a client application needs a service provided by the server, a protocol *request* is sent to the server. An X protocol request can be any multiple of four bytes in length. Every request contains a four-byte header consisting of one byte for the operation code, two bytes for the length of the request, and a single data byte. Zero or more additional data bytes may follow the request header. Every request on a given connection is implicitly given a sequence number, starting with one, that is used in correspondence back to the client, i.e. replies, errors, and events. The protocol request can carry a broad assortment

of information, such as a specification for drawing a line, text, or an inquiry about the current window position and size.

A protocol *reply* is the response sent from the server to the client as a result of a previous protocol request. Replies can be any multiple of four bytes in length, with a minimum length of 32 bytes. Every reply contains the sequence number of the corresponding request. Not all requests cause a corresponding reply to be generated.

The principal method of communication from the server to the client is the protocol *event.* Protocol events are 32 bytes in length. Events are generated by the server as a result of some user action or to notify the client of changes in the state of a window or a resource. Every event also contains the sequence number of the last request issued by the client that has been or is currently being processed.

Protocol *errors* are also 32 bytes in length. Every error includes an error code, an operation code for the failed request, and the corresponding sequence number of the failed request.

Correlation between the X Window System and Network Activity

As previously discussed, only remote clients generate network traffic. One might think that remote client processing is a rare circumstance,

but experience by others [Sche87] indicates that users in a workstation environment routinely make use of the remote capabilities of X. Furthermore, the advent of the X terminal[1], where all client processing is remote, has made remote processing rather commonplace. Therefore, only remote clients communicating over an Ethernet network are given attention in this work.

Figure 3 illustrates the flow of information from the X Window System, either the client or the server, to the physical network, and the relationship of this flow to the ISO/OSI network model. The illustration, assuming a TCP/IP transport connection across an Ethernet network, also pictures the overhead encountered along the data path. As seen in the figure, a minimum of 66 bytes of overhead is required for each X protocol transaction -- 20 for TCP, 20 for IP, and 26 for Ethernet.

The potential effect of X upon network performance may be comprehended by examining the primary user transactions [Hewl89] executed in an X session while keeping in mind the preceding discussion relating to the protocol message exchange between client and server. User transactions result primarily in

- the display of data, either text or graphics, and

1. The term "X Terminal" is used in a general sense in this work to refer to either of the following: a dedicated X Window Graphics Terminal or a software package that allows a personal computer (IBM PC, Macintosh, etc.) to be used as an X server.

- the input of data, either keyboard or mouse entry.

Text is typically displayed using either the PolyText or ImageText protocol request and is illustrated in the sample X session of Figure 4. Note that in the session, these requests do not generate a reply from the server. The PolyText or ImageText protocol request can transfer approximately 1400 characters in a single Ethernet packet. When characters of the same font are to be drawn, the total overhead of the

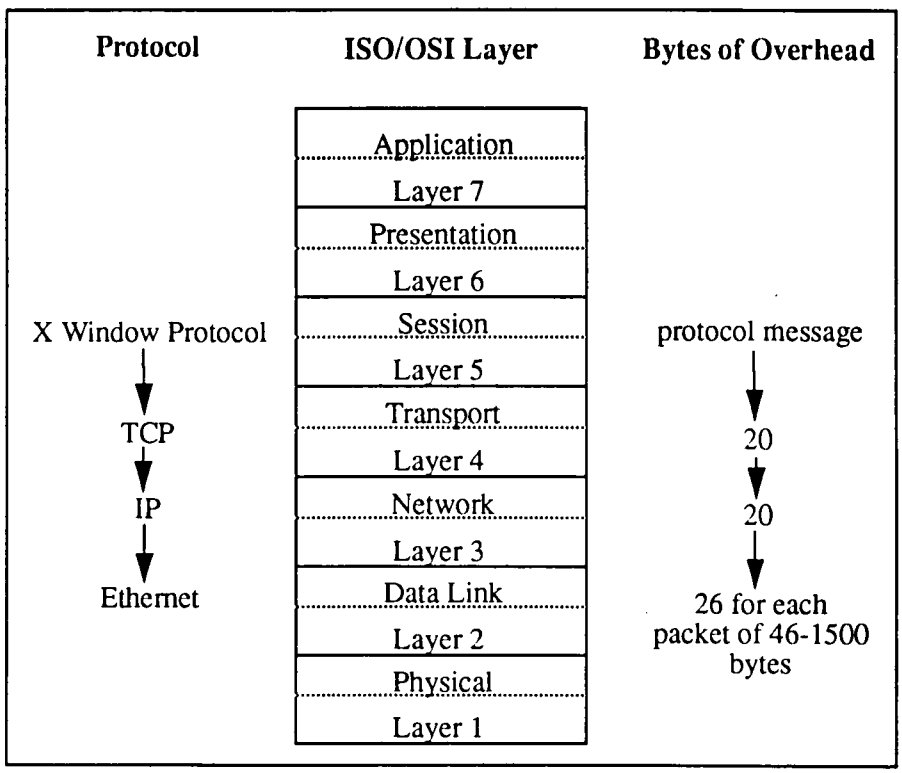


FIGURE 3 Data Flow within the ISO/OSI Network Model

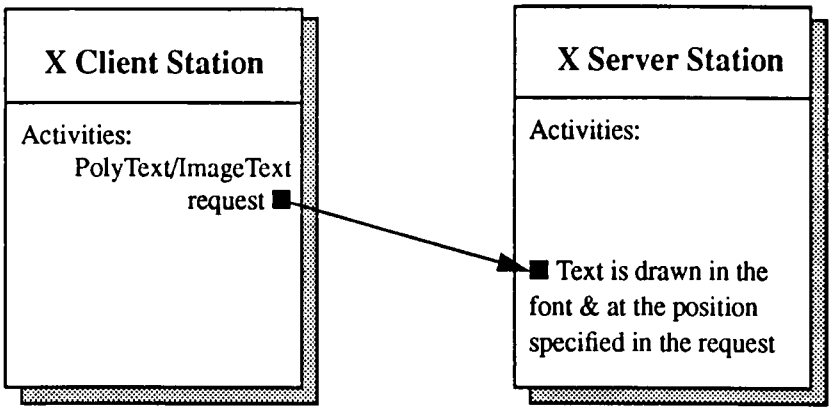


FIGURE 4 Message Flow Generated by the Displaying of Text

X protocol is only 16 bytes per character string. Consequently, pages of text can be transferred from the client to the server in an efficient manner.

Data from line-based graphics such as that generated by Computer-Aided Design (CAD) systems can also be efficiently downloaded from the client to the server by the use of either a PolyLine or PolySegment protocol request. Graphically shown in Figure 5, the PolyLine and PolySegment requests are similar to the requests used for displaying text in that no reply is generated by the server. The PolyLine and PolySegment requests can transfer approximately 175-350 vector point pairs in a single Ethernet packet. Therefore,

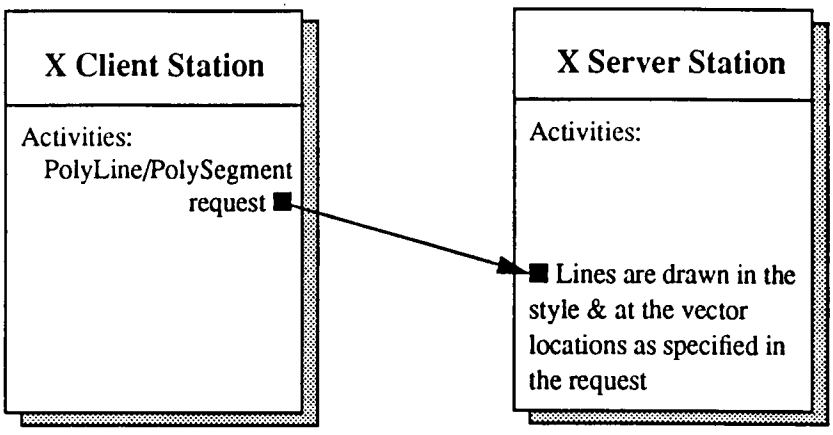


FIGURE 5 Message Flow Generated by the Displaying of Graphics

downloading a complex object can be accomplished using relatively few packets.

Each user keystroke, unless previously masked by the application program, generates two events: the KeyPress and KeyRelease events. Each event as noted earlier is 32 bytes in length. Typical response to a keystroke is to echo the character back to the server in the form of a PolyText request of 20 bytes in length. This repetitious sequence of happenings is depicted in Figure 6. Taking into account the Ethernet and TCP/IP overhead, each user keystroke generates three packets of approximately 100 bytes each.

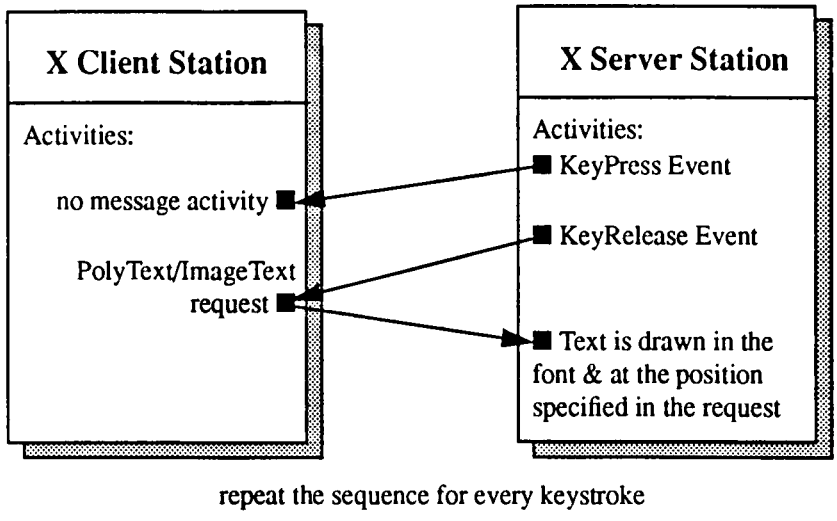


FIGURE 6 Message Flow Generated by Keyboard Input

A mouse button press is handled similarly to a keyboard key press. ButtonPress and ButtonRelease events are generated, promoting a response from the client program. This response could be to post a menu or popup some user dialog. Due to the similarity between mouse button and keyboard key events, separate discussion is not given to the mouse ButtonPress and ButtonRelease events.

Interactive mouse tracking demands more from the network than any other single transaction the user can offer. A typical algorithm for tracking a cursor that is being pulled by the mouse would be as follows:

1. the X server sends the client a 32 byte MotionNotify event.

2. the client responds by sending an eight byte QueryPointer request.
3. the server replies to the QueryPointer request with the coordinates of the pointer in a 32 byte reply.
4. the client performs some response, perhaps drawing a rectangle with a PolyRectangle request as feedback to the action of the user. The PolyRectangle request for a single rectangle is 20 bytes in length. Generally two rectangles will need to be drawn - one to erase the previous rectangle and a second to draw the new rectangle, for a total byte count of 28.
5. steps 1-4 are repeated as fast as the server can asynchronously generate the MotionNotify event.

Figure 7 illustrates these steps required for a client program to interactively track the mouse.

Thus, while performing interactive mouse tracking, four small data packets of around 100 bytes each are pushed onto the network at a potential rate of 100 of these packets per second.

Table 1 gives a summary of the influence of these user transactions upon the network. The Ethernet activity in Table 1 assumes the use of TCP/IP as the transport protocol which accounts for 40 bytes of

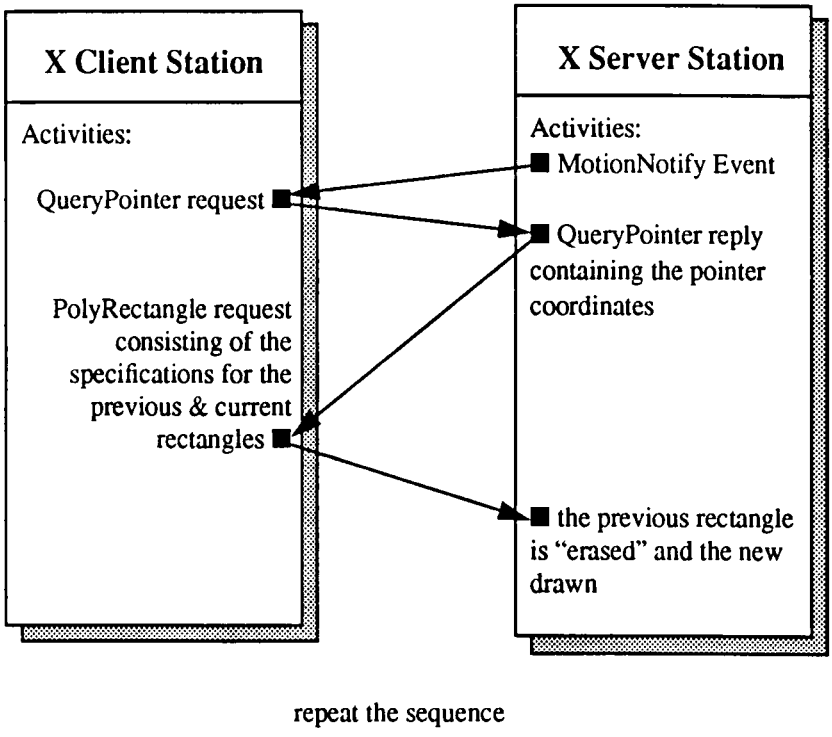


FIGURE 7 Message Flow Generated by Interactive Mouse Tracking

control overhead [Stal88] [Tane81] for each transaction. The Ethernet activity data also reflects a minimum Ethernet packet size of 46 data bytes, a maximum of 1500 data bytes, and 26 bytes of control overhead [Chor89].

Using the raw Ethernet activity data from Table 1 and making some assumptions about the frequency of occurrence, the load offered to the network can be projected for each of these user transactions along with the corresponding assumed frequency. The offered load for these

TABLE 1 Network Influence Resulting from User Transactions

User Transaction	X Protocol Message(s)	X Protocol Message(s) Length in Bytes	Ethernet Activity in Bytes
Displaying Text	PolyText/ImageText Request	16 + length of the string	82 + length of the string
Displaying Graphics	PolyLine/PolySegment Request	12 + 4/8 times the number of vectors	78 + 4/8 times the number of vectors
Keyboard Input	KeyPress Event, KeyRelease Event, PolyText Request	32, 32, 20	98, 98, 86
Mouse Cursor Tracking	MotionNotify Event, QueryPointer Request, QueryPointer Reply, PolyRectangle Request	32, 8, 32, 28	(98, 74, 98, 94) for each iteration

user transactions are tabulated in Table 2 where the offered load is presented as a percentage of the total available network bandwidth (10 Mbits/second for Ethernet).

Examination of the data in Table 2 indicates that one X session by a user is not a candidate for the single-handed overloading of a network. Neither can users type fast enough nor can display devices process graphic protocol requests fast enough to be a significant burden to the network; however, applications requiring a high level of

TABLE 2 Network Load Offered by User Transactions

Transaction	Transaction Rate	Offered Load as a Percentage of Network Bandwidth
Displaying Text	1800 characters per second (one page per 15 seconds)	0.01
Displaying Graphics	100 vectors per second (one drawing per minute)	0.07
Keyboard Input	Six characters per second (~60 words/ minute typist)	0.14
Mouse Pointer Tracking	Continuous dragging	2.91

interaction with the host system, i.e. mouse tracking, may experience slower response times.

In general, a single X session presents little impact to a network; nonetheless, many companies are moving into the X environment by exchanging ASCII terminals and RS-232 connections for X terminals and network connections. In this regard, the network impact of remote X clients as a function of the number and type of users must be considered.

Characterization and Development of the System Models

Most X-based applications issue some or all of the user transactions recorded in Table 1, resulting in the exchange of protocol messages across the network; however, this study focuses on applications whose primary operations have a direct proportional relationship to the transactions of the user. Applications such as scientific computing and software development require considerable user interaction, but, rather than the network, other parts of the computer system, such as the CPU or disk subsystem, typically become the bottleneck.

From the data in Table 1 and Table 2, the applications which will likely introduce a bottleneck into the network, are those applications which:

- have high output rates of either text or graphics, or
- have a high degree of interaction with the user, and where this interaction requires little "think time" on the part of the user.

Potential environments commonly used today and characterized by these attributes are:

- office automation areas such as word processing and electronic mail,

- desktop publishing, and
- computer aided design (CAD).

Rather than attempting to numerically characterize these users, attention is focused on the activities of the users, and the effect of these activities upon network traffic. From this perspective, the results of the investigation can be applied to any blend of these various user activities.

Each of the user activities presented earlier in this chapter and summarized in Table 1 are subdivided into the three classes or activity levels of *light*, *medium*, and *heavy* to account for the variances in typing speeds, think times, graphics use, etc. The transaction rates for these categories and classes are tabulated in Table 3.

TABLE 3 Profile of User Transactions

Usage Level	Transaction Rate				
	Text Input (words/ min)	Graphics Input (vectors/min)	Text Output (pages/min[a])	Graphics Output (drawings/min[b])	Drag Operations (drags[c]/min)
Light	30	5	1	0.2	1

TABLE 3 Profile of User Transactions (Continued)

Usage Level	Transaction Rate				
	Text Input (words/ min)	Graphics Input (vectors/min)	Text Output (pages/min[a])	Graphics Output (drawings/min[b])	Drag Operations (drags[c]/min)
Medium	45	10	2	0.6	3
Heavy	60	20	5	1.0	5

a. 1800 characters per page.

b. 6000 vectors per drawing.

c. duration of drag operation is 2 seconds.

These user activities and their corresponding usage levels comprise the models used in this performance evaluation.

CHAPTER 3 Methods

Often times in the field of computer communications there arises a need to predict the effects on the network of a new design or a change in the current design of a system. Several choices are available when considering a tool to assist in this analysis [Stall91]:

- simple projections,
- queuing models,
- simulation models, and
- an after-the-fact analysis.

The simplest and easiest of these is to make a simple straight-line projection based upon the experience of the designer or data collected from the system. These scrap-paper calculations provide reasonable estimates for feasibility discussions; however, most communications

systems exhibit an exponential response to increases in demand. This phenomenon results anytime there is a shared facility such as the communication medium or a router, hence more accurate prediction tools are often required.

Queuing models provide good results with quick and easy calculations when the inputs to the system, such as the number or length of messages, are only approximately known. When the inputs are more accurately known, then simulation models generally provide better estimates due to the many assumptions required of queuing theory. An after-the-fact measurement and analysis obviously provides the most accurate results, but this method is not an option to the system designer.

Any of these four methods can be applied to the problems previously discussed. The data presented in TABLE 2 of CHAPTER 2 can be used to provide simple projections, but, as noted, these projections may be highly inaccurate. Consideration in this chapter is given to the latter three methods:

- queuing models,
- simulation models, and
- collection of experimental data.

Each method and the application of the method to the problem is discussed separately. Although the after-the-fact approach is generally an invalid design method, it is used here to establish the integrity of the other methods.

Queuing Models

Queuing models provide a good fit to real-world problems where items arrive at a system, wait for service, are serviced, and then depart. Queuing systems can be characterized by five elements [Tane81]:

- an arrival pattern,
- a service-time pattern,
- the number of servers,
- a queuing discipline, and
- the amount of buffer space, or queue size.

The arrival pattern describes the time interval between consecutive arrivals at the server. The time between arrivals may follow one of two extremes or may fall anywhere between. The two extremes are:

- constant as in an assembly line process or
- near random as represented by customer arrivals at a bank.

The amount of time consumed by the server while servicing items is characterized by the service-time pattern. In a bank, those customers making deposits require less service time than those applying for a loan. Continuing in the use of the banking analogy, the number of servers represents the number of tellers processing customers from a waiting line.

The queuing discipline[1] describes the method by which the server selects the next item from the queue for service. Banks use the first-come, first-served queuing discipline while a communications system might utilize a queuing discipline which processes those items that have the shorter service times first, while holding the longer service-time items until later.

Queuing theory is mathematically complex, and, therefore, the ensuing simplifying assumptions are generally made [Stall91]:

- *queue size*: In real-world systems, all queues are finite, and thus items can be lost from the system. Typically, an infinite queue size can be assumed with only minor differences in the analysis.

1. The queuing calculations presented in this chapter and the results and discussions of subsequent chapters apply to any single-server queuing discipline that *does not* depend upon the service time.

- *item population*: Usually, an infinite population is assumed. A finite population adversely affects the arrival pattern. Classically, network problems assuming an infinite population have been handled without detriment.
- *arrival pattern*: Almost invariably, the arrival rate is assumed to observe the Poisson distribution, or, in other words, the arrival of items into the system occur randomly and independently of each other. This arrival pattern is often described in literature as having an exponentially distributed interarrival time, or the times between arrivals follow the exponential distribution. Without this assumption, queuing analysis becomes unwieldy.

An X/Y/n notation is used to describe queuing systems, where X is the arrival pattern, Y is the service-time pattern, and n is the number of servers. The most common distributions for X and Y are:

G - general,
M - exponential distribution, and
D - deterministic arrivals or fixed length service.

A single-server model with Poisson arrivals and exponential service times is thusly referred to as an M/M/1 model and is illustrated in Figure 8.

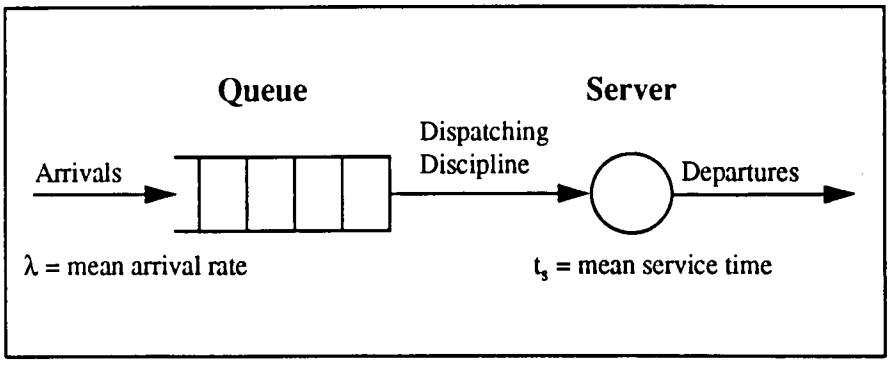


FIGURE 8 Single-Server Queuing Model

The arrival rate, denoted as λ, represents the average rate at which items arrive at the system and is measured in items per second. Each item requires a certain amount of service time by the server, t_S, measured in seconds. In a steady-state condition, the fraction of time that the server is busy, is expressed as

$$\rho = \lambda \times t_s \qquad \textbf{(EQ 1)}$$

and is referred to as the server utilization.

The mean time spent in the queue is the sum of the mean time spent waiting on service and the mean service time, or,

$$t_q = t_w + t_s \qquad \textbf{(EQ 2)}$$

The mean number of items in the queue, in a steady-state condition, waiting for service is

$$w = \lambda \times t_w \qquad \textbf{(EQ 3)}$$

Substituting **EQ 3** into **EQ 2**, the mean time spent in the system becomes

$$t_q = \frac{w}{\lambda} + t_s \qquad \textbf{(EQ 4)}$$

Khintchine and Polloczek [Mart72] developed the basic theorem of single-server queuing theory; it defines the mean number of items waiting in the queue as

$$w = \frac{\rho^2}{2(1-\rho)}\left\{1 + \left[\frac{\sigma_{t_s}}{t_s}\right]^2\right\} \qquad \textbf{(EQ 5)}$$

where σ_{t_s} is the standard deviation of the service time.

Substituting **EQ 5** into **EQ 4** yields a formula for the mean time spent in the system of

$$t_q = t_s + \frac{\rho \times t_s}{2(1-\rho)}\left\{1 + \left[\left(\frac{\sigma_{t_s}}{t_s}\right)^2\right]\right\} \qquad \textbf{(EQ 6)}$$

The term

$$\frac{1}{2}\left\{1 + \left[\left(\frac{\sigma_{t_s}}{t_s}\right)^2\right]\right\}$$

is dependent upon the service-time pattern.

When the service times are constant, the standard deviation is zero and **EQ 6** reduces to

$$t_q = t_s\left[1 + \frac{\rho}{2(1-\rho)}\right] \qquad \textbf{(EQ 7)}$$

When the service times are exponentially distributed, the standard deviation equals the mean, and the formula for the mean time in the queue is

$$t_q = \frac{t_s}{1-\rho} \qquad \textbf{(EQ 8)}$$

Most service times in computer systems fall between the constant and exponential service-time cases. Systems rarely exhibit entirely constant service times, and, similarly, the dispersion of service times is not often as great as the case for random service times. The random service-time case is often referred to as the "worst case," and many designs are done using exponential service times to give conservative results [Mart72].

Figure 9 shows curves plotting the time spent in the queue, t_q, against the utilization, ρ, for different values of σ_{t_s}, the standard deviation of the service time. The uppermost curve is for exponentially distributed service times and the lowermost curve is for constant service times. Inspecting the slope of these curves reveals that when the utilization rises above 80 percent, the time spent in the queue grows at a

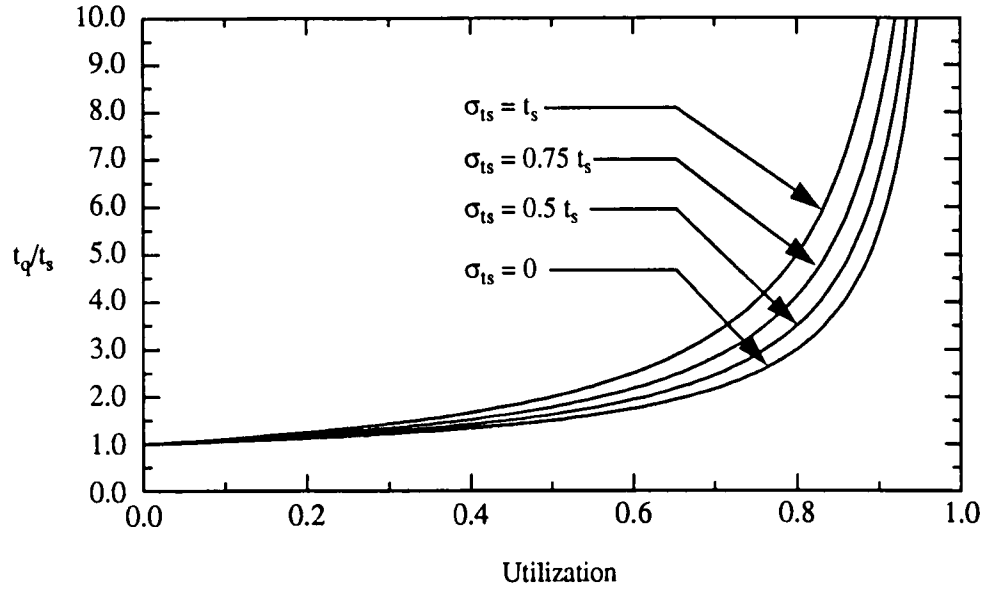


FIGURE 9 Mean Queuing Times for Single-Server Queues

disturbing rate. This is an extremely important factor to be considered in the design of communication systems. If a system is designed with utilizations of more than 80 percent, then small increases in traffic may cause serious degradation of performance.

Applying these queuing principles to an Ethernet model, the queue server is the ether or communication medium, the dispatching discipline is the binary exponential backoff algorithm, the mean service time is the time required to transmit the mean length message across the ether, and the mean arrival rate is the arrival rate of mean

length messages. The user response time is t_q -- the time spent in the queue, and the network utilization is the server utilization ρ.

Several assumptions have already been made and justified in the preceding formula derivations; yet another assumption regarding the distribution of service times in Ethernet remains. Data collected by others [Shoc80] indicate an assumption of exponentially distributed service times is reasonable.

Given these assumptions, the performance of an Ethernet, as measured in terms of the network utilization and user response time, can be calculated using **EQ 1** and **EQ 8** by knowing only the mean service time and the mean arrival rate. The mean service time, t_s, can be calculated from Table 1, and, the mean arrival rate, λ, from Table 3. These values are presented in Table 4.

TABLE 4 Mean Values Used in Queuing Models

User Activity	Activity Class	Mean Service Time (μseconds)	Mean Arrival Rate (arrivals/second)
Text Input	Light	225.6	3.0000
	Medium		4.5000
	Heavy		6.0000
Text Output	Light	1571.2	0.0167
	Medium		0.0333
	Heavy		0.0833

TABLE 4 Mean Values Used in Queuing Models (Continued)

User Activity	Activity Class	Mean Service Time (μseconds)	Mean Arrival Rate (arrivals/second)
Graphics Input	Light	382.4	0.0833
	Medium		0.1667
	Heavy		0.3333
Graphics Output	Light	40396.0	0.0033
	Medium		0.0100
	Heavy		0.0167
Mouse Tracking	Light	43680.0	0.0167
	Medium		0.0500
	Heavy		0.0833

Simulations

One of the most powerful tools that can be employed in evaluating the requirements of a data-communications system is simulation. With simulations, reality can be modeled in great detail and many of the assumptions required in queuing theory can be eliminated. The model behaves in many respects like the real-world system, but, it is greatly simplified.

Simulator programs appear in three general levels [Mart72]. First, the model can be programmed in a high-level computer language such as C or FORTRAN. Constructing a large and complex model by this method is generally expensive and has limited scope.

Secondly, a variety of general-purpose simulation languages is commercially available. Such languages include GPSS, SIMSCRIPT, and SIMULA. Models compiled from these languages allow for easy modification, are highly flexible, and can model most any system.

Finally, complete simulator programs for the type of equipment in question may be available. Two such products for modeling Ethernet systems are BONeS (Block Oriented Network Simulator) from Comdisco Systems and LANNET II.5 from CACI Corporation. Purely due to the ready availability to the author, LANNET II.5 was used to model the problems presented herein. The reader is encouraged to peruse the discussion of LANNET II.5 contained in APPENDIX B before continuing.

Use of the LANNET building blocks to describe the problems considered in CHAPTER 2 is fairly straightforward, and, therefore, detailed examination and explanation is limited to the *medium* class of the text input user profile. The LAN is defined by the problem as an IEEE 802.3 Ethernet, whose definition is supplied as a part of LANNET. The consideration of a single LAN eliminates the gateway, route, and destination mix building blocks from the model. Station building blocks are used to represent the client and server of the X Window System.

As reported in APPENDIX B, network traffic is simulated through the activities of a station. An activity is a list of actions to be performed in accomplishing that activity. These activity actions correlate nicely with the X Protocol message exchanges outlined in CHAPTER 2; a message action (the sending of a message from one station to another) is required for each X protocol message transmitted over the LAN.

A message action is described by the name of the message action, the destination of the message, and the length of the message. The attributes of these message actions and their corresponding X protocol messages are tabulated in Table 5. Note that the message lengths differ from the Ethernet message lengths of Table 1 because LANNET automatically accounts for the 26 bytes of Ethernet overhead; whereas this overhead was also included in the bit-count of the messages in Table 1.

The activities of the X Window System client and server stations are taken directly from Figure 6 on page 22 for simulating keyboard activity. Thus the server station has two activities:

- keyboard entry of 45 words per minute, and

TABLE 5 Actions List for Medium Class, Text Input User Profile, LANNET Model

LANNET Action Name	X Protocol Message (if any)	Action Type	Action Length[a]
text display request - one char	PolyText/ ImageText Request for a single character	message	60
keypress event	KeyPress Event	message	72
keyrelease event	KeyRelease Event	message	72
draw text - one char	N/A	processing	0

a. This column contains the length in bytes for actions of type message, or, the processing time in seconds for processing type actions.

- echoing the keyboard-entered character in response to the PolyText request of the client.

Assuming the average size of an English word to be six characters, the first activity consists of the *keypress event* and *keyrelease event* message actions from Table 5. The activity is scheduled by an average delay time of 0.22 seconds (45 words per minute at 6 characters per word = 4.5 characters per second, or one character every 0.22 seconds) for the medium usage level. In order to more

closely approximate a real-world situation, an exponentially distributed delay time with a mean of 0.22 seconds is chosen.

The second activity of the server station is a processing activity scheduled by receiving the *text display request - one char* message from the client station. For the purpose of evaluating the network impact, it is assumed that the station has an infinite processing capability, such that no subjective bias is introduced into the study.

The sole activity of the client station is processing keyboard events. This activity is scheduled by the receipt of both the *keypress event* message and the *keyrelease event* message. This activity is composed of a single action -- the *text display request - one char* message action.

Figure 10 pictures the LANNET model previously described. A review of the model reveals that there are two types of stations: the client and the server. The client station has one activity, and the server station has two activities. Actions for each of these activities are detailed in Table 5.

Collection of Experimental Data

The broadcast nature of Ethernet makes it particularly well suited for the passive collection of measurements: a station or instrument can

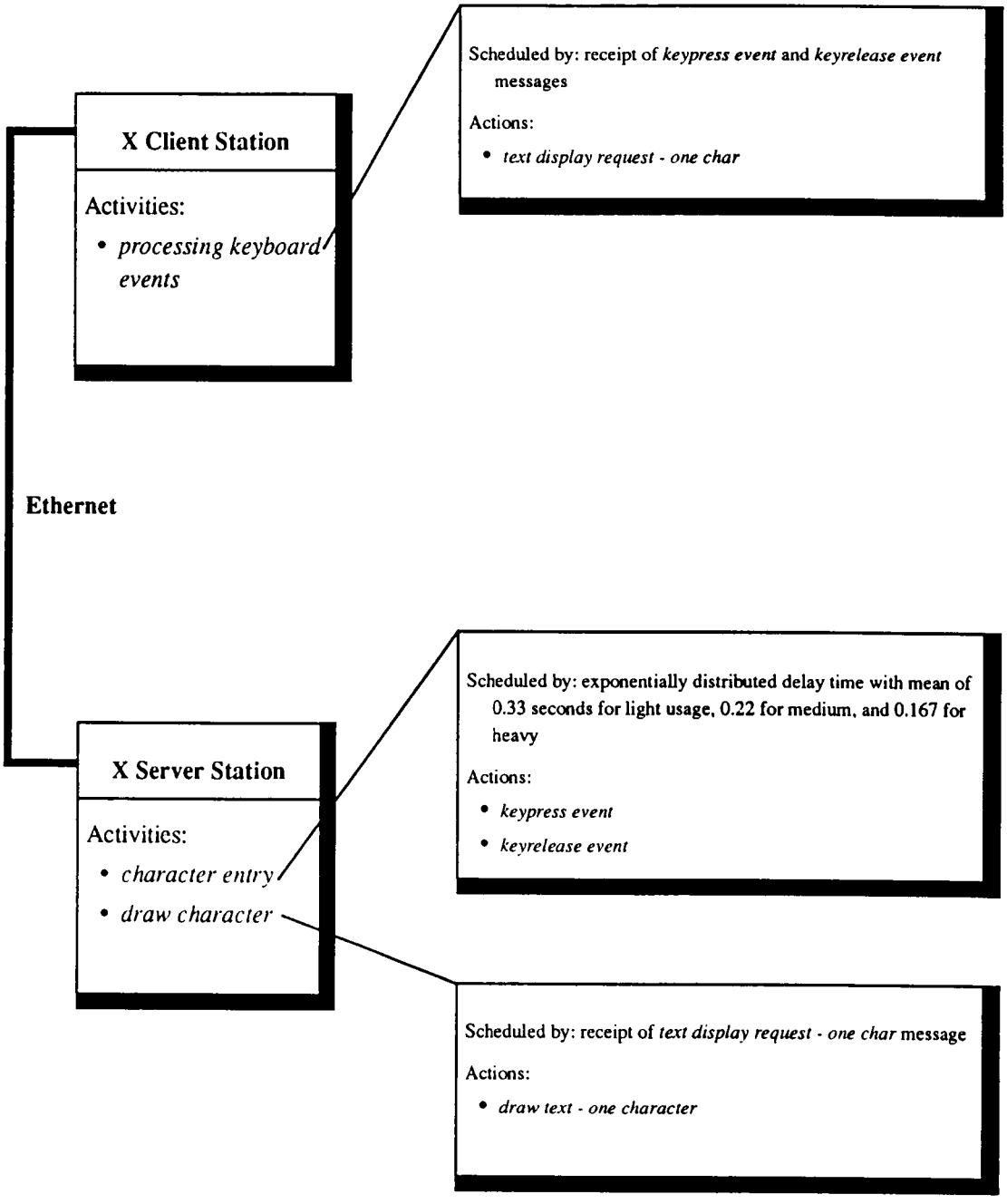


FIGURE 10 LANNET Model of the Medium Class, Text Input User Profile

sense the state of the cable and can receive all of the packets passing by while avoiding many of the Heisenberg uncertainties - the system is not altered while measurement is taking place and no portion of the communications bandwidth is consumed during testing.

The X TEST SUITE is a portion of the X Window System provided by MIT which contains numerous tests that are used in the development and testing of X. Many of these tests are provided so that vendors can test the code ported to their platform. One portion of this test suite contains programs that allow an X Window System user session to be recorded and subsequently replayed. Such is the method employed here for collecting experimental data. A range of users can be achieved by varying the number of workstation-playbacks.

CHAPTER 4

Implementation of Methods and Results

Before evaluating the study findings[1] in regard to the network impact, this chapter briefly addresses implementation of the methods outlined in CHAPTER 3 - queuing models, simulations, and collection of actual data; and also compares the data from each of these methods for variances in their results.

Implementations

For each of the five user transactions studied -- the input and output of text, the input and output of graphics, and the interactive tracking of the mouse pointer, queuing equations **EQ 1** and **EQ 8** from CHAPTER 3 were supplied with the values for mean service time

1. All data referenced herein, collected from queuing and simulation models, and from the Ethernet measurements are graphically shown in APPENDIX C.

and mean arrival rate from Table 4, to calculate the network utilization and response delay. Graphical presentations of the data resulting from this queuing analysis are contained in Figure 16 through Figure 20 of APPENDIX C. In each case the number of users was varied from 10 to 250 in increments of 10.

In a similar manner, the message flow diagrams of Figure 4 through Figure 7 were used along with the transaction rates from Table 3 to construct simulation models emulating each of the user profiles. Data gathered from the execution of these simulations are contained in Figure 21 through Figure 25.

Actual X Window System sessions were recorded while the author entered text, read text, and used the mouse. These recorded sessions were replayed on X terminals while an Ethernet "sniffer" was used to collect network utilization statistics. The number of emulated users ranged from 10 to 60 in increments of 10. The utilization data collected from these experiments are presented in Figure 31. The playback rate of each session was scaled so as to approximate the medium usage level of the respective user profile. The sniffer had no means of measuring response time, and therefore response-time data are absent from the experimental method.

General Result Considerations

Collection of actual Ethernet data were limited to the text input, text output, and mouse tracking user profiles while varying the number of users from 10 to 60. Plot data presented beyond 60 users are linear projections of the preceding data and are not actual measurements. The purpose in measuring actual Ethernet utilizations was to substantiate the data produced by the other methods, and to this end, the collected data were adequate.

In general, the values of response times presented in APPENDIX C represent the portion of delay between a user request and the fulfillment of that request, which is attributable to network delay. Specifically, the response time for each user profile is

- text input - the amount of delay between the user keystroke and the display of the typed character
- text output - the amount of delay between the user requesting the display of one page of text and the display of that text
- graphics input - the amount of delay between the input of coordinates for a vector and the display of that vector

- graphics output - the amount of delay between the user requesting the display of a single drawing and the display of that drawing
- mouse tracking - the amount of delay encountered during a one-second drag operation.

Utilization and response-time values for each of the measurement methods agree very closely in all cases except for the mouse tracking user profile. As an example of data agreement, consider the utilization data for the text input user profile presented in Figure 11. Here, as in most of the test cases, the queuing model and simulation studies provide data that reasonably matches the measured data.

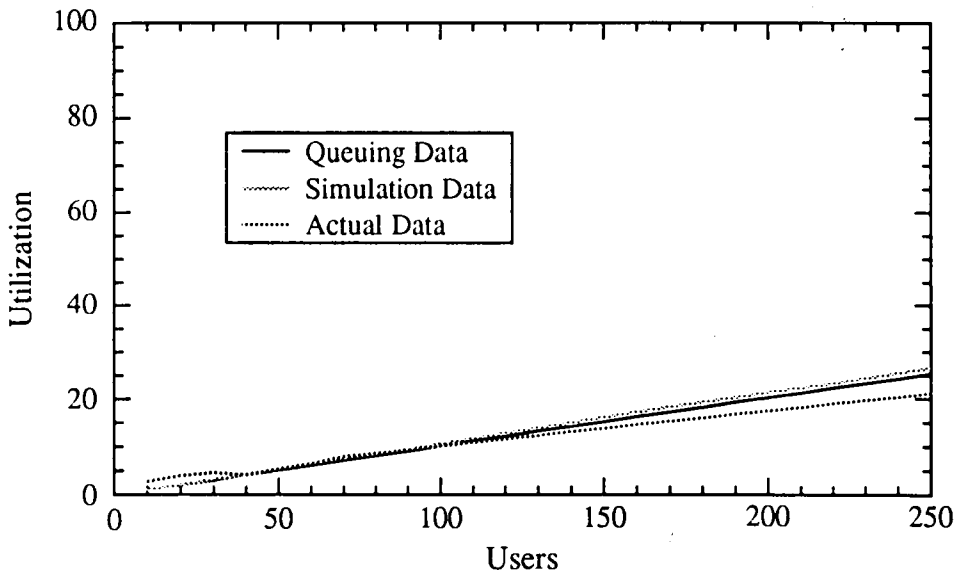


FIGURE 11 Comparison of Methods for Medium Text Input User Profile

However, simulations of tracking the mouse pointer produced results significantly different than the data resulting from the queuing model and from the linear projection of the measured Ethernet utilization. The simulation data leveled off at a utilization of approximately 60 percent while the other two methods continued to increase linearly. Why the difference?

First, the queue model does not account for the decreasing efficiency of Ethernet as the packet size decreases. The queue model assumes that once an item is removed from the queue for service, that indeed the service is completed; however, this is not the case when a collision occurs on the network -- the item whose service was terminated by the collision must be put back into the queue for future attention.

Network efficiency is a function of the packet size and the network slot time which, crudely speaking, is the worst case time required for a signal to propagate between the two farthest stations, plus account for the electronic transients. If t_s is the time required to transmit the mean-length packet, then the channel efficiency, β, is

$$\beta = \frac{t_s}{t_s + 5.4\tau} \qquad \textbf{(EQ 9)}$$

where τ is the slot time.

Consider Figure 12, which plots the theoretical data transmission rate versus the actual transmission rate as predicted by **EQ 9**, where τ = 51.2 microseconds as defined in IEEE 802.3. If the channel were used for data transmission alone, then this plot would be a straight line with a slope of one; however, since the channel is used for both data and control, the result is a curve which lies beneath the straight-line plot for data only. The difference between the two plots is the amount of bandwidth consumed by the overhead associated with channel control. Notice that as packet size increases the ratio of control overhead to data decreases and higher actual utilizations are achieved.

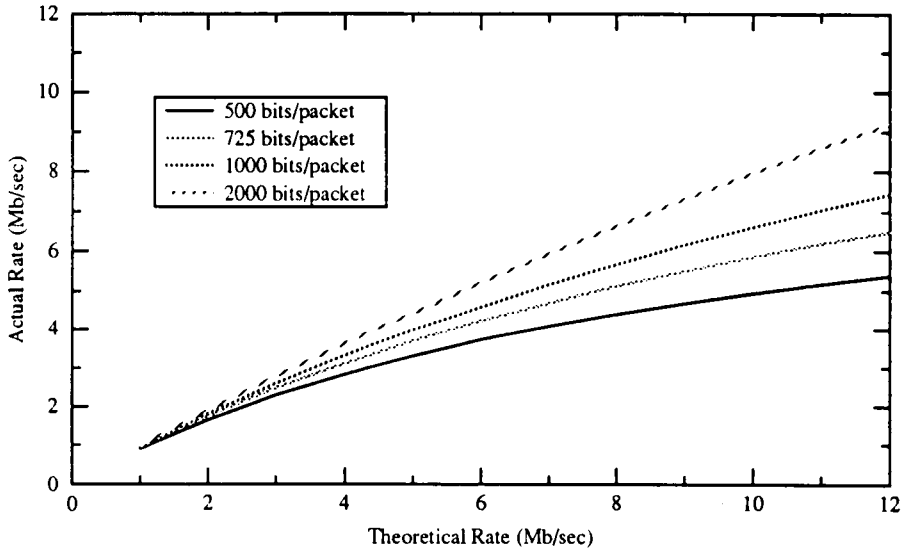


FIGURE 12 Maximum Mean Throughput Rate

The average packet size transmitted during pointer tracking is 725 bits. From Figure 12, this corresponds to an efficiency of 58.7 percent at the theoretical rate of 10 Mbits/second, thus corroborating the results provided from simulations. Similar conclusions regarding efficiency may be found in [Stal85] and [Stuc85]. Consequently, for utilization calculations where the network efficiency deviates significantly away from 100 percent, the simulation engine provided results which are more representative of the real-world situation.

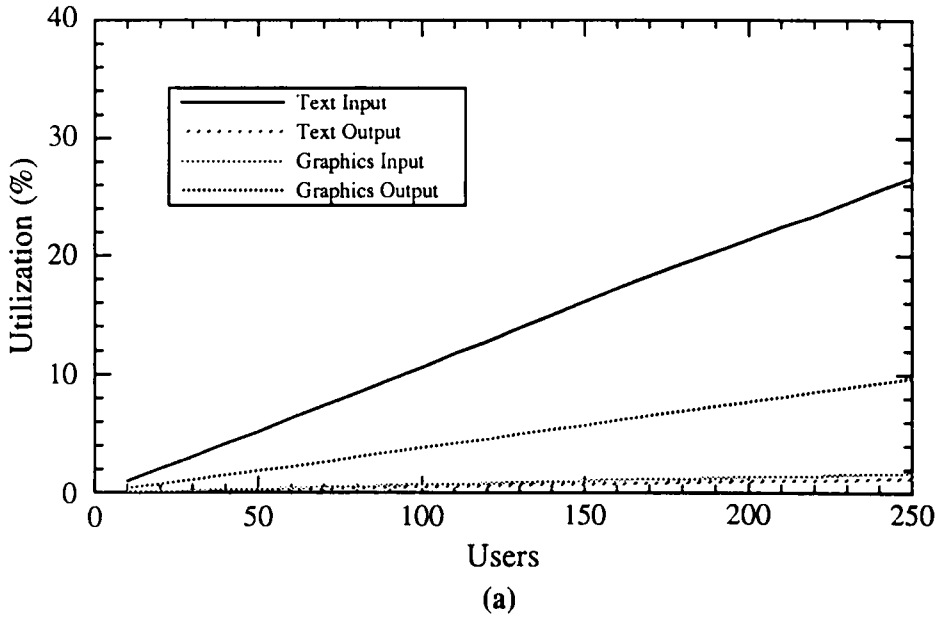
In regard to the measured data, actual data were only collected for 10 to 60 users while the data from 70 to 250 users are linear projections of the prior data. This confirms the earlier assertion that shared communication facilities exhibit non-linear behavior to increases in demand, and therefore are only appropriate for feasibility-type estimates.

In summary of the three methods employed, simulation data proved as accurate as the actual measured data while queuing theory did not accurately portray the network when operating in regions outside ideal conditions. For these reasons, further analyses and discussions of the network impact are limited to data produced by the simulation engine.

Results

Utilization and response-time results for the medium and heavy usage classes of the text input, text output, graphics input, and graphics output user profiles are shown in Figure 13 and Figure 14, respectively. Neither the text output nor the graphics input profiles offer significant loads to the network. Users cannot digest textual information quickly enough, nor can vectors be drawn fast enough for those profiles to pose network loading problems.

In contrast to these profiles, medium usage levels of the graphics output profile offers a 10 percent load to a network with 250 users while the text input profile offers comparable loads with less than 100 users. The heavy usage load for the same number of users (250 users for graphics output, 100 users for text input) is 16 percent. The graphics output load arises from the large amount of information contained in drawings that is being pushed across the network; whereas in the text input case, users type fast, and, each keystroke results in three small packets of network traffic. Although appreciable portions of the network bandwidth are consumed by these profiles, response times as perceived by the user, remain within acceptable limits.



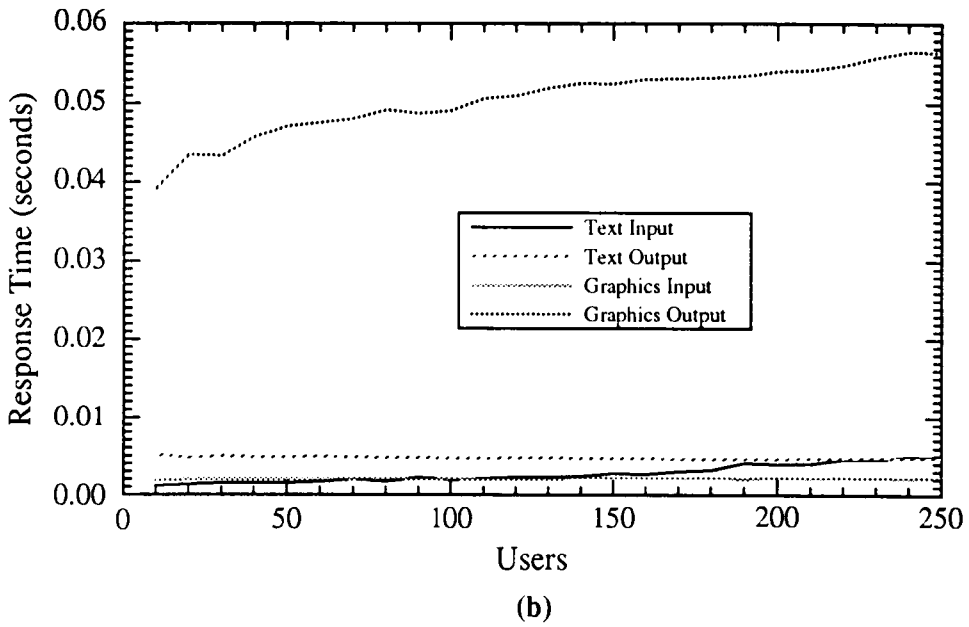
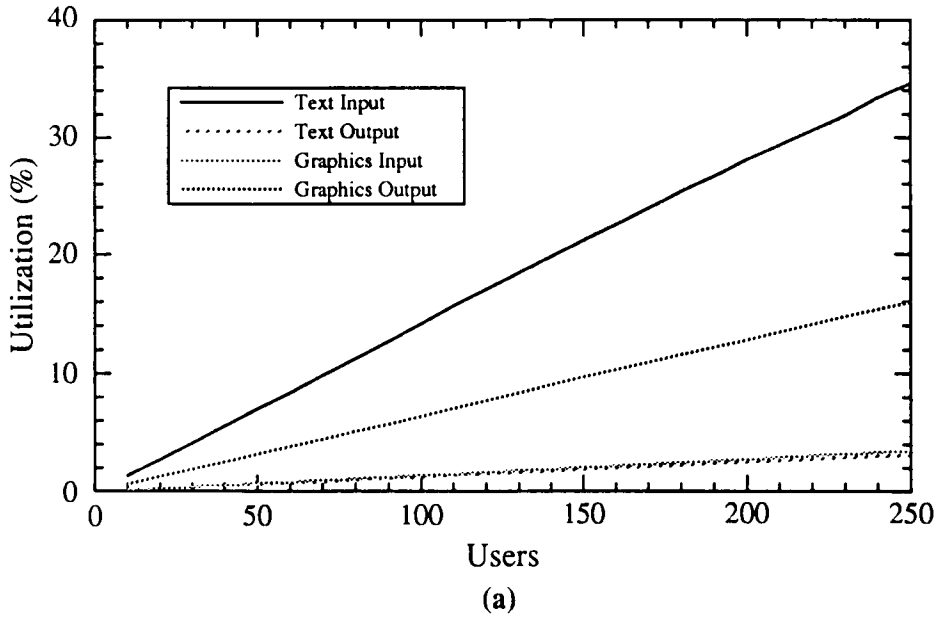
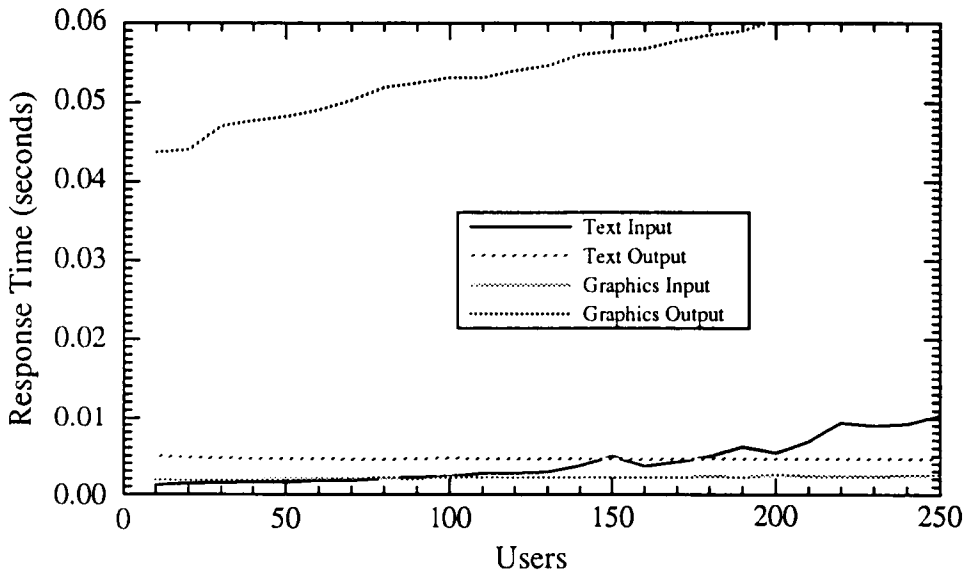


FIGURE 13 Utilizations and Response Times for the Medium Usage Profiles - Text Input, Text Output, Graphics Input, and Graphics Output



(a)



(b)

FIGURE 14 Utilizations and Response Times for the Heavy Usage Profiles - Text Input, Text Output, Graphics Input, and Graphics Output

Unlike the other user profiles, tracking mouse pointer movement during drag operations extends a more serious load to the network. As indicated in Figure 25, significant network loads are offered with as few as 30 users. This problem is further illustrated in the response times where delays of one-half second are experienced with 20 users and three seconds with 50 users -- clearly unacceptable delays.

Effect of Background Utilization

The preceding analyses and results were performed on a network under no-load conditions. The next phase of the study examined these same user profiles on a network with background utilizations of 10, 20, and 30 percent. A full Ethernet packet (1500 bytes) was chosen as the background traffic to minimize the per-packet control overhead and thus avoid any compounding effect due to interaction with user profile traffic. Recall from CHAPTER 2 that the majority of interactive X traffic is around 100 bytes per packet. Simulations were performed for the medium usage class of each user profile and the results are graphically shown in Figure 26 through Figure 30 of APPENDIX C.

These results are consistent with the prior no-load results -- the text input and output, and the graphics input and output profiles yield utilizations that are additively shifted above the no-load levels by the

amount of the background utilization. Again, response times for these profiles are generally acceptable.

Similar to the no-load situation, tracking the mouse pointer peaked out at around 65 percent. As expected, non-satisfactory delays for this profile were reached with fewer users due to the background traffic -- unacceptable levels were reached at 10 users with 30 percent background utilization compared to 20 users under no-load conditions.

CHAPTER 5

Conclusions and Recommendations

The purpose of this thesis was to investigate the demands placed upon an Ethernet by remote clients of the X Window System. Three methods of investigation were applied - queuing analysis, simulations, and an after-the-fact analysis based upon utilization statistics gathered while monitoring network traffic. For the final analysis, simulation was chosen as the principal tool to use due to its flexibility and accuracy of its output. Results from the queuing analysis and data from the actual Ethernet were used as testimonies to the validity of the simulation data. In addition, the queuing model proved to be inaccurate[1] in accounting for the decrease in network efficiency which accompanies small packet sizes.

1. see "General Result Considerations" on page 49.

In CHAPTER 2, the efficiency of the X protocol at transmitting text and graphics and the less-efficient handling of interactive activities such as keyboard entry and pointer tracking were discussed. Subsequently, models were developed targeting each of these aspects of the protocol -- the entering and display of text and graphics and the interactive tracking of mouse pointer movement during drag operations. Transaction rates for each of the models were established based upon realistic user speeds and not upon continuously queued sources. Representative samples of network utilizations for all five of the user profiles are given in Table 6.

TABLE 6 Profile Snapshot for the Medium Usage Level

User Profile	Transaction Rate	Number of Users	Utilization (%)	Response Time (seconds)
Text Input	45 words per minute	10	1.03	0.001
		50	5.17	0.001
		100	10.6	0.002
Text Output	two pages per minute	10	0.06	0.005
		50	0.26	0.005
		100	0.50	0.005
Graphics Input	10 vectors per minute	10	0.07	0.002
		50	0.34	0.002
		100	0.69	0.002
Graphics Output	one drawing per two minutes	10	0.40	0.039
		50	1.89	0.047
		100	3.83	0.049

TABLE 6 Profile Snapshot for the Medium Usage Level (Continued)

User Profile	Transaction Rate	Number of Users	Utilization (%)	Response Time (seconds)
Interactive Mouse Tracking	three two-second drag operations per minute	10	3.10	0.088
		50	15.49	3.171
		100	30.12	4.431

Under no external load, the text output and graphics input profiles produced near negligible network loading with very good response times. While response times for the text input and graphics output profiles remained acceptable, substantial portions of the LAN capacity were expended in their support.

Interactive tracking of the mouse pointer provided the worst-case performance of all users profiles studied. Peak network utilization[1] was reached at 100 "heavy" users, and unacceptable response times were reached with as few as 20 users.

Inducing background loads[2] of 10, 20, and 30 percent utilization into the study did not reveal any new information. Utilizations were additively shifted by the amount of background utilization; and

1. The actual peak utilization for a 10 Mbits/second CSMA/CD with packet lengths of 100 bytes is approximately 60 percent. Reference "General Result Considerations" on page 49.

2. Background traffic was generated using full (1500 bytes) Ethernet packets. Utilization data could be significantly different when using minimal sized packets. See "General Result Considerations" on page 49 and "Effect of Background Utilization" on page 57 for more details.

response times remained acceptable for all profiles except the mouse tracking profile, where unsatisfactory response times were realized with even fewer users.

Conclusions

An appraisal of these results leads to the following general observations:

- Information retrieval applications provide light network loads while maintaining good response times.
- Text entry applications can produce significant network loads yet maintain good user response.
- Applications with a high degree of user interaction can saturate the network and provide very poor response times with relatively few users.

A couple of specific applications come immediately to mind in regard to the last observation: *desktop publishing* and the *window manager.* In desktop publishing, the user is heavily involved in the sizing, positioning, and shaping of text and graphics. Many operations of this type are performed by "dragging" the pointer, which requires the mouse to be interactively tracked by the client application. In the case of the window manager, its principal services are the positioning and

sizing of windows. When the window manager is a remote client, as is the case with most X terminals, network loading becomes an issue. Having arrived at this same conclusion, some X terminal manufacturers are beginning to provide window manager clients integrated into the firmware of the terminal. For applications such as these, this study shows that less than 20 X terminals can be simultaneously supported.

Recommendations

In general, network loading is only of grave concern for highly interactive applications. For these applications, network saturation can often be avoided by arranging the network into work-group clusters composed of a few (no more than 20-30) X terminals and their respective host. Figure 15 illustrates a couple of configurations where isolation occurs between the X terminal cluster and the LAN backbone.

In the first configuration, the host system maintains a "private" sub-net used for the interactive X traffic, and consequently, the backbone traffic is reduced. The second configuration uses a bridge to isolate the LAN traffic. The first approach works well if all of the remote clients run on the work-group host, whereas the second approach

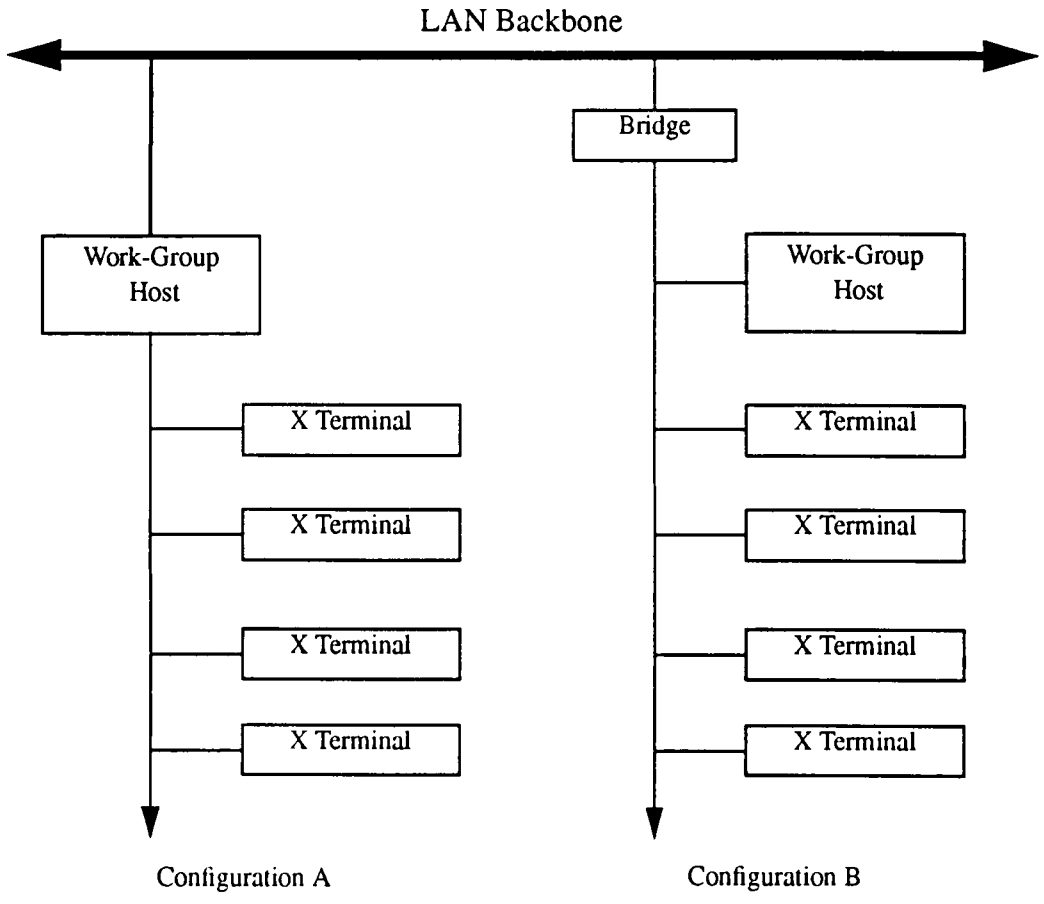


FIGURE 15 Configurations for Isolating X Traffic

works better where traffic to/from the remote client must use the main backbone.

List of References

Cerf91 — Cerf, Vinton G. "Networks." *Scientific American*: September 1991, v265, n3, pp 72-81.

Chor89 — Chorafas, Dimitris N. "*Local Area Network Reference.*" McGraw-Hill Book Company. 1989.

Coul91 — Coulson, Christopher J. "The Performance Road Test." *DEC Professional*: June 1991, v10, n6, pp 50-55.

Hewl89 — Hewlett-Packard, Inc. "*HP 700/X Family of X Window Graphics Terminals: Performance White Paper.*" December 1989.

IEEE85 — The Institute of Electrical and Electronics Engineers, Inc. "*Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Methods and Physical Layer Specifications.*" American National Standard ANSI/IEEE Std. 802.3-1985.

Mart72 — Martin, James. "*Systems Analysis for Data Transmission.*" Englewood Cliffs, NJ.: Prentice-Hall, Inc. 1972.

Mart81 — Martin, James. "*Computer Networks and Distributed Processing: Software, Techniques, and Architecture.*" Englewood Cliffs, NJ.: Prentice-Hall, Inc. 1981.

Metc76 — Metcalfe, R.M. and Boggs, D.R. "Ethernet: Distributed Packet Switching for Local Computer Networks." *Communications of the ACM*: July 1976, v19, n7, pp 395-404.

Metc91 — "Ethernet vs. Godzilla." Interview with Robert Metcalfe printed in *DBMS*: March 1991, v4, n3, pp 22-27.

ORei90 — O'Reilly & Associates, Inc. "*The X Window System Series, Volume 0: X Protocol Reference Manual.*" Sebastol, CA.: May 1990.

Pado91 "X Windows challenges you, but there's a programming payoff." *Systems Integration*: May 1991, v24, n5, pp 23.

Parr91 Parrott, Michael. "The Many Models of Distributed Processing." *Midrange Systems*: October 1, 1991, v4, n19, pp 40-41.

Sche87 Scheifler, Robert M. and Gettys, James. "The X Window System." *ACM Transactions on Graphics*: April 1987, v5, n2, pp 79-109.

Sche90 Scheifler, Robert M., Gettys, James, et. al. "*X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD.*" Digital Press, 1990.

Shoc80 Shoch, John F. and Hupp, Jon A. "Measured Performance of an Ethernet Local Network." *Communications of the ACM*: December 1980, v23, n12, pp 711-721.

Stal88 Stallings, William. "*Data and Computer Communications.*" New York, NY.: Macmillan, Inc. 1988.

Stal91 Stallings, William. "A Practical Guide to Queuing Analysis." *BYTE*: February, 1991, v16, n2, pp 309-316.

Stuc85 Stuck, B.W. and Arthurs, E. "*A Computer and Communications Network Performance Analysis Primer.*" Englewood Cliffs, NJ.: Prentice-Hall, Inc. 1985.

Tane81 Tanenbaum, Andrew S. "*Computer Networks.*" Englewood Cliffs, NJ.: Prentice-Hall, Inc. 1981.

Thar91 Thareja, Ashok K. and Ramachandran, Sridhar. "Migration From ASCII to X." *UNIX Review*: November 1991, v9, n11, pp 35-38.

Vere90 Vereeke, David. "Distributed Computing." Address to Broadband Strategies: An ElectroniCast Corporation Conference, Monterey, CA, April 23-25, 1990. Printed in *The Computer Conference Analysis Newsletter*: April 27, 1990, n254, pp 5-7.

Vere91 Vereen, Lindsey. "Window of Opportunity: the X Window System may give users of PC networks the best of all possible worlds." *LAN Magazine*: November 1991, v6, n11, pp123-128.

Witt91 Wittie, Larry D. "Computer Networks and Distributed Systems." *Computer*: September 1991, v24, n9, pp 67-76.

Youn90 Young, Douglas A. "*The X Window System: Programming and Applications with Xt, OSF/Motif Edition.*" Englewood Cliffs, NJ.: Prentice-Hall, Inc. 1990.

Zach91a Zachmann, William. "X Windows Technology Is Not for the Mainstream." *PC Week*: May 13, 1991, v8, n19, pp 57.

Zach91b Zachmann, William. "Casting X Windows as Technology or Religion." *PC Week*: June 24, 1991, v8, n25, pp 67.

Zamb91 Zamboni, Ron. "A Guide for the Confused." *Computing Canada*: August 1, 1991, v17, n16, pp 21-22.

Appendices

APPENDIX A

X Window System Protocol Encoding

This appendix contains the descriptions and encodings of the protocol messages specifically mentioned in this work. An English description of the message format is followed by a Table which describes the byte-for-byte encoding of the message. Much of the information presented herein was taken from Scheifler, Robert M., Gettys, James, et. al. "*X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD.*" Digital Press, 1990.

ButtonPress Event

The ButtonPress event is sent by the server to the client whenever the user presses a pointer button. The event source is the window that the pointer was in when the button was pressed.

TABLE 7 ButtonPress Event Encoding

Number of Bytes	Description
1	event code
1	button detail identifying the pointer button pressed
2	sequence number
4	time stamp
4	root window identification
4	event window identification
4	child window, 0 for none
2	x location in root relative coordinates
2	y location in root relative coordinates
2	x location in event window relative coordinates
2	x location in event window relative coordinates
2	state of buttons and modifier keys
1	flag indicating whether the event window and root window are a part of the same screen
1	unused

ButtonRelease Event

The ButtonRelease event is sent by the server to the client whenever the user releases a pointer button. The event source is the window that the pointer was in when the button was released.

TABLE 8 ButtonRelease Event Encoding

Number of Bytes	Description
1	event code
1	button detail identifying the pointer button released
2	sequence number
4	time stamp
4	root window identification
4	event window identification
4	child window, 0 for none
2	x location in root window relative coordinates
2	y location in root window relative coordinates
2	x location in event window relative coordinates
2	x location in event window relative coordinates
2	state of buttons and modifier keys
1	flag indicating whether the event window and root window are a part of the same screen
1	unused

ImageText Request

The ImageText request paints the character string at the baseline x and y coordinates of the drawable as specified in the request. No reply is returned by the server.

TABLE 9 ImageText Request Encoding

Number of Bytes	Description
1	request opcode
1	number of characters in the string (n)
2	request length
4	drawable identifying the window or pixmap where the text is to be drawn
4	graphics context containing the foreground color, background color, etc.
2	beginning x coordinate of the text
2	beginning y coordinate of the text
n	string of one-byte characters
p	used only for padding, p=pad(n)

KeyPress Event

The KeyPress event is sent by the server to the client whenever the user presses a keyboard key. The event source is the window that the pointer was in when the key was pressed.

TABLE 10 KeyPress Event Encoding

Number of Bytes	Description
1	event code
1	key detail identifying the keyboard key pressed
2	sequence number
4	time stamp
4	root window identification
4	event window identification
4	child window, 0 for none
2	x location in root window relative coordinates
2	y location in root window relative coordinates
2	x location in event window relative coordinates
2	y location in event window relative coordinates
2	state of buttons and modifier keys
1	flag indicating whether the event window and root window are a part of the same screen
1	unused

KeyRelease Event

The KeyRelease event is sent by the server to the client whenever the user releases a keyboard key. The event source is the window that the pointer was in when the key was released.

TABLE 11 KeyRelease Event Encoding

Number of Bytes	Description
1	event code
1	key detail identifying the keyboard key released
2	sequence number
4	time stamp
4	root window identification
4	event window identification
4	child window, 0 for none
2	x location in root window relative coordinates
2	y location in root window relative coordinates
2	x location in event window relative coordinates
2	y location in event window relative coordinates
2	state of buttons and modifier keys
1	flag indicating whether the event window and root window are a part of the same screen
1	unused

MotionNotify Event

In general, a MotionNotify event is only sent by the server to the client when the motion begins and ends in a window. If the PointerMotionHint is selected, then the server is free to send only one MotionNotify event to the client until either the key or button state changes, the pointer leaves the event window, or the client issues a QueryPointer or GetMotionEvents request.

TABLE 12 MotionNotify Event Encoding

Number of Bytes	Description
1	event code
1	event detail identifying the event as normal or a hint
2	sequence number
4	time stamp
4	root window identification
4	event window identification
4	child window, 0 for none
2	x location in root window relative coordinates
2	y location in root window relative coordinates
2	x location in event window relative coordinates
2	y location in event window relative coordinates
2	state of buttons and modifier keys
1	flag indicating whether the event window and root window are a part of the same screen
1	unused

PolyLine Request

The PolyLine request draws lines between each pair of points (point[i], point[i+1]). The lines are drawn in the order listed. The first point is always relative to the origin of the drawable; the remainder of the points may be either relative to the origin of the drawable or relative to the previous point, depending upon the coordinate mode. No reply is returned by the server.

TABLE 13 PolyLine Request Encoding

Number of Bytes	Description
1	request opcode
1	coordinate mode, either origin or previous
2	request length
4	drawable identifying the window or pixmap where the lines are to be drawn
4	graphics context containing the foreground color, background color, line style, etc.
4n	coordinates for n points where two bytes each are used for the x and y coordinate of each point

PolyRectangle Request

The PolyRectangle request draws the outlines of the specified rectangle(s) as if a five point PolyLine request were specified for each rectangle: (x,y), (x+width), (x+width, y+height), (x,y+height), and (x,y). No reply is returned by the server.

TABLE 14 PolyRectangle Request Encoding

Number of Bytes	Description
1	request opcode
1	unused
2	request length
4	drawable identifying the window or pixmap where the rectangles are to be drawn
4	graphics context containing the foreground color, background color, line style, etc.
8n	n rectangle specifications where two bytes each are used for each the x and y coordinate of the upper left corner, the width, and the height for each rectangle

PolySegment Request

This request draws a line segment between each pair of vector points [x1,y1] and [x2,y2]. The lines are drawn in the order listed. No reply is returned by the server.

TABLE 15 PolySegment Request Encoding

Number of Bytes	Description
1	request opcode
1	unused
2	request length
4	drawable identifying the window or pixmap where the line segments are to be drawn
4	graphics context containing the foreground color, background color, line style, etc.
8n	n segments of vector point pairs [x1,y1] and [x2,y2] using two bytes each for the x and y coordinate of each segment

PolyText Request

The PolyText request paints the character string at the baseline x,y coordinates of the drawable as specified in the request. No reply is returned by the server.

TABLE 16 PolyText Request Encoding

Number of Bytes	Description
1	request opcode
1	unused
2	request length
4	drawable identifying the window or pixmap where the text is to be drawn
4	graphics context containing the foreground color, background color, etc.
2	beginning x coordinate of the text
2	beginning y coordinate of the text
n	n repetitions of text items where a text item is either a font description or a text string description as described in Table 17
p	used only for padding, p=pad(n)

TABLE 17 Text Item Encoding

	Number of Bytes	Description
Text String Description	1	length of string
	1	delta offset in the x direction for the string
	m	character string of length m

TABLE 17 Text Item Encoding (Continued)

	Number of Bytes	Description
Font Description	1	font identifier code (255) delineating the font description from other possible character codes
	1	most-significant font byte (3)
	1	font byte (2)
	1	font byte (1)
	1	least-significant font byte (0)

QueryPointer Request

The QueryPointer generates a reply from the server that includes the pointer coordinates relative to the root's origin. If the same screen flag is true, then the coordinates relative to the requested window's origin are also returned. The current state of the modifier keys and the pointer buttons are also returned.

TABLE 18 QueryPointer Request Encoding

	Number of Bytes	Description
Request Encoding	1	request opcode
	1	unused
	2	request length
	4	window identifying the origin for the returned relative coordinates of the pointer

TABLE 18 QueryPointer Request Encoding (Continued)

	Number of Bytes	Description
Reply Encoding	1	reply opcode
	1	same screen flag indicating whether the requested window and the root window are a part of the same screen
	2	sequence number
	4	reply length
	4	root window identification
	4	child window identification, 0 for none
	2	x location in root window relative coordinates
	2	y location in root window relative coordinates
	2	x location in requested window relative coordinates
	2	y location in requested window relative coordinates
	2	state of buttons and modifier keys
	6	unused

APPENDIX B LANNET II.5

This appendix contains a brief description of the LANNET II.5 local area network performance analysis tool. Many details about the tool are omitted where there is no primary relevance to the immediate context. The information presented herein was taken from CACI Products Company, "*LANNET II.5 User's Manual.*", 1991.

"LANNET II.5 is a design tool which takes a user-specified LAN description and provides measures of utilization, conflicts, delay, and response times." The LAN to be simulated is described using any of the six basic building blocks:

- LAN,
- station,
- gateway,

- route,
- destination mix, and
- statistical distribution function (SDF).

The LAN is the heart of LANNET II.5 and provides connections to stations and gateways. Traffic on the LAN is the result of an Action by a Station and is in the form of a message.

Every LAN has a type, an implementation, and a connection list. The type of LAN is either collision, token ring, or token bus while the implementation is a specific case of one of these types. A library of LAN implementations patterned after the IEEE 802 specifications is a part of the tool. The connection list is a list of all components connected to the LAN. These components may be either a station or a gateway.

A station is any producer or consumer of LAN traffic. Station examples are mainframes, workstations, printers, etc. Every station must have one or more activities if it is to play an active role in the model. An activity controls the generation and processing of messages and is characterized by a scheduling method and an action list. The scheduling method determines the time at which a station enters the activity and is based either upon the receipt of a list of messages or on a time delay. When the method is a message list, the

then station waits until it has received all of the messages on the list before entering the activity. When a delay time scheduling method is used, the delay is the time between one completion of the activity and the initiating of the next occurrence of that activity.

Upon entering an activity, a list of actions is sequentially executed. The action list is described by the type of action and the number of executions. Action types are either processing, message, or file input/output. Computation internal to the station is referred to as a processing action, while a message action provides for sending a message from this station to any other station. File input/output actions allow for simulating storage on the station.

For a processing type of action, the number-of-executions attribute specifies the number of times that an action is executed before the next action in the list is processed.

A gateway is a generic term for a repeater, router, bridge, gateway, or any piece of hardware that provides bidirectional interconnection between two or more LANs. A route contains a list of gateways followed by a destination station that defines the path for a message to take in course to a final destination. If no route is specified, then a shortest path algorithm is applied.

A destination mix is a list of stations, routes, and destination mixes with a percentage associated to each. A statistical distribution function (SDF) is used to introduce a stochastic nature into a simulation. An SDF may be specified by the user or one from a pre-defined library may be used.

LANNET II.5 automatically models layers 1, 2, and 3 of the ISO/OSI model and adds the frame overhead for both the MAC and LLC sub-layers of the IEEE 802 suite. The remaining layers of the ISO/OSI model can be simulated by presenting to the LAN carefully chosen traffic loads.

APPENDIX C Aggregate Data

Data presented in this appendix comprise the totality of all raw data collected from the queuing and simulation models and the experimental data collected from an actual Ethernet. All data is presented by plots for the utilization or response time versus the number of users.

Refer to CHAPTER 2 for definitions and descriptions of the various user profiles. Discussion devoted to the interpretation of this data is contained in CHAPTER 4 and CHAPTER 5.

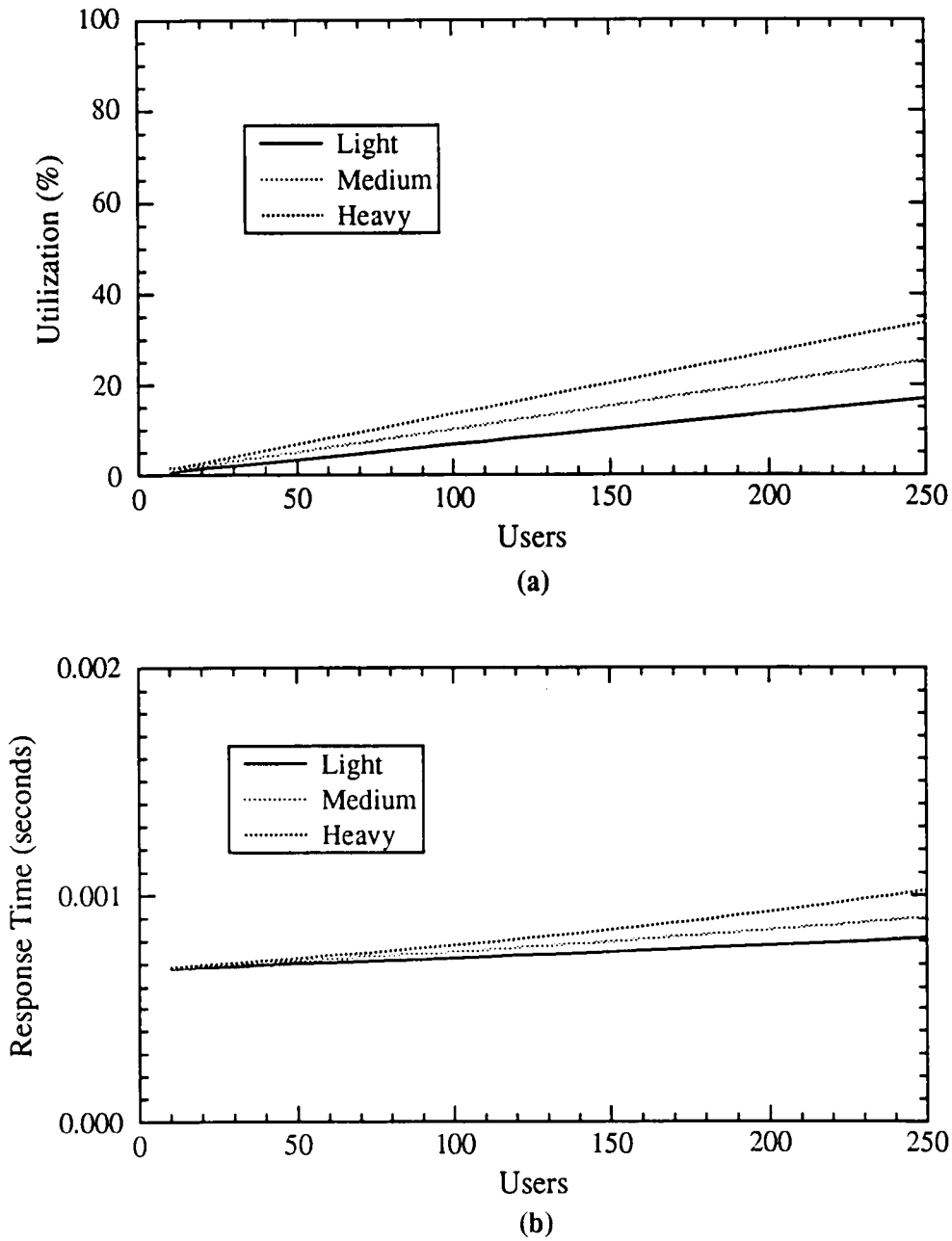
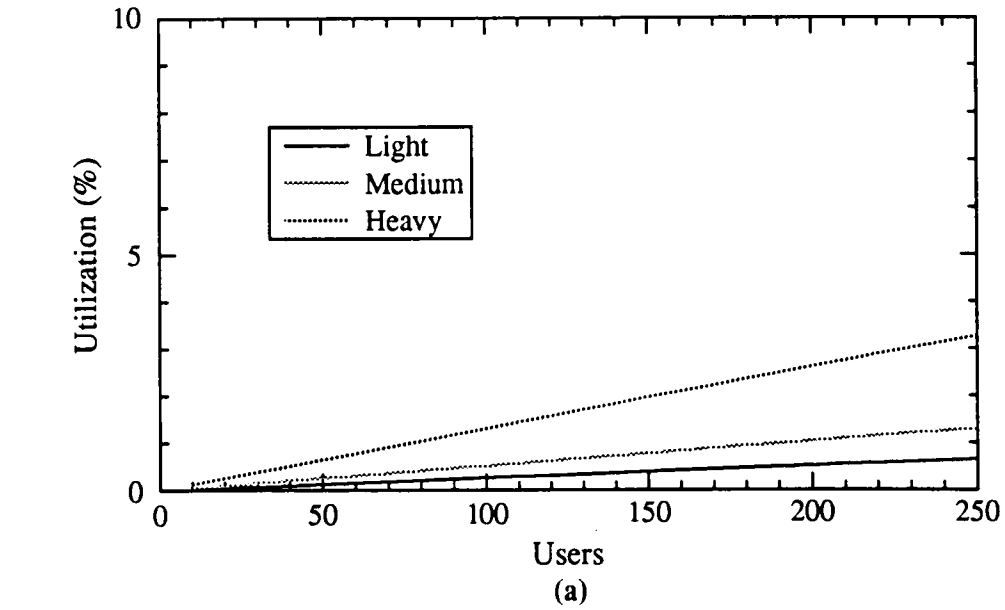


FIGURE 16 Text Input User Profile, Queuing Models



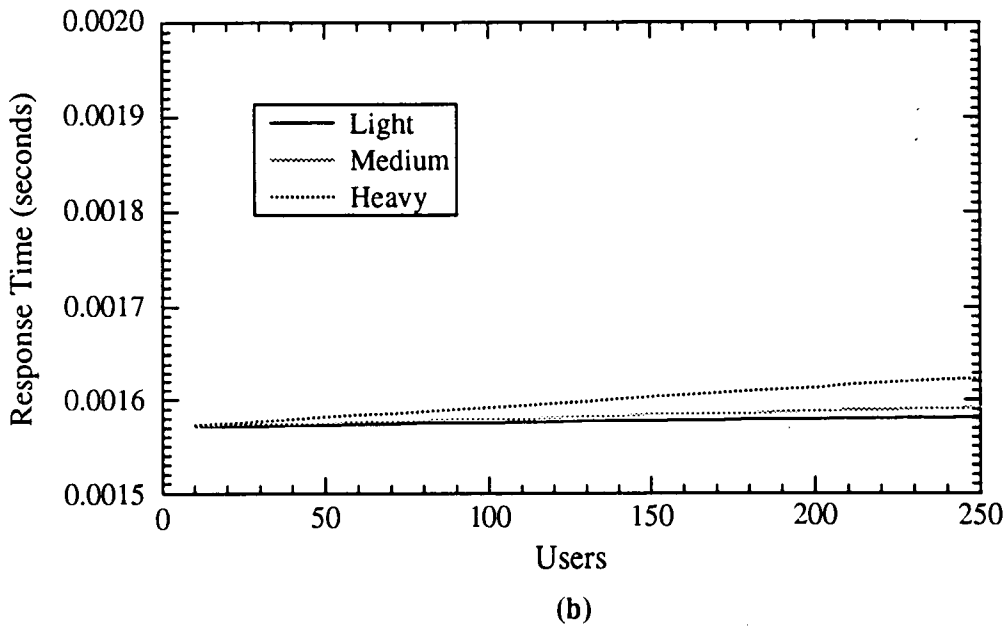


FIGURE 17 Text Output User Profile, Queuing Models

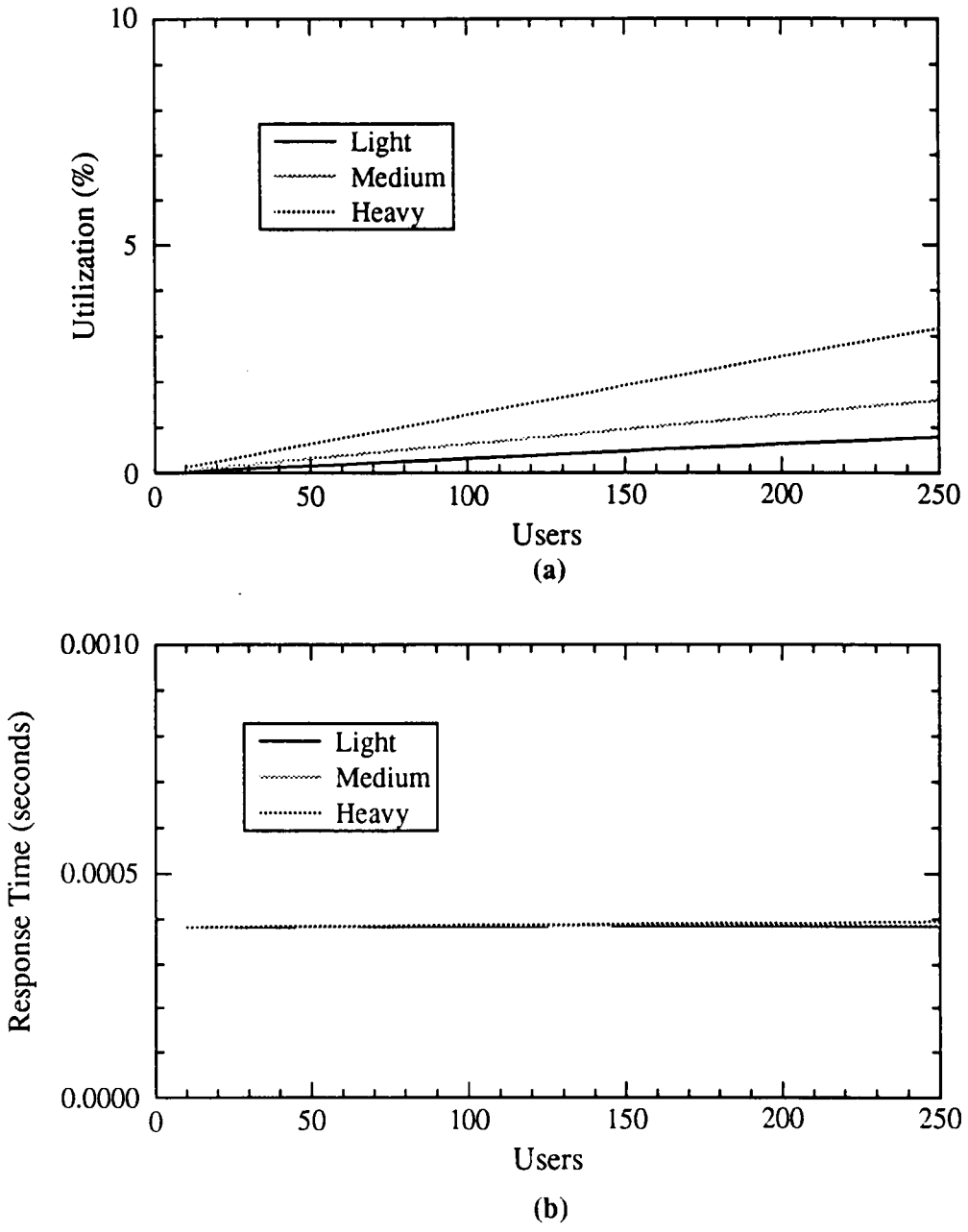


FIGURE 18 Graphics Input User Profile, Queuing Models

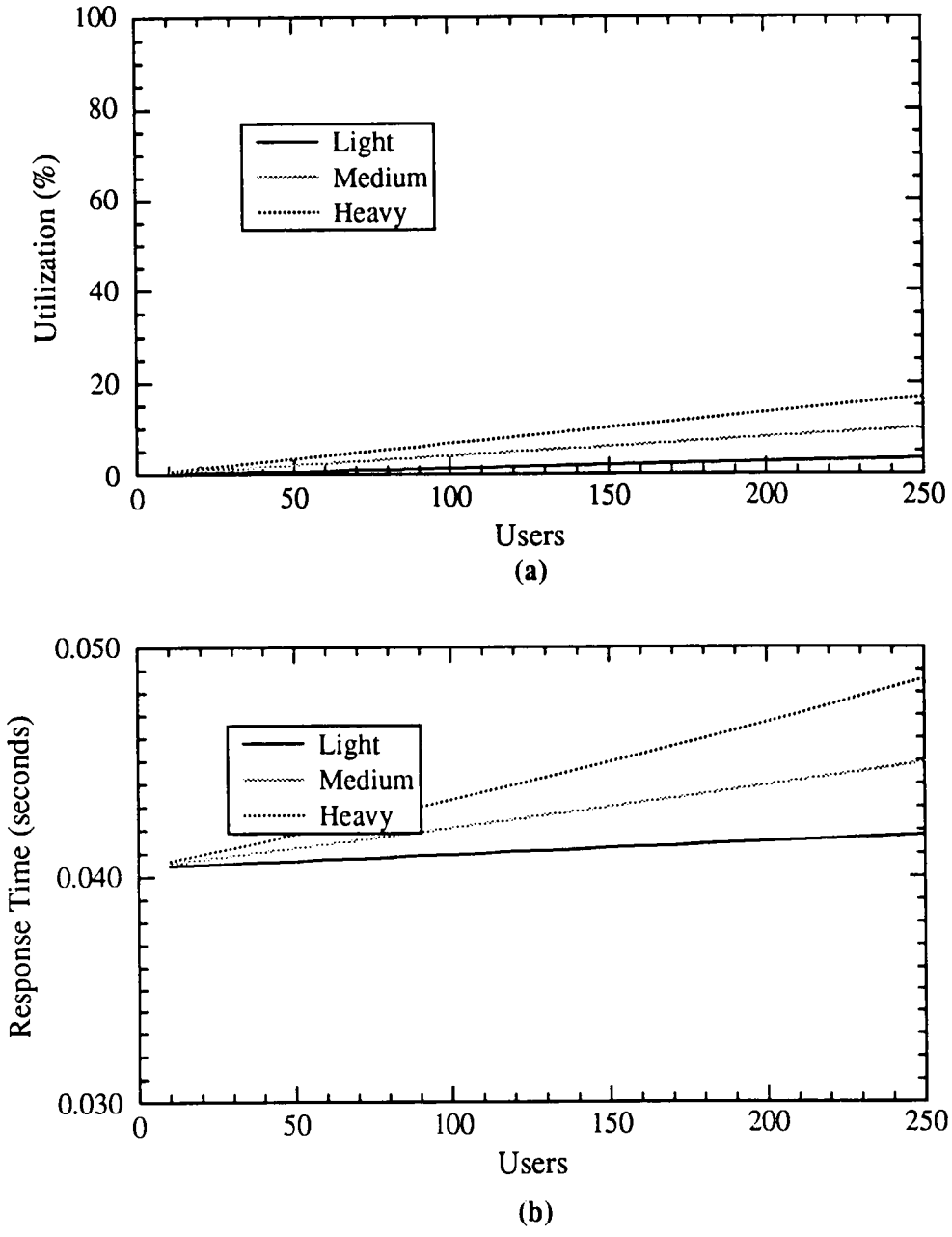
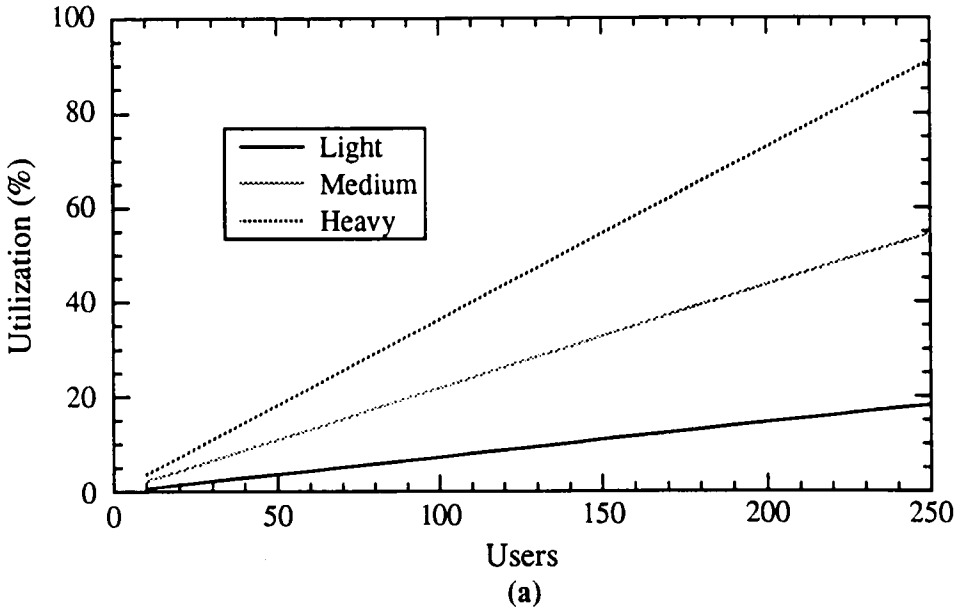


FIGURE 19 Graphics Output User Profile, Queuing Models



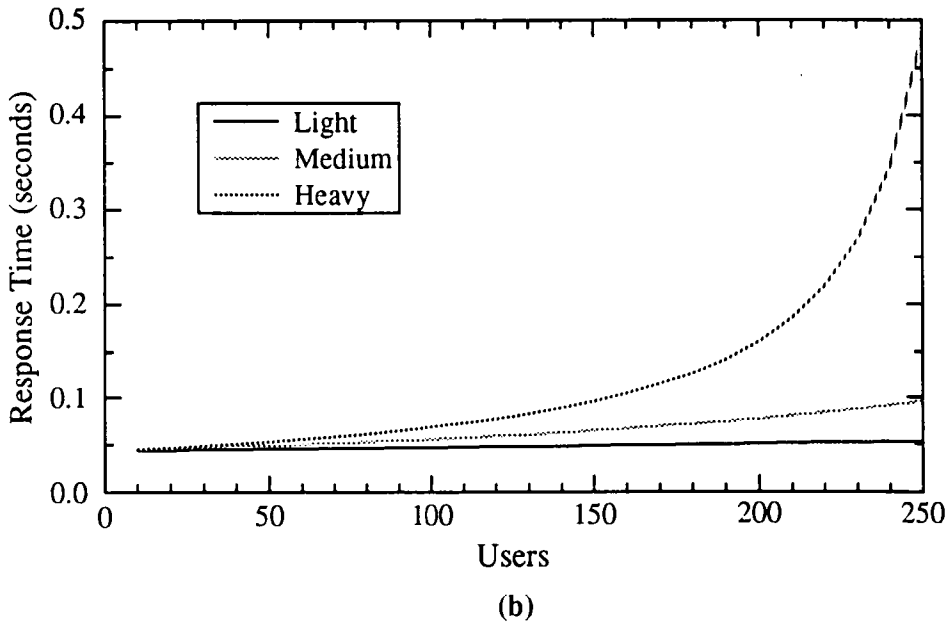


FIGURE 20 Mouse Tracking User Profile, Queuing Models

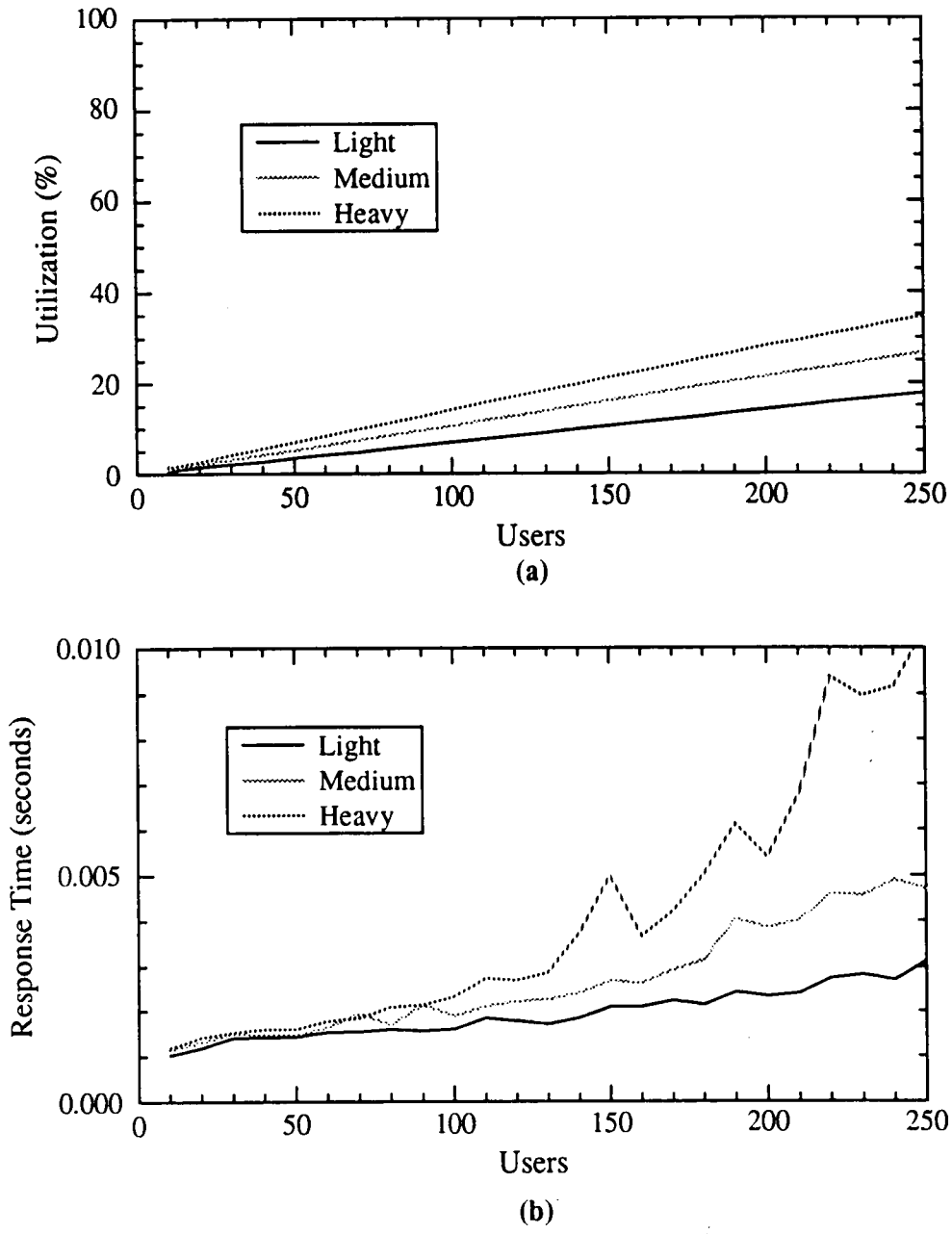


FIGURE 21 Text Input User Profile, LANNET Models

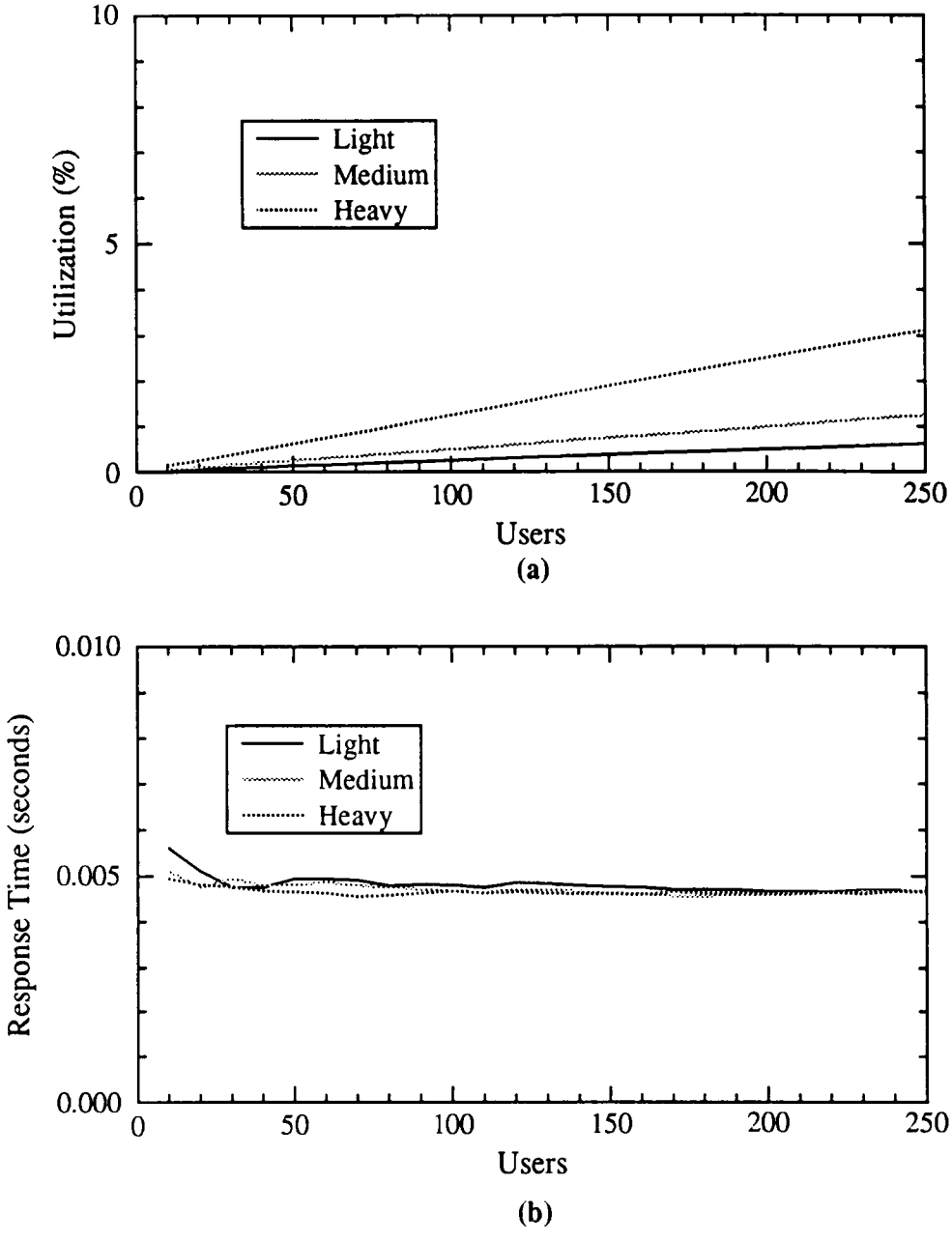


FIGURE 22 Text Output User Profile, LANNET Models

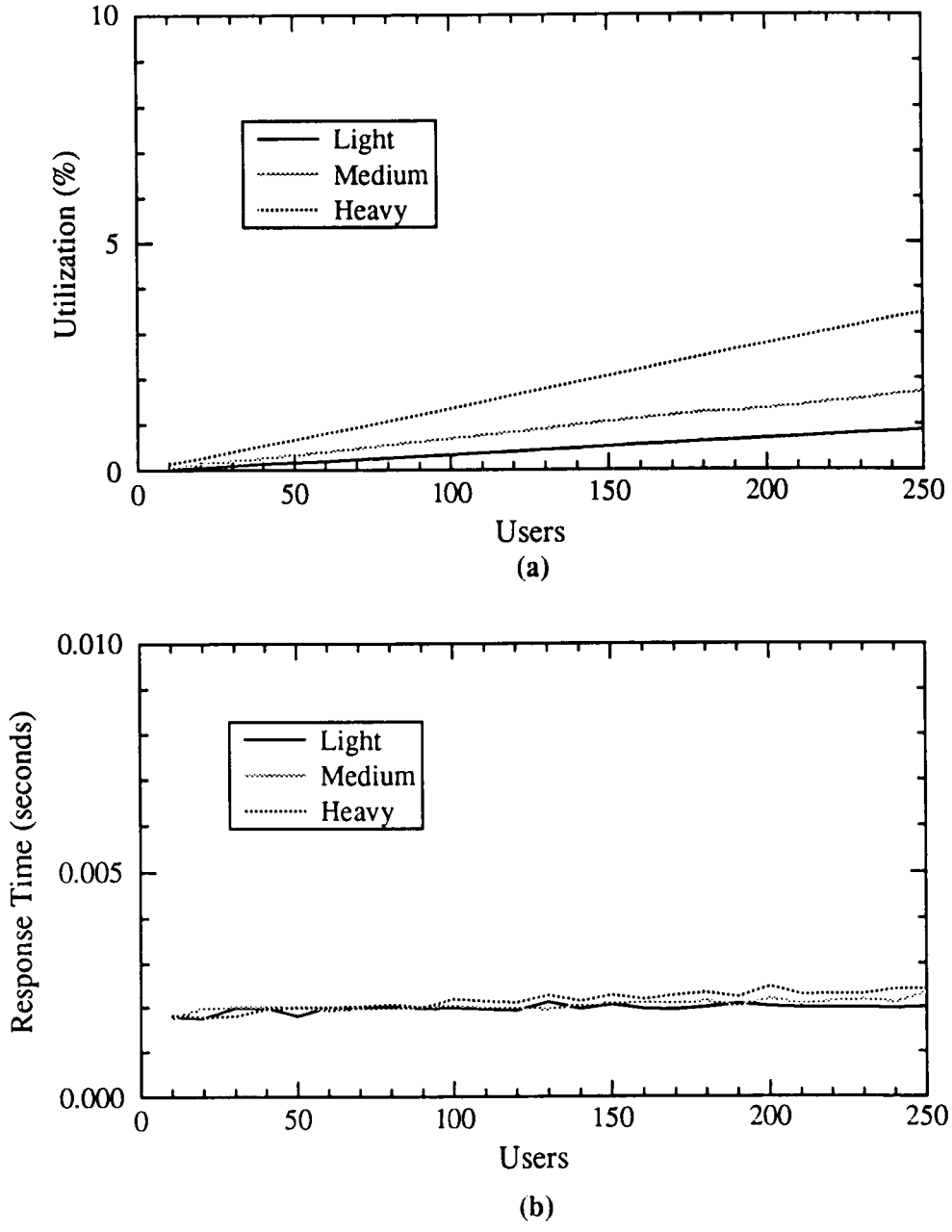


FIGURE 23 Graphics Input User Profile, LANNET Models

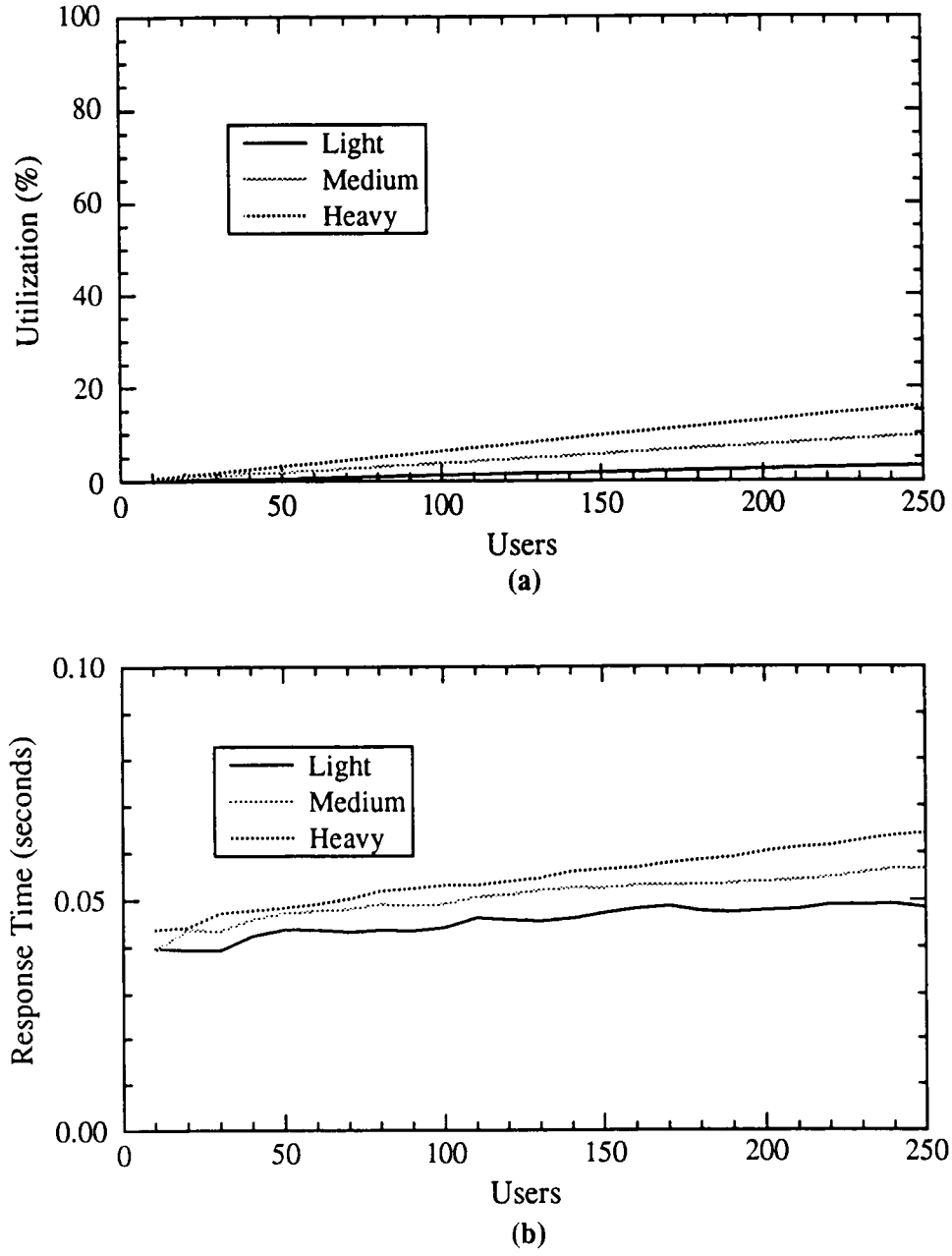


FIGURE 24 Graphics Output User Profile, LANNET Models

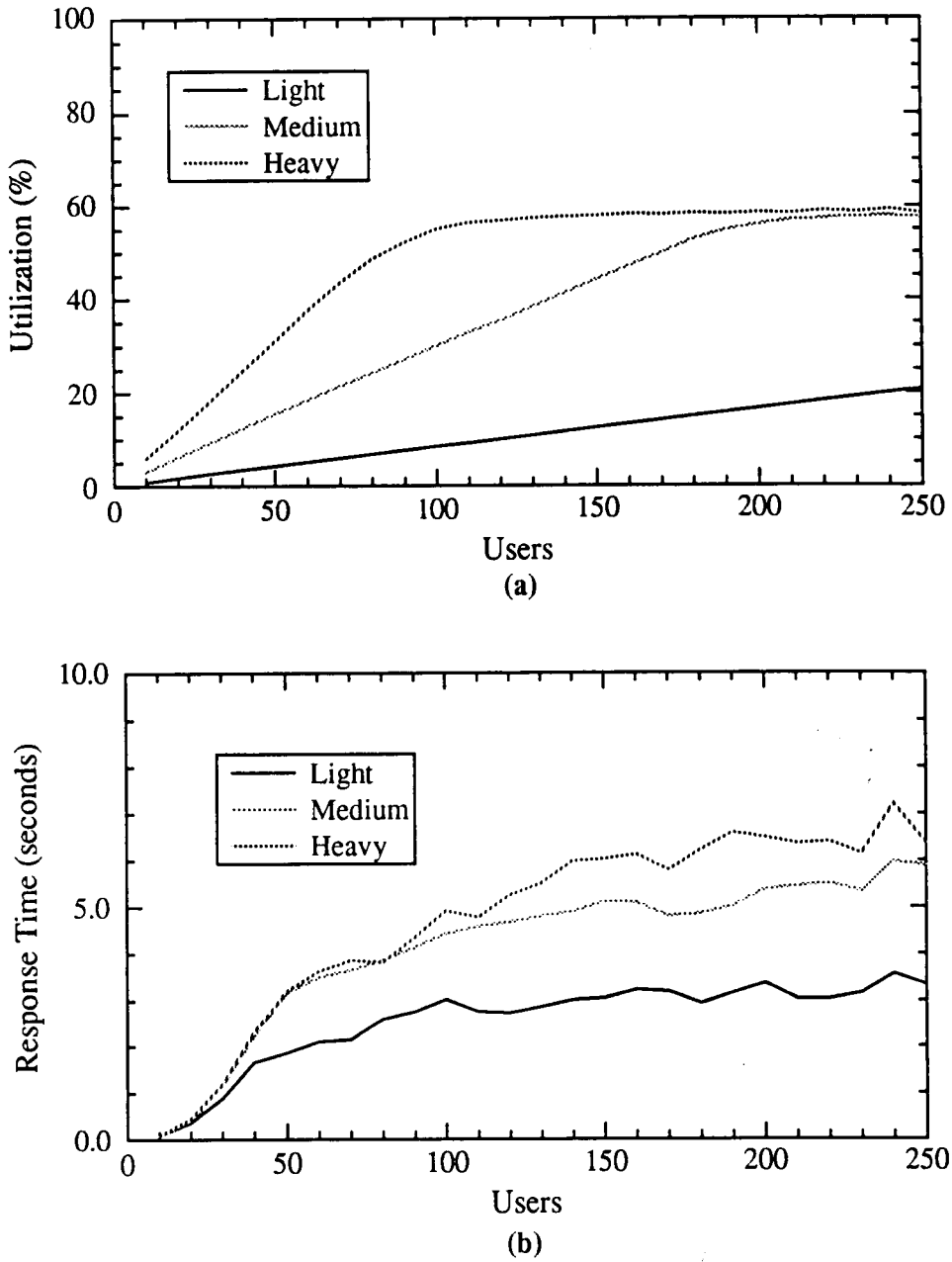


FIGURE 25 Mouse Tracking User Profile, LANNET Models

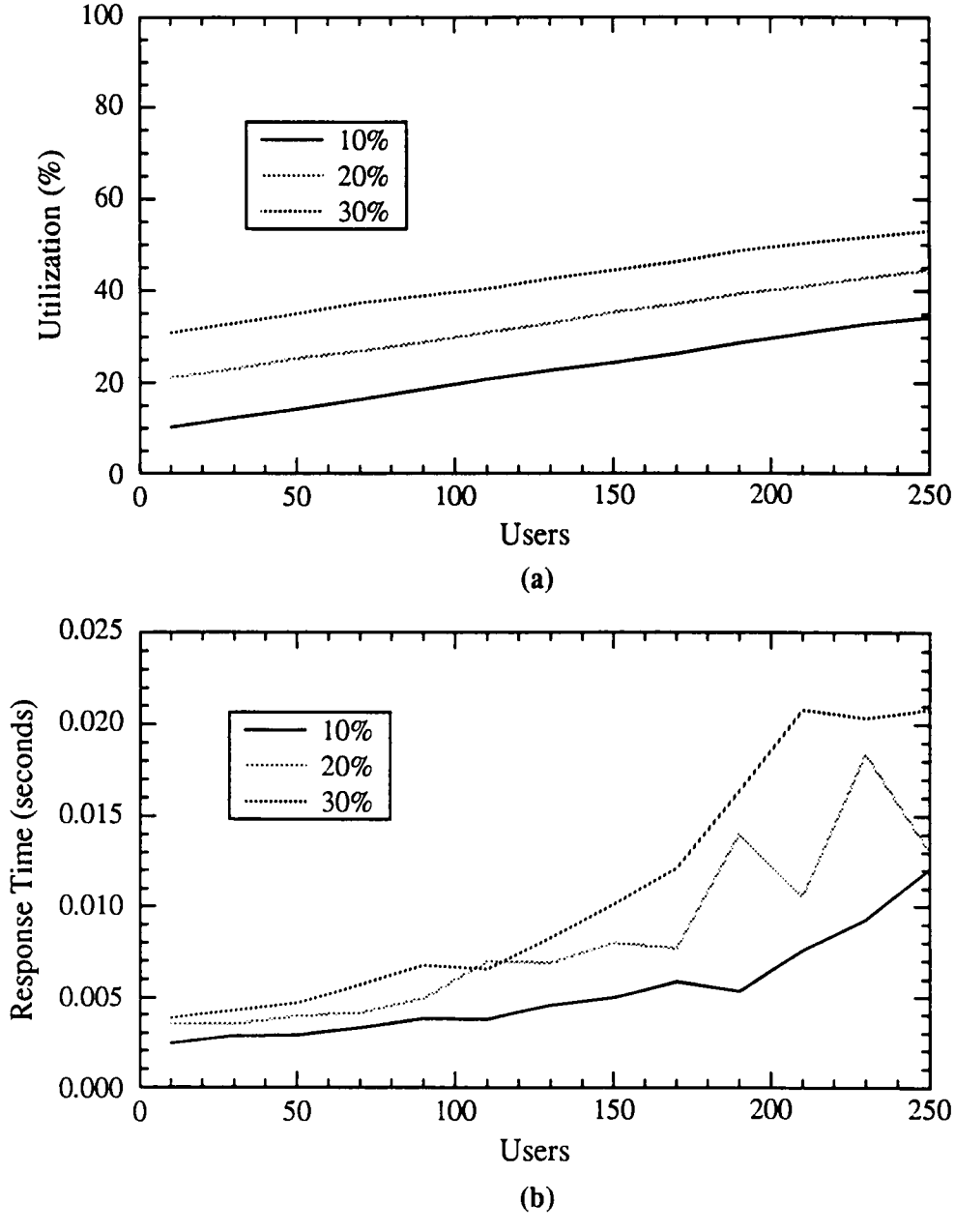


FIGURE 26 Text Input User Profile (Medium) with Background Utilizations of 10, 20, and 30 Percent

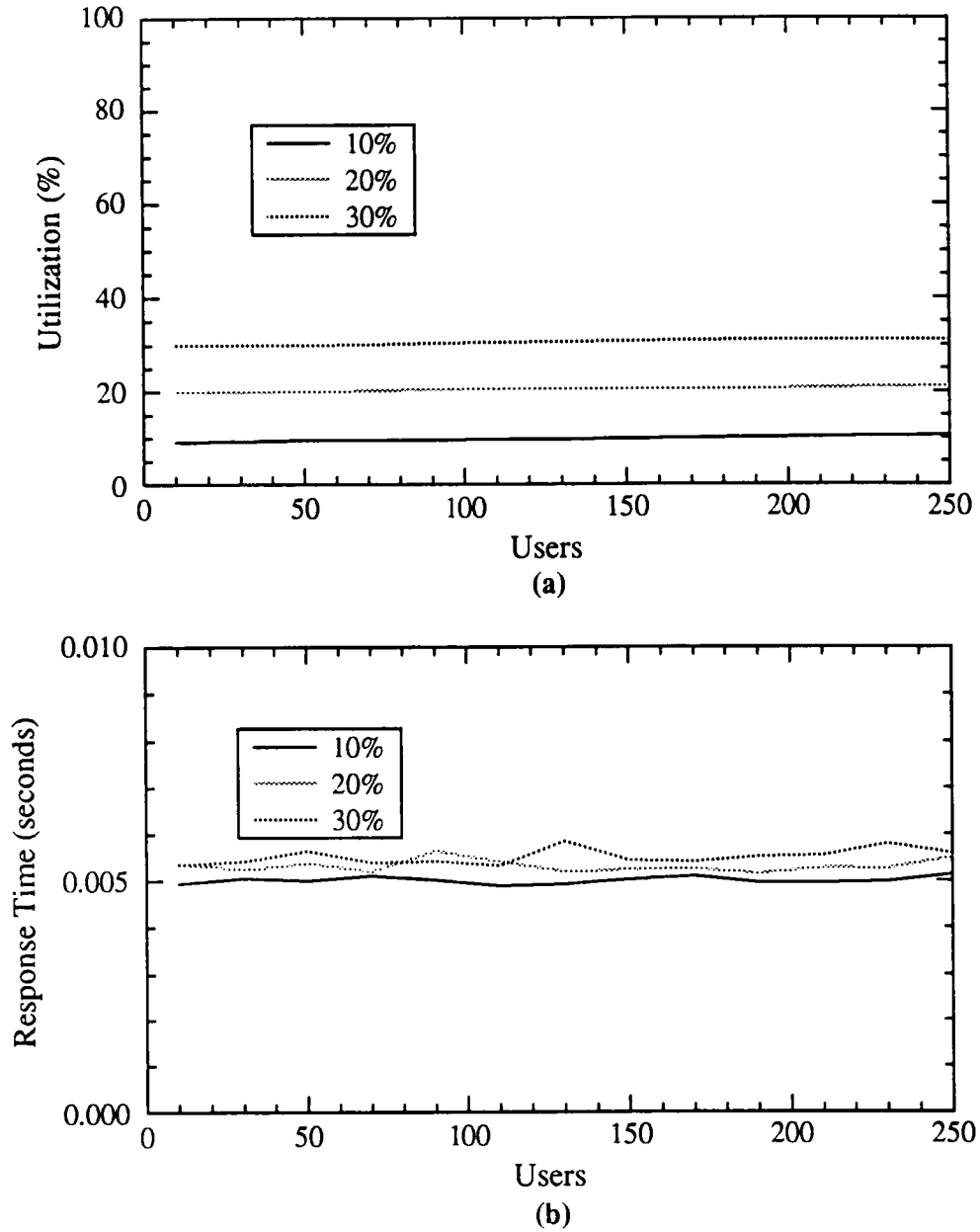


FIGURE 27 Text Output User Profile (Medium) with Background Utilizations of 10, 20, and 30 Percent

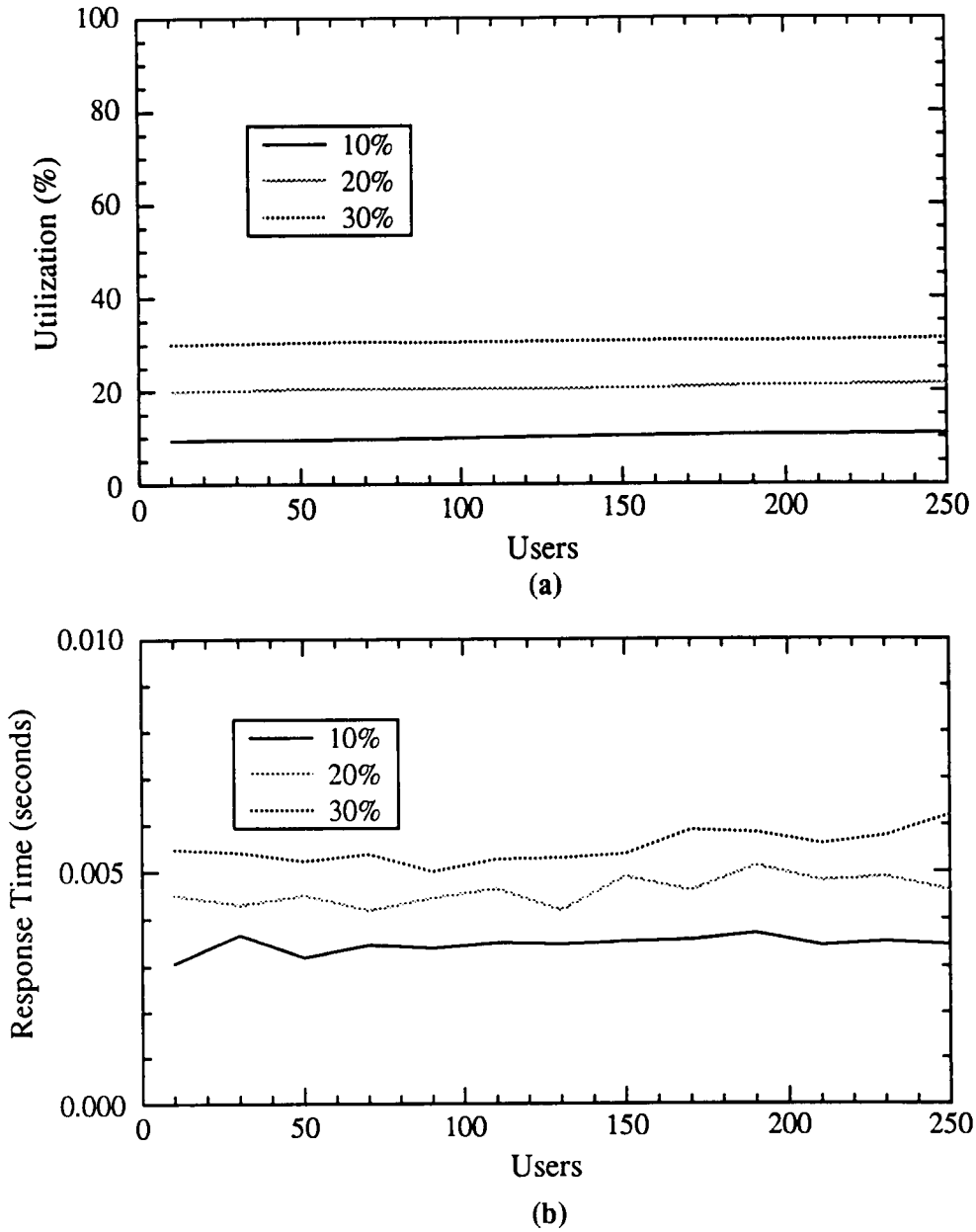


FIGURE 28 Graphics Input User Profile (Medium) with Background Utilizations of 10, 20, and 30 Percent

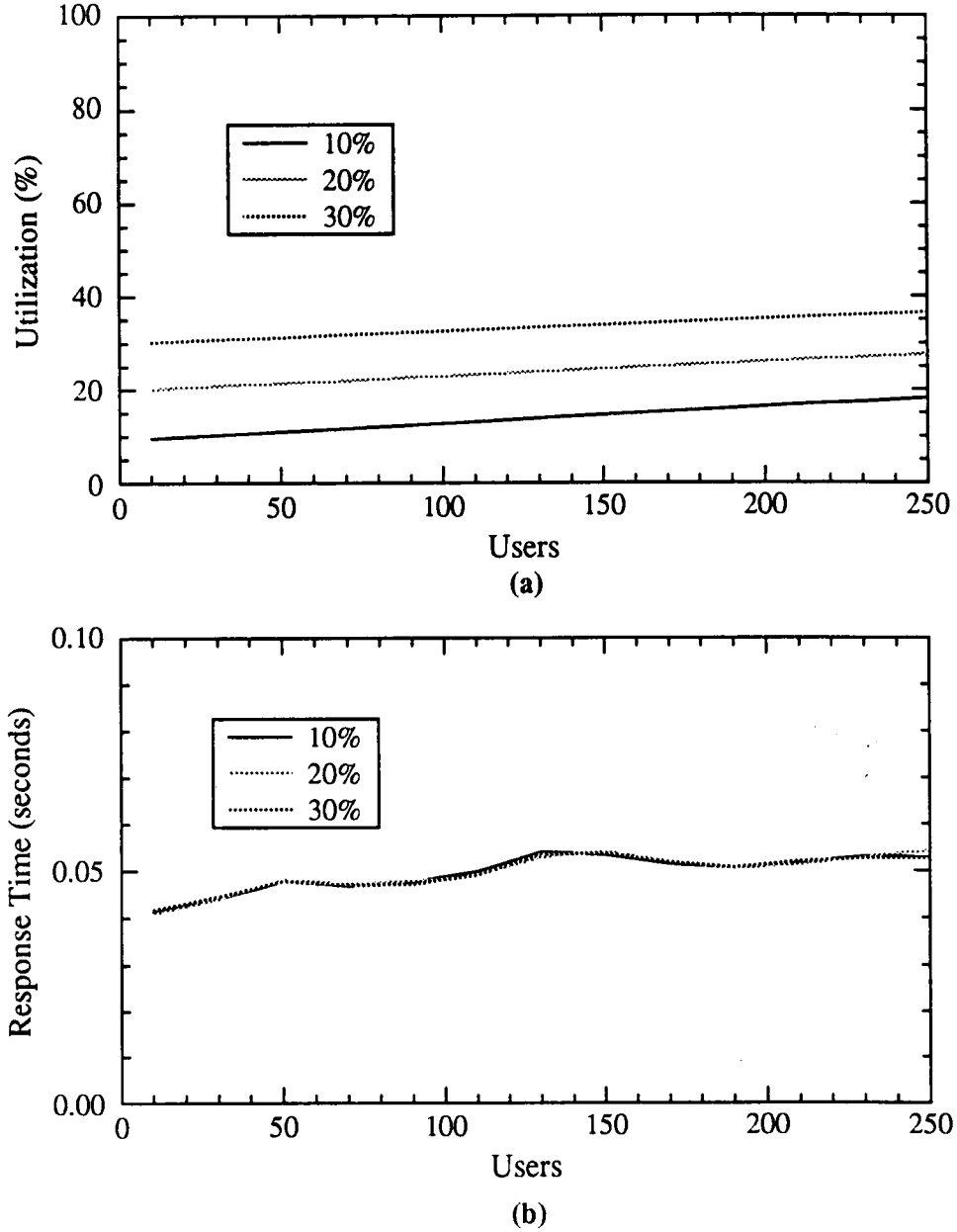


FIGURE 29 Graphics Output User Profile (Medium) with Background Utilizations of 10, 20, and 30 Percent

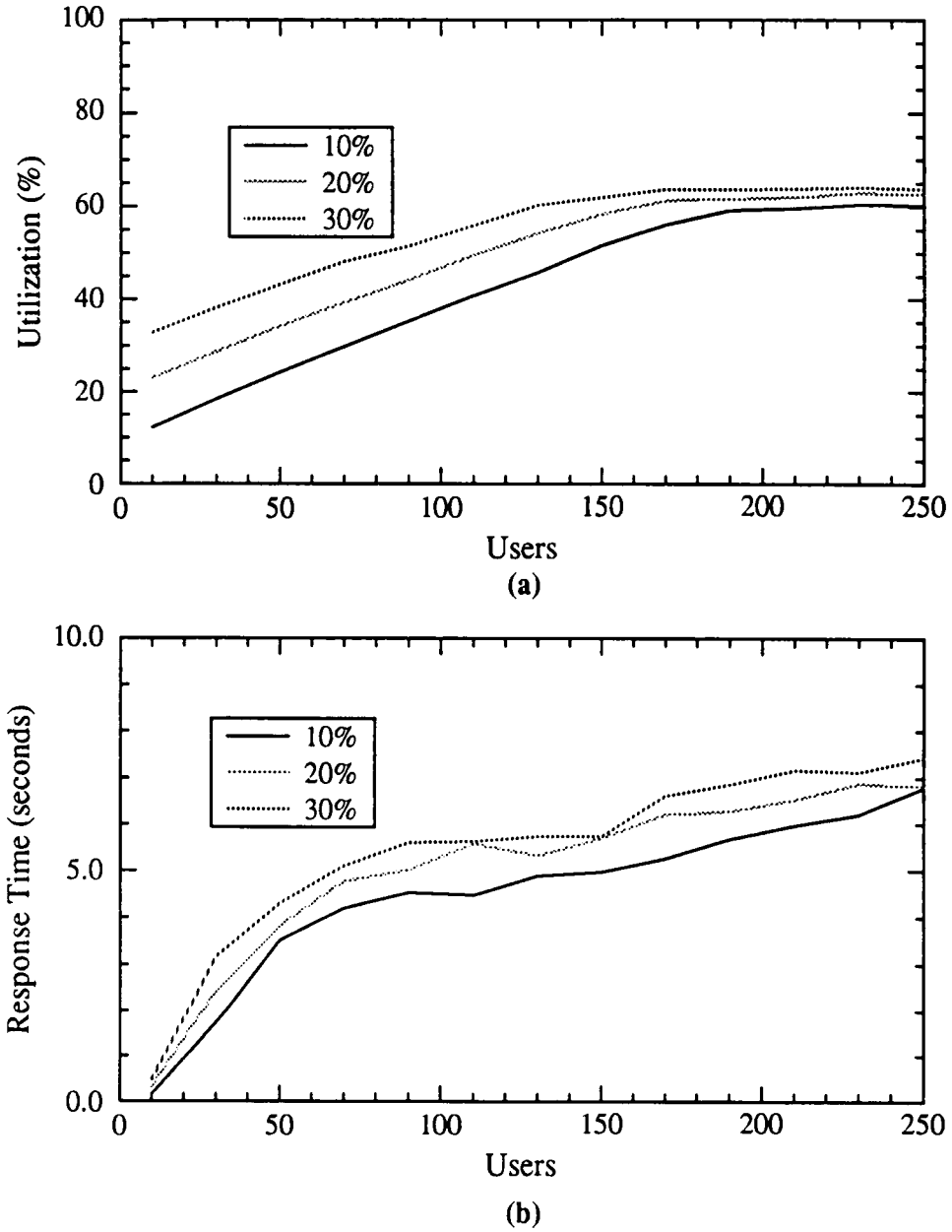


FIGURE 30 Mouse Tracking User Profile (Medium) with Background Utilizations of 10, 20, and 30 Percent

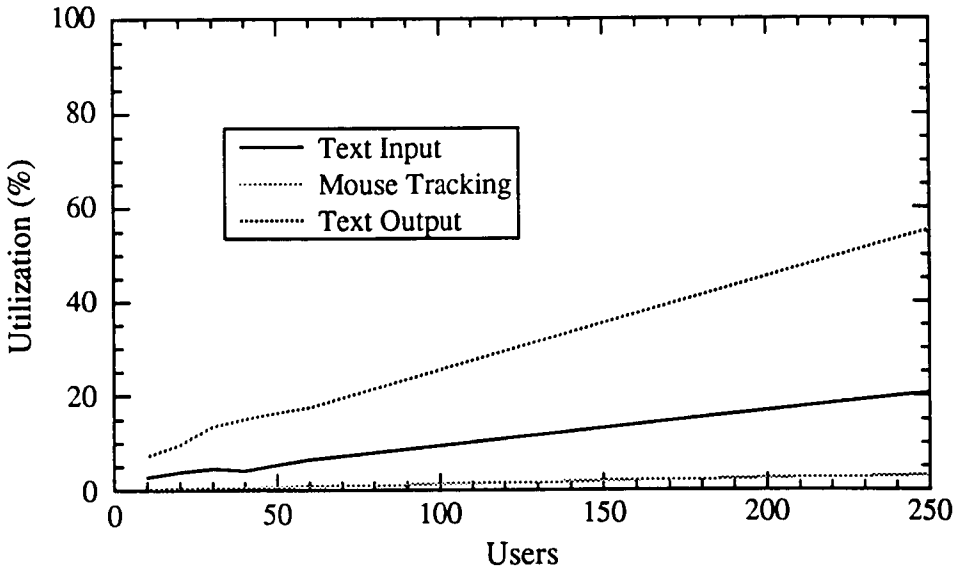


FIGURE 31 Actual Ethernet Data

Note that the data beyond 60 users is a linear projection and not actual data taken from an ethernet.

Vita

Timothy William Hickerson was born in Linden, Tennessee, on November 17, 1959. He attended public schools in Hohenwald, Tennessee, and graduated from Lewis County High School as class Valedictorian in May, 1978. The following fall, he entered Tennessee Technological University where he graduated with honors receiving the Bachelor of Science degree in Electrical Engineering, in March, 1983. Since June 1983 he has been employed by Martin Marietta Energy Systems, at Oak Ridge, Tennessee, as a design engineer in the Computer Applications Department at the Y-12 Plant.

In the fall of 1986 he entered the Graduate School at the University of Tennessee, Knoxville. He completed the requirements for the Master of Science degree with a major in Computer Science, in December, 1992.

He is married to the former Cindy Sue Todd of Geff, Illinois. He and his wife have a son, Jonathan William, and a daughter, Jessica Nicole.