Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

PERFORMANCE EVALUATION AND PREDICTION FOR LARGE HETEROGENEOUS DISTRIBUTED SYSTEMS

Joseph Betser

November 1988
CSD-880091

UNIVERSITY OF CALIFORNIA

Los Angeles

Performance Evaluation and Prediction for Large Heterogeneous Distributed Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Joseph Betser

1988

The dissertation of Joseph Betser is approved.

Bennet P. Lientz

Michael K Stenstrom

Michael K. Stenstrom

Richard R. Muntz

Jack W. Carlyle

Jack W. Carlyle, Committee Co-Chair

Walter J Karplus

Walter J. Karplus, Committee Co-Chair

University of California, Los Angeles

1988

to GISELLE

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to extend my gratitude to my dissertation committee members, Dr. Bennet Lientz, Dr. Michael Stenstrom, Dr. Richard Muntz, and the Co-Chairmen Dr. Jack Carlyle and Dr. Walter Karplus. Dr. Carlyle provided continuous feedback and critique for my work, as well as academic advice that has been very useful. Dr. Karplus has always been sharp in applying his scientific and engineering experience as well as his human insight throughout the progress of this work. Both Co-Chairs provided continuous friendship and encouragement throughout my efforts. Dr. Karplus and Dr. Carlyle have been most instrumental in inspiring progress, momentum, and enthusiasm throughout our relationship.

I would also like to thank Dr. Stenstrom for his assistance in the capacity of the Academic Director of SEASnet. His feedback and cooperation are acknowledged, as well as his resonsibility for the development and smooth operation of the testbed system for this work.

This work is a result of my years of graduate study at the UCLA Computer Science Department. It is my pleasure to acknowledge the support I received during earlier years with the Locus research group. Dr. Jerry Popek's commitment and excitement, Dr. Mario Gerla's insightfulness, and Dr. Richard Muntz's critique and advice are gratefully acknowledged.

I would like to thank Dr. Dan Berry for his academic advice and lasting, sincere friendship throughout the years.

Special thanks are due to my co-worker, Alberto Avritzer. Beto's fine sense of humor and our creative exchanges have given my last year at UCLA some interesting memories.

This work was enabled through the generous support of the IBM Corporation, through the IBM Joint Study D850915, of which I am a senior member. I would also like to acknowledge previous financial support of the ARPA research contract MDA-903-82-C-0189.

My parents, Dr. Abraham and Zipora Betser, have instilled in me love, dedication, and intellectual curiosity that have nurtured my growth and ambitions. It is a tough, lengthy undertaking to be a parent. I look up to the example they set, and hope to carry on these values into the future.

To my dear wife, Giselle, I dedicate this work. Giselle has provided her endless love and affection. These rare qualities stimulate my drive and enhance my creativity. Very few men are as lucky and as proud as I am to have such a unique partner to life.

VITA

1973-1975	System Programmer, Physics, TECHNION, Israel Inst. of Tech.
1976	B.S. (Honors), Aero. Eng., TECHNION, Israel Inst. of Tech.
1976-1977	Systems Project Officer, Israel Navy.
1977-1981	Research and Development Engineering Officer, Israel Air Force.
1981-1982	UCLA School of Engineering Fellowship Recipient.
1981-1983	Post Graduate Research Engineer, UCLA Computer Science.
1982-1983	UCLA Hortense Fishbaugh Scholarship Recipient.
1982-1983	Teaching Associate, UCLA Computer Science.
1984	M.S. Computer Science, UCLA.
1984-1987	System Analyst, UCLA Computer Science.
1987-1988	Senior Performance Analyst, UCLA Computer Science.

PUBLICATIONS

1. "Configuration Synthesis for a Heterogeneous Backbone Cluster and a PC-Interface Network", Computer Science Department, UCLA, CSD-880074, September 1988, to appear IEEE INFOCOM '89, Ottawa, Canada, 24-27 April 1989, (with Avritzer, Carlyle, Karplus).

2. "Performance Modeling and Analysis for a Large Heterogeneous Distributed System: UCLA-SEASnet", Computer Science Department, UCLA, CSD-880073, 1988, to be presented at the ACM Computer Science Conference

1989, Louisville, Kentucky, 21-23 February, 1989 (with Avritzer, Carlyle, Karplus).

3. "Potential for Load Sharing within Locus Server Clusters" Computer Science Department, UCLA, 1988 (with Avritzer, Carlyle, Karplus).

4. "Configuration Synthesis/Specification and Load Balancing for a Distributed TCF Cluster Environment", UCLA Computer Science Department, 1988 (with Carlyle, Karplus).

5. "Locus and SEASnet - Performance Analysis and Progress Report", Computer Science Department, UCLA, 1987 (with Lai, Carlyle, Karplus).

6. "A Dual Priority MVA Model for a Large Distributed System: LOCUS", Proceedings of PERFORMANCE "84 - the 10th International Symposium on Computer Performance, Paris France, 19-21 December, 1984 (with Gerla, Popek).

7. "Performance Modeling and Enhancement within the LOCUS Distributed System", M.S. Thesis in Computer Science, University of California, Los Angeles (UCLA), 1984.

8. "Page Level Update Propagation in LOCUS - Design and Implementation", UCLA Computer Science Department, 1982.

9. "Aircraft Performance Prediction/Evaluation, and Feasibility Studies", numerous reports, Israel Air Force 1977-1981.

10. "A Three Dimensional Model of Turbulent/Viscous Flow for Jet Engine Turbine Flow Applications", Technion 1981.

11. "The Design and Operational Analysis of a Camera Controlled Store Separation System for Airborne Operation" Technion / Israel Aircraft Industries 1976.

ABSTRACT OF THE DISSERTATION

Performance Evaluation and Prediction
for Large Heterogeneous Distributed Systems

by

Joseph Betser

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1988

Professor Jack W. Carlyle, Co-Chair

Professor Walter J. Karplus, Co-Chair

Heterogeneity is a trend setting phenomenon in today's world of networked computing. Combined with distributed computing, and large scale systems, such heterogeneous systems exhibit overwhelming complexity. This dissertation addresses performance issues, such as measurement, analysis, modeling, simulation, and configuration synthesis for such systems.

Our study is experimentally validated on a novel testbed system, which consists of heterogeneous hardware units and operating system types, and provides transparent access to hundreds of users in an academic instructional environment. The main emphasis of this case study work is to arrive at a first order model, that renders successful results, by observing key behavior parameters of the systems under consideration. Such a model and methodology carry general applicability to other heterogeneous systems.

A detailed study was conducted to define the user model of such a system, and arrive at fundamental building blocks for a variety of user communities and load patterns. We describe these results as a three-class user model structure.

Numerous sets of performance measurements were designed and executed to characterize the heterogeneous distributed system under investigation. These measurements are required as input for our modeling process. They also carry a substantial value for future modeling purposes.

A modeling effort was carried out, leading to the derivation of a queueing network model for such systems. This effort was particularly challenging due to the limited availability of low-level system parameters. Discrete event simulation was utilized to arrive at performance results, and we show good fit of modeling results to measured performance.

Configuration synthesis algorithms were further developed and demonstrated for heterogeneous systems that we study. Given quantities are assumed to be the expected user workload in a three-class characterization, hardware costs, and delay constraints. We develop an optimization algorithm that uses performance design goals to arrive at the system configurations of choice.

Finally, resource sharing issues are addressed by studying session assignment strategies, and the resulting global performance and optimization characteristics. This work also provides a basis for further research on load balancing ongoing at UCLA.

CHAPTER 1

Introduction

In recent years, Computer Science has been transforming from interactive computing towards distributed computing, and machine boundaries have become less and less apparent. While transparency among homogeneous set of machines has been studied, much less is known about heterogeneous architectures. In particular, many performance issues are of immediate interest, since heterogeneous systems have inherent communication overheads by virtue of the differences among system components. Since there is currently a clear trend of connecting different hardware/software systems on common close-coupled networks, a performance study of such systems is highly desirable.

1.1 Large Heterogeneous Systems

Currently, many large heterogeneous systems are built around Local Area Networks (LANs), and consist of user terminals, workstations, and backbone machines that are more capable than the workstations in terms of their processing power and storage capacity. The workstations may be of different architectures, and likewise the backbone servers may be heterogeneous. In addition, different operating systems could be running on the various machines. Moreover, some network-wide resources are made available to all users. All of the above demands make it very challenging to achieve a high performance solution for the design task of large heterogeneous distributed systems.

1.2 The Performance Problem

Applying performance analysis methods to such systems has not been extensively done. A few systems have been built using design experience and intuitions acquired from less complex systems. It would be safe to assume that the resulting design might be less than optimal. In fact, what performance parameter is most important to the user ?

We argue that the most important performance parameter to users is system speed, as **they** feel it. This, in essence, is what is meant by **system response time**. Response time has several levels of granularity. At the finest grain one would consider character echo time, at an intermediate level, interactive script execution time, and at the coarse-grained level, time to execute computationally intensive tasks (e.g. program compilations, word processing, numerical analysis/simulation). The latter is not considered interactive, as the user is not specifically waiting for system response and might be doing an interactive task while the non-interactive task might complete in the background (batch mode). However, the user would like non-interactive jobs to complete as quickly as possible, to enhance productivity; thus one may consider **finishing time** for a suite of tasks. As a general criterion users might desire that character echo should complete within a fraction of a second, and interactive tasks should complete within seconds. Expectations for non-interactive tasks would be considerably more variable.

1.3 Analysis of Large Heterogeneous Systems

This is currently a significant task, as the complexity we are considering becomes greater than that dealt with by established techniques that we are familiar with. We are looking at a system that is heterogeneous in both hardware and software aspects. We are dealing with personal machines as well as faster backbone machines. In many cases the operating systems are different as well. The systems are distributed, using local area networks to connect their various components.

There are some other major modeling issues in addition to system speed. These are *reliability/availability* and system behavior in the face of *Scaling*. Distributed systems have become a reality due to the desire for reliability/availability coinciding with decreasing microprocessor prices.

As the number of nodes in a system increases, performance figures across the system vary. These fundamental variations tell us about the potential of the system to scale up. We can anticipate advantages and drawbacks from such a scaling operation. Load patterns and their effect on scaling will give us general insights about distributed system scaling response.

How is the load being managed under variable conditions ? How can we achieve graceful degradation ? How can a catastrophic failure be avoided ? A good strategy would be to strive for good resource sharing policies. In fact, in the synthesis part of the work we demonstrate session allocation policies that successfully accomplish this goal.

Some of these questions were posed as the LOCUS operating system was conceived [PopeWalk85]. Our present research, while not requiring a Locus-like environment, has benefited from the presence at UCLA of a significant distributed system based in part on Locus and its descendant AIX/370 [IBM88]. With this as an experiment testbed, we have been able to carry out a research program leading to a verifiable methodology for modeling large heterogeneous distributed systems.

1.4 Our Modeling Effort

Our primary testbed system consisted of SEASnet - the School of Engineering and Applied Sciences Network. A smaller, secondary research network was also used. SEASnet is described in detail in Chapter 3; in brief, it is currently among the larger production systems in an academic environment demonstrating the melding of heterogeneous hardware and operating systems. We have studied the user community behavior as well as measured system parameters; this enabled the construction of a model for such heterogeneous distributed systems.

The modeling effort was especially challenging, since the our working assumption was that the data available to us would be incomplete, in the sense that fine grained information would be lacking. Accordingly, we were able to obtain useful global measurements for system parameters, but access to internal parameters was limited. This led us to the decision to pursue a *first-order model*, that demonstrates good validity for system performance prediction. This required careful assessment of available data, measurements, and design strategies. We sought to determine the first order behavior of the system sufficient to derive a queueing network model that usefully characterizes the actual system.

1.5 Synthesis of Large Heterogeneous Distributed Systems

Once our analysis capability is demonstrated, the challenge is to advance to synthesis capability. Using the analysis model as a building block, one seeks to construct a synthesis algorithm that arrives at optimal configurations, given user requirements, available budget, and design constraints. The key idea in such synthesis procedures is to iterate over several choices, checking how the objective function behaves. Design parameters are changed, until the objective function achieves an optimum value. We note that the evaluation of the objective function results in from the model and its analysis/simulation.

In this dissertation, our ultimate accomplished goal was to achieve the synthesis capability. Since synthesis depends on a validated model, the phases of our research have reinforced each other, as successful configuration synthesis reaffirms that our modeling effort was fruitful.

1.6 Synopsis of the Dissertation

In Chapters 2 through 9 we present the body of the dissertation. We now briefly describe the contents of these chapters.

Chapter 2 describes academic computing systems and their characteristics. Chapter 3 gives the overview of SEASnet necessary for the reference in the rest of the work. In Chapter 4 we develop our user model with SEASnet as testbed. Chapter 5 contains the core of our measurement work, with the modeling goal in mind. In Chapter 6 we describe our modeling effort, leading to a queueing network model, again with SEASnet as our testbed. In Chapter 7 we develop a configuration synthesis algorithm for large heterogeneous distributed systems; this is a cost performance optimization in which sessions are assigned with a balanced load constraint. Discussions of research in several disciplines related to our work have been collected for presentation in Chapter 8. In Chapter 9 we summarize our contributions, elaborate on their significance, and draw some directions for future research.

It should be emphasized that an effort has been made to create chapters which are largely self contained. For that reason there is a limited degree of redundancy among several chapters, so that the reader may examine each chapter individually. Moreover, we felt that some detailed material was beyond the scope of individual chapters, and we placed this material in the Appendices, for completeness.

CHAPTER 2

Academic Computing Systems

Perhaps the most distinctive characteristic of academia is the diversity of issues that are addressed within an academic institution. In fact, the academic environment is a microcosm of the real world. As prospective professionals are brought up in the academic process, they confront some representative experiences that emulate the real life experience. The major difference is the rate of systematic learning. New theories, tools, and techniques are introduced at an accelerated rate. This rate of knowledge acquisition comprises the production performance in an academic environment.

The introduction of computing systems into academia has often been at the forefront of computer technology. Leading academic institutions with good research programs have traditionally attracted many vendors to supply the schools with their latest product; this has contributed to heterogeneity. SEASnet[1] is an example of such a relationship involving IBM and other vendors with the UCLA School of Engineering and Applied Science (SEAS). In the following chapter we shall elaborate on the particular qualities of the academic computing environment, since SEASnet served as our main research testbed.

[1] SEASNET - School of Engineering and Applied Sciences Network

2.1 Local and Central Resources in Computing Paradigms

During the previous decade the predominant computing paradigm was the central mainframe computer, with interactive users time sharing it via user terminals. the cost of such systems was prohibitive for any entity smaller than a university campus, a major corporation, or a large government agency.

Since the advent of very affordable microcomputers, the paradigm of a central mainframe has been constantly updated. The proliferation of smaller processors into modest laboratories has been a fact for over a decade now. Moreover, personal computing is a fact of life for the last five years or so. Individuals have their own microprocessor based workstations on their desk as the standard paradigm. A private individual would most probably use his machine in a standalone mode. At times some sporadic communication across telephone lines might occur. However, major facilities operate substantially more intensive local communication media. Local Area Networks (LANs) are becoming popular, and an increasing number of different processors are gaining access capacity to LANs and other communication media. Ultimately, network wide resources can become available to any user.

This increased level of machine connectivity has made distributed computing feasible, but also led to heterogeneity in hardware and software systems. There exist a multitude of applications for personal computers and workstations. Many of these applications can run on the workstations in an autonomous fashion.

However, some services, such as file service, cycle service, and communication services are best handled by the central, larger nodes. The issues pertaining to the manner in which distributed resources are utilized are complex. The central resources are more powerful, but not as locally available as the personal node's resources. In the rest of this dissertation we shall call the central machines *Backbone Servers.* The personal machines will be called PCs or *Personal Workstations.*

2.2 Uniqueness of Academic Environments

The academic computing setting has some unique aspects in comparison to general facilities in the industrial or business worlds. The user community is very demanding in terms of software novelty and offered features. Recent releases are very much in demand and support for many applications is required. In addition, easy access to hardware and software resources is desirable, both in terms of time and location. Courseware is rapidly changing, and the computer support for it needs to accommodate for these enhancements at a competitive pace.

Heterogeneity is a fact of life in today's academic environment, because of the rich set of vendors interested in supporting academic applications and operations. Since the introduction of Local Area Networks (LANs) earlier in the decade, the co-existence of heterogeneous environments on the same network has been a challenge in academia.

Other criteria are emphasized in order to obtain the above features. We mention here the fact that life-death reliability is not required, and that the real time constraint is not significant in most academic computing systems. System stability is very desirable, but users have a higher degree of tolerance since they wish to use novel features, which might still be under development. System security is not cardinal in an environment where many public-domain programs are present, and easy accessibility is paramount. Clearly, business-like considerations (e.g., detailed record keeping, audit trails, profitability) are not so important in academia, as the main goal is to learn and extend human knowledge.

2.3 The Academic User Community and Computing Applications

Inherent diversity is the characteristic quality of this community. The number of applications is as staggering as the number of disciplines studied in higher education. Among the many disciplines we mention physics, architecture, medicine, film animation, linguistics, engineering, management, and library science. In addition, some of the users are teaching Faculty and students enrolled in courses, and others are Faculty and research staff pursuing research. Usually there is some interaction with administrative computing as well.

Depending on the discipline, we mention a few examples of the application diversity. The Psychologist will be using statistics to reduce study data, the Physicist would use numerical packages to obtain curve fittings, optimize parameters, and carry complex analyses of physical models. In general, many of the Engineering and so called "Scientific Computing" applications solve mathematical models that represent field problems and others. The Architecture faculty and students would use Computer Aided Design (CAD) programs to help create new architectural concepts. The Economist would use the computing facilities to run economic simulations, and perform various data reductions. The Biology professional would like to analyze experimental data and protein structures. The Medical applications call for patient scheduling, diagnosis, real time monitors, data analysis, and so on. The Library and Information sciences call for major on-line catalogs, searches, and retrievals. The Business school is concerned with the multitude of business applications, from spreadsheets to accounting, from financial planning to trade and production analyses. For a more elaborate description of academic applications, the reader is referred to [ACIS86], [ASEE87], [ACADCOMP88] and publications specific to the various disciplines.

We should note that in each and every discipline mentioned, there has been a significant trend for micro-computerization. A great majority of the applications mentioned can indeed run on a personal computer or workstation and alleviate load on the main computing resources on campus. This important trend is a major motivating issue in this dissertation.

Hence, the user community in the academic setting is highly diversified. In order to obtain meaningful results for a specific case study group, we have concentrated on teaching activity within the school of Engineering. Without loss of generality, we proceed in the following section with further details on the Engineering and Computer Science disciplines. Similar studies could be made in other disciplines.

2.4 Applications Specific to Engineering and Computer Science

As we focus on the areas closer to us, we observe that there is a multitude of various applications within the Engineering and Applied Sciences area. Circuit designers typically employ CADAM software for vlsi design. Nuclear engineers use simulation programs to determine nuclear reactor operation. Computer scientists use architecture, communication, languages, operating systems, methodology, and artificial intelligence applications among many others. Civil engineers use structural and water resources applications. Chemical engineers use chemical process control techniques. Aerospace Engineers use numerical aerodynamics, flutter and active control, and propulsion computations. There are many more applications in the general engineering area. However, they are all represented in the course material of the UCLA-SEAS, and some more detail will be provided in chapter 4 below, as we focus on the SEASnet user community.

2.5 Summary

In this chapter we have provided the necessary context regarding academic computing. We presented an elaboration on the important current trends towards personal computing, networking, distributed computing, heterogeneity, and central resource sharing.

We have illustrated the important characteristics of academic computing, and have emphasized the extreme diversity in discipline within the community. We concluded with a brief exposition of the Engineering and Computer Science disciplines, as a prelude to a more elaborate description of SEASnet and its user community in chapters 3 and 4 below.

CHAPTER 3

SEASnet - A Large Heterogeneous Distributed System

SEASnet is the major computing environment of the UCLA School of Engineering and Applied Science (SEAS). It has been in operation since 1985, and has a growing impact on the school instructional and research environment [Sten87]. SEASnet provides an alternative to resources traditionally found in the campus-wide office of Academic Computing (OAC). SEASnet consists of backbone server machines and client workstations. There is a substantial degree of heterogeneity involved, as will be detailed in the architecture subsection below. SEASnet is running the Locus operating system as its backbone operating system. An overview of Locus will be provided below as well. The challenges of modeling our heterogeneous and distributed system will be outlined towards the conclusion of this chapter.

3.1 SEASnet Educational and Operational Goals

SEASnet was provided to the school in order to enhance its computing resources, especially for instruction. This resulted in greater autonomy and independence of centralized campus resources. The increased availability was conducive to the promotion of computer use in the courses taught at the school. In fact, the utilization of personal computers as the user front end is a successful concept. Many users own PCs, or otherwise have access to a PC. In addition, a vastly growing number of DOS applications are in existence. Hence, class work and research work are done on standard tools which are easily accessible. The net result is that hands-on computing experience is achieved throughout the school, as the entire faculty and student body are so easily exposed to computing resources.

SEASnet architecture will be described in the next section (3.2). We try to keep this section (3.1) independent of specific architectural detail. If, however, the reader so desires, cross reference to section 3.2 might be appropriate.

3.1.1 The SEASnet User Community

The major emphasis of SEASnet operation is class instruction with an additional emphasis on research support. Hence there has been a major thrust of "computerizing" an increasing number of engineering courses [SEAS85,SEAS86,-SEAS87]. Some courses were transferred to SEASnet from the central campus facility. Additionally, courses which were based on manual work were upgraded to include computer work. This increased the student productivity, as computations previously infeasible were easily completed.

Most of the user community consists of upper division students. Additional users are graduate students, faculty and lower division students. About half of the user community has some familiarity with personal computing. The inexperienced users obtain their introductory experience on DOS and PCs during their first class on SEASnet.

We make a note at this point, that direct interactive access to Locus is provided to some research users; this Unix-flavor access is provided in order to support the vast amount of software available for the Unix operating system. Classroom users, however, see this backbone environment indirectly, through an operating system bridge, as will be explained in section 3.2.

3.1.2 SEASnet Operation and Applications

The applications run on SEASnet are as diversified as Engineering courses are. They range from computational fluid dynamics to artificial intelligence, from water resources control to vlsi design, from data structures to signal processing, from computer graphics to chemical plant simulation, and the list goes on and on.

Most of the applications used in class material are DOS programs. SEASnet provides the application software packages, and students use the packages to solve the various problem sets assigned to them. A higher degree of student involvement might be a requirement to provide procedures/routines that will be merged with larger software packages. Classes which are closer to computer science might require the students to generate major software modules as programming assignments. Regardless of the degree of programming required of the user, we note that the same DOS application environment is invoked, in which the user operates to accomplish a specific task.

We should note that most SEASnet activity originates from three workstation classrooms, containing a total of over 70 PCs. All classwork is done through these PCs. In addition, individual faculty members have PCs in their offices. Some interactive ascii terminals exist in the school. These terminals provide interactive Unix/Locus access for research users. A newer classroom contains IBM RT machines.

A substantial elaboration on the behavior of the user community, and the manner in which SEASnet resources are utilized, will be given in chapter 4. The user characteristics will be detailed, with respect to application usage, communication demands, and file service requirements.

3.2 SEASnet Architecture

SEASnet ia a heterogeneous distributed system, serving the UCLA School of Engineering and Applied Science. Figure 3.1 below gives a partial schematic description of the SEASnet hardware configuration. Figure 3.2 describes the conceptual architecture of SEASnet, with our particular subset of interest circled at the bottom. We note that SEASnet consists of backbone machines running Locus, as well as client workstations of the PC-DOS variety. Hence, SEASnet is heterogeneous in the way of hardware, as its backbone configuration includes different types of processors, in addition to the drastic differences among servers and client machines. Additionally, SEASnet consists of heterogeneous operating

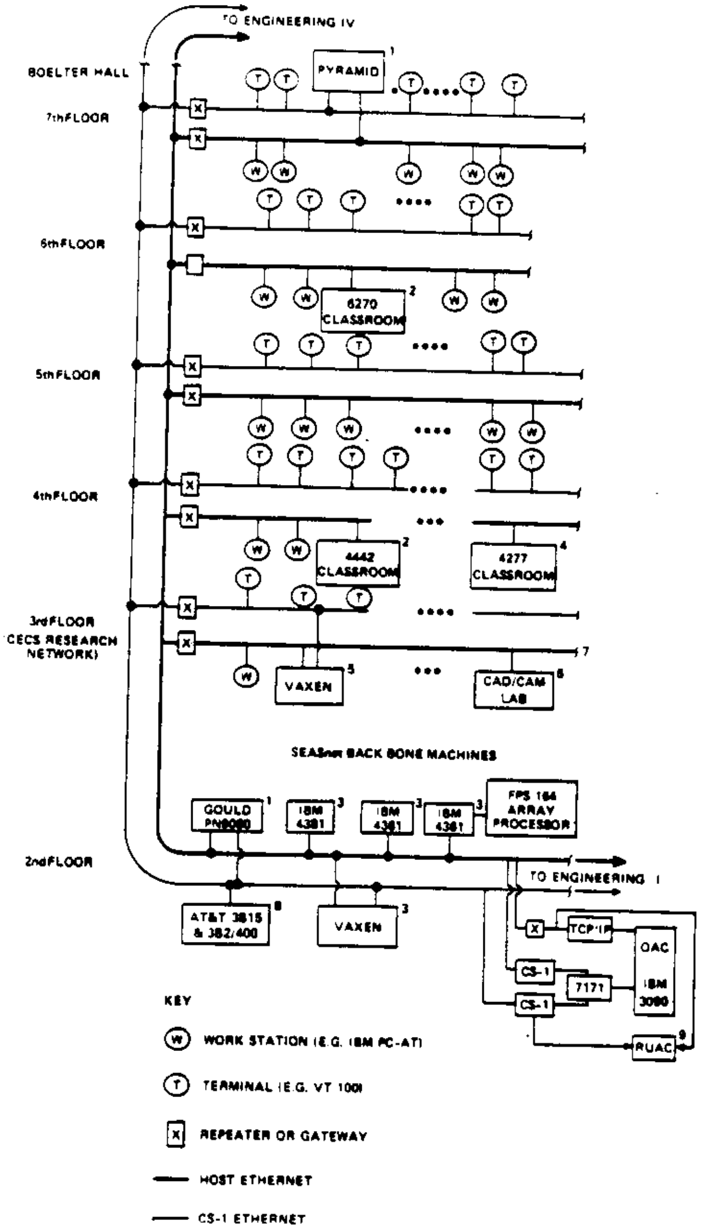


Figure 3.1: UCLA SEASnet Physical Layout within SEAS

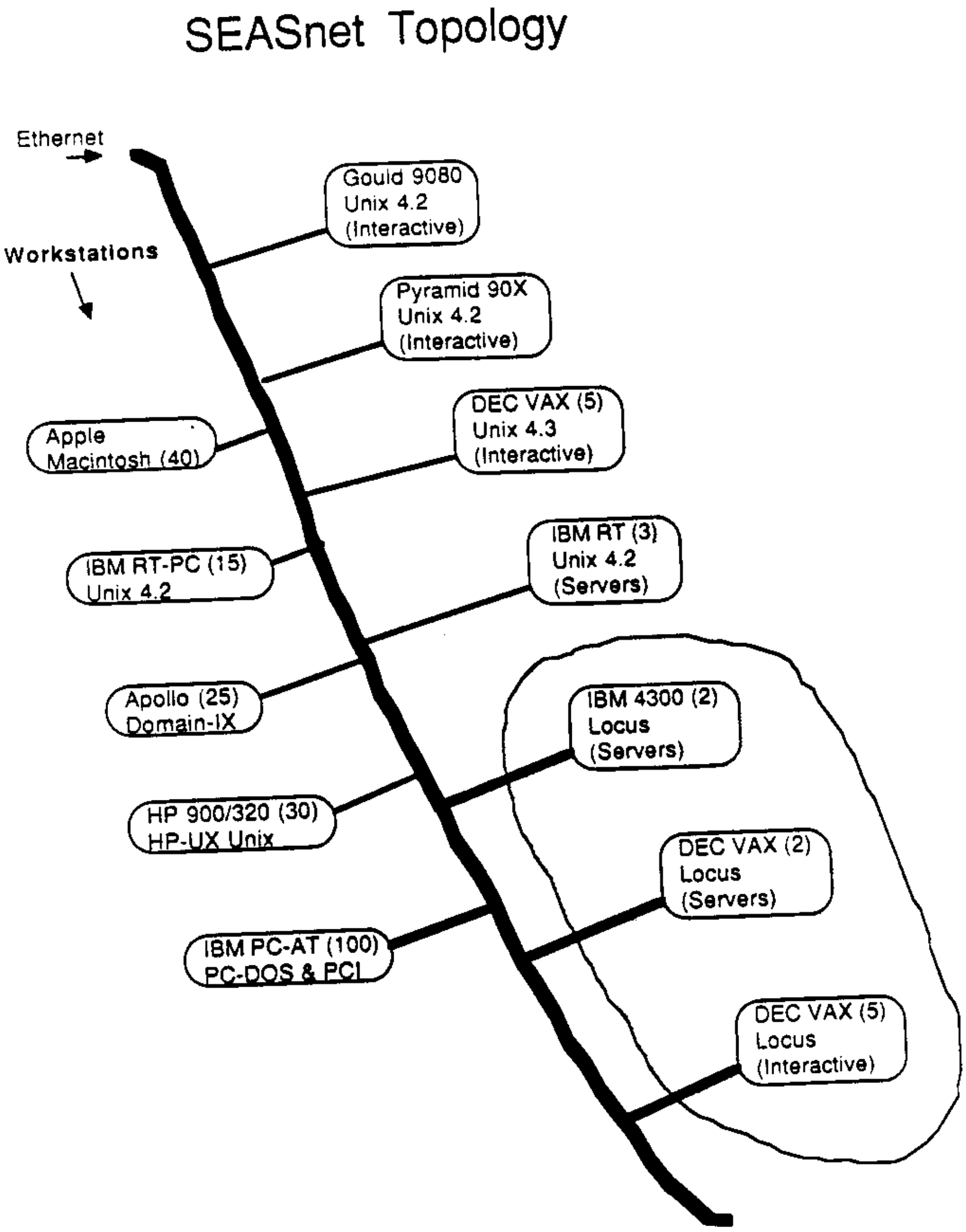

SEASnet Topology
Ethernet
Workstations
Gould 9080 Unix 4.2 (Interactive)
Pyramid 90X Unix 4.2 (Interactive)
Apple Macintosh (40)
DEC VAX (5) Unix 4.3 (Interactive)
IBM RT-PC (15) Unix 4.2
IBM RT (3) Unix 4.2 (Servers)
Apollo (25) Domain-IX
IBM 4300 (2) Locus (Servers)
HP 900/320 (30) HP-UX Unix
DEC VAX (2) Locus (Servers)
IBM PC-AT (100) PC-DOS & PCI
DEC VAX (5) Locus (Interactive)

Figure 3.2: UCLA SEASnet Conceptual Architecture

systems. The servers run Locus, and the clients run DOS. These completely different operating systems communicate with each other via an operating system bridge, called PC-Interface. The entire SEASnet facility is interconnected by a 10 Mbit/sec Ethernet. We shall describe the SEASnet architecture in further detail in the following sections.

3.2.1 Backbone Servers

SEASnet currently consists of three different types of backbone servers running Locus. These servers are IBM4381, IBM4361, and DEC VAX750 machines. These machines have a distinctly different architecture and performance capacities. There is a major difference between the IBM and the VAX architectures, as Locus runs on VAX as a native operating system, and as a VM application on the IBM architectures. We will see in forthcoming measurements and analyses that this has a noticeable effect on performance characteristics.

We note that memory capacity on all machines is in the 8-16 MByte range. As far as mass storage is concerned, the 4381 and 4361 are using 3380 and 3370 disk drives respectively. The 750s use Eagles for disk storage. Further technical detail is presented in Appendix A. The interested reader is encouraged to refer to it. Forthcoming upgrades to SEASnet will substantially increase backbone capacity while retaining the overall architectural philosophy.

3.2.2 Client Work Stations

The workstations on SEASnet are primarily PC-ATs and compatible machines, such as the ATT6300. These machines run the popular DOS operating system, and are connected via an Ethernet to the backbone machines. These machines have 1MByte of memory and 20MB of hard disk storage. Some applications run directly off the hard disk. These have to do with very frequently used applications, such as editors, some compilers (turbo-Pascal), graphics, and others. However most of the different applications are located on the backbone machines, and are used through the Ethernet via PC-Interface. The communication issues presented by this workstation-server relationship are central to this dissertation.

3.2.3 PC-Interface

PC-Interface (PCI) is an operating system bridge connecting Unix flavor machines to DOS machines [PCI86]. PCI allows the DOS user to transparently access files on the backbone Unix machine, as if they were on a local drive. The user views remote files as files on the so called "network drive" (drive E). Files from drive E are used as if they were stored on any of the local A,B,C,D drives.

In order to function, PCI performs a significant amount of work under the covers. It has a PCI module residing on the PC, which traps references to drive E, and creates the necessary network requests to the Unix host. At the Unix host end, these requests are handled by the DOS-server. The requests are transformed into Unix I/O operations, and the resulting data is shipped back over the network to the PC, where the PCI module translates the data to DOS format and returns it to the PC as if it came from the local drive. All this operation is done *transparently*, under the covers, without the user being aware of any detail of the operation.

The notion of transparency is a key asset of the Locus operating system. The user cares little about the actual location of resources, and the system locates and accesses resources in an automatic fashion. This notion carries special importance over heterogeneous boundaries, as users are even less likely to be familiar with remote environments to get their work done.

3.3 The Locus Distributed Operating System

This section will provide the reader with some necessary context for the purpose of this dissertation. More elaborate presentations can be found in [Pope.etal81], [Betser84], and [PopeWalk85].

3.3.1 Heterogeneity

The primary strength of Locus is its capacity to *transparently* operate among several architectures and software environments. For example, in the VAX case, Locus runs directly on the hardware architecture as a native operating system. In fact, Locus was initially developed for the VAX, as a concept of distributing Unix among several machines. However, since networking introduces many types of architectures on the same network, Locus was soon faced with different architectures, which were not readily available to accept a Unix flavor operating system.

One such architecture was the IBM370 architecture. IBM370 has its own Control Program (CP) that provides most of the lower level control functions. Incorporating Locus into such an architecture was a significant task, as there was no natural fit. Data/instruction formats were different, as well as hardware design, I/O channels, and communication devices. We shall see that an intermediate operating system layer was used temporarily (SSS), and was ultimately eliminated. Locus now runs directly on CP as a VM process. In fact, Locus was announced by IBM [IBM88], to be released as AIX/370 with TCF (Transparent Computing Facility) on a variety of hardware.

It should be noted that Locus was ported to other architectures such as 80286 (PC/AT) and 80386 (PS/2 Model 80) architectures. This demonstrates the rich set of architectures Locus supports. The testbed machines used in experimental work for this dissertation were mostly the IBM370 and DEC-VAX architectures, as well as PC/AT.

3.3.2 Transparency

Network Transparency is a key fundamental feature of Locus. This design goal states that distributed resources should be accessible to any user, as if the network is invisible (transparent), creating the perception that remote resources were local. This simplistic goal is extremely difficult to accomplish, as all network communication, remote heterogeneous environments, and foreign protocols are hidden from the user. The philosophy here is that the user need only be familiar with the local environment. It is the operating system's function to translate and interact with remote resources *under the covers.*

An example of transparency is *name transparency.* There is a globally unique namespace for the entire filesystem. Mappings are made internally to determine the location of a particular file within the Locus file system. Another important aspect is *performance transparency.* one would like to obtain high performance for remote operation, such that they would be comparable to local operations. Hence the user would not experience different response for the remote vs local operations, and the network again would appear transparent. This is a very difficult task to achieve, as remote operations have excessive communication delays and other overheads associated with them.

This dissertation will address primarily performance aspects, as they are among the most interesting to study, and most crucial to the end user.

3.3.3 Distributed File System

The file system, especially its distributed naming catalog plays a central role in the system structure, both because file system activity typically predominates most operating systems performance, and also because the generalized file name service provided is used by so many other parts of the system. This name service is responsible for the globally unique Unix-like tree structure name space. Hence, name transparency, location transparency, and performance transparency are closely related to the naming service. In fact, transparent access to other system resources, such as devices, is enabled by the transparent name service. Other topics such as synchronization, network partition and reconfiguration, basic file operations, and availability and reliability issues are described elsewhere in detail. We mention here only the parts that pertain directly to this dissertation.

3.3.3.1 File Replication

Locus provides automatic file replication as a feature supporting distributed computation. Since data is a major resource without which computation cannot proceed, it is of utmost importance to have programs and software as available as possible. File replication makes as much data as possible locally available. This is accomplished by replicating data of high access rate at local nodes. This saves remote communication overheads that might be incurred had the files been remote.

We should note that there is a substantial problem in maintaining a replicated file system in a consistent state. This is true particularly when the modification rate of individual copies is substantial. The update propagation issues of keeping consistency within a replicated file system were addressed in detail in [Betser84]. It was shown that both high modification ratios and high replication factors contribute to a substantial system overhead.

Hence, it was consequently decided to employ replication for heavily read system files, which are infrequently updated. Therefore, primary site replication was chosen. In this choice, there is a master site controlling updates for a particular file system, and all updates are cleared through that site. This is a good compromise of functionality and performance, since users access system files very heavily, but need only read or execute privileges. System files are modified by system administrators, a process invisible to the user.

3.3.4 Process Migration

Locus supports remote process execution and process migration. Remote process execution has a transaction flavor, as a self-contained task is shipped to a remote site for execution. The communication overhead is limited to data transfers associated with process initiation/conclusion. The costs of migrating running process are far greater than a-priori remote tasking. A running process has state information stored in the process image. This information pertaining to open files, executing programs, signals, etc needs to be shipped in completion to the remote site. The remote site needs to expand that information and create a similar process. Finally, the remote site can resume execution of the process remotely. In both cases, additional ongoing communication costs apply whenever a reference is made to resources remote to the execution site.

There is very little knowledge of the costs associated with these remote tasking operations. There exist some effort to understand potential load balancing costs and gains [DeSoGerl84,87], [TanTowWol88], [EagLazZah88]. We can note at this point that it is clear that a priori process assignment carries more immediate potential, as costs are smaller, and control will be easier. Real time process migration for load balancing is far more difficult to control, and will carry more expensive overhead price tags.

3.4 The Challenge of Analyzing SEASnet

SEASnet is a marriage of a large heterogeneous distributed system and a large academic application. Both issues are complex on their own right, and make the analysis of SEASnet an extremely challenging undertaking.

3.4.1 System Complexity

The architecture of SEASnet is unique. There are very few systems of similar flavor in academic installations. In fact, construction of heterogeneous systems of comparable size and degree of heterogeneity has become a trend setting goal for several campuses and research institutions. SEASnet consists of multiple vendor, multiple model backbone architecture, as well as multiple vendor multiple architecture workstations. In addition to the hardware heterogeneity, software systems present a significant degree of heterogeneity, as Locus oriented backbones are communicating with DOS flavor workstations. The DOS workstations are running a multitude of DOS applications. These applications generate network communication with the Locus backbones, in particular with their distributed file system.

The inter dependencies among system resources are very complex. Network communication, I/O bottlenecks, processor computation are contended for, as well as software modules, data, and other devices. Clearly, it is impossible to include all system parameters in a performance model for SEASnet. Hence, the *art* is to identify the most significant governing parameters, and take those into account. Hence a modeling effort might have several levels of granularity. We shall start with the highest level of system view.

3.4.2 Pace of System Enhancement and Expansion

SEASnet is an ever evolving system, as it consists of a novel distributed system which is undergoing continuous modification. As new enhancements are incorporated into SEASnet, new software releases are installed, and new hardware modules are integrated. In the way of hardware, new communication interfaces, new I/O channels, new workstations, and even new processors are received by SEASnet. In the way of software, new Locus releases are installed as they become available, and new system utilities and applications are acquired or created in house.

It is important to note that at each given point in time in the experimental side of our research, we were working on a particular system configuration. While that particular configuration might have had some implication on results obtained, we tried to emphasize the generality of our findings and their applicability to heterogeneous systems in general.

3.5 Summary

This chapter provided us with a description of the SEASnet environment. SEASnet is the major testbed for our performance work, which is detailed in the core of this dissertation. We described the educational goal of SEASnet as well as its architecture and its major design philosophies. The various heterogeneous properties of SEASnet were highlighted, as well as points of novelty, pertaining to both hardware and software issues. We have presented the difficulties in contemplating a performance analysis of SEASnet, due to its complexity. We also indicated that an ultimate driving goal for this research is our desire to be able to improve system wide resource utilization through better resource management. Load sharing, configuration synthesis, and other task and resource assignment issues would be the ultimate beneficiaries of better understanding of SEASnet like systems.

In the next chapter we shall study the user community of SEASnet. This will improve our understanding of the load model which is utilizing SEASnet on a daily basis. Later we employ this user model as input to a general system model, and obtain performance results. We proceed with the study of the SEASnet user model.

CHAPTER 4

The SEASnet User Model

In this Chapter we shall study the characteristics of the SEASnet user community. This study is an instance of a general methodology for load pattern determination for heterogeneous distributed systems. The work is a combination of parametric evaluation and experimental measurements.

The user community of SEASnet consists of mostly students taking courses, as well as Faculty and research staff conducting research[1]. Our goal is to determine the driving qualities of this user community, so we can construct a good model for user activity. It is important that we keep this model as simple as possible, and at the same time retain its first order behavior. We would also like to obtain a modular description that could be adapted to various user situations and future changes.

4.1 The SEASnet User Community

The vast majority of the SEASnet user activity is derived from project work that is done by the students taking classes in the School of Engineering. There are several thousand students using SEASnet during every academic year. Each

[1] Many research projects have their own labs and computing resources, so SEASnet resources are less taxed by research activity

of the students has quite a few homework assignments, and usually at least one large project.

4.1.1 Major Usage Modes

SEASnet is accessible to users via two main modes. Access is provided to interactive Unix applications, as well as to PC-Interface (PCI[2]). PCI is an operating system bridge that provides transparent file service between heterogeneous hardware servers running under UNIX/LOCUS and PCs running under DOS. Since our main interest is vested in *heterogeneity* of distributed systems, we chose to emphasize the PCI aspects of the service provided by SEASnet. It should be noted that the vast majority of SEASnet service at the time of this study (1987-1988) indeed consisted of PCI service. Figure 4.1 shows SEASnet's subset of interest to us, namely the backbone servers running LOCUS or AIX/370 and the PC classrooms which are served by SEASnet via PCI.

When PCI is used from a PC, the user logs on to the server, and has transparent access to files stored on the so called network drive (E:). Whenever remote files are accessed, the local and remote PCI servers communicate, and transparently provide the requested service to the user. This is done as if the request was issued to a local resource, such as the local hard disk (C:), RAM disk (D:), or a floppy (A:,B:).

Our main interest is to determine how central resources are being taxed, as

[2] PCI is a trade mark of Locus Computing Corporation (LCC)

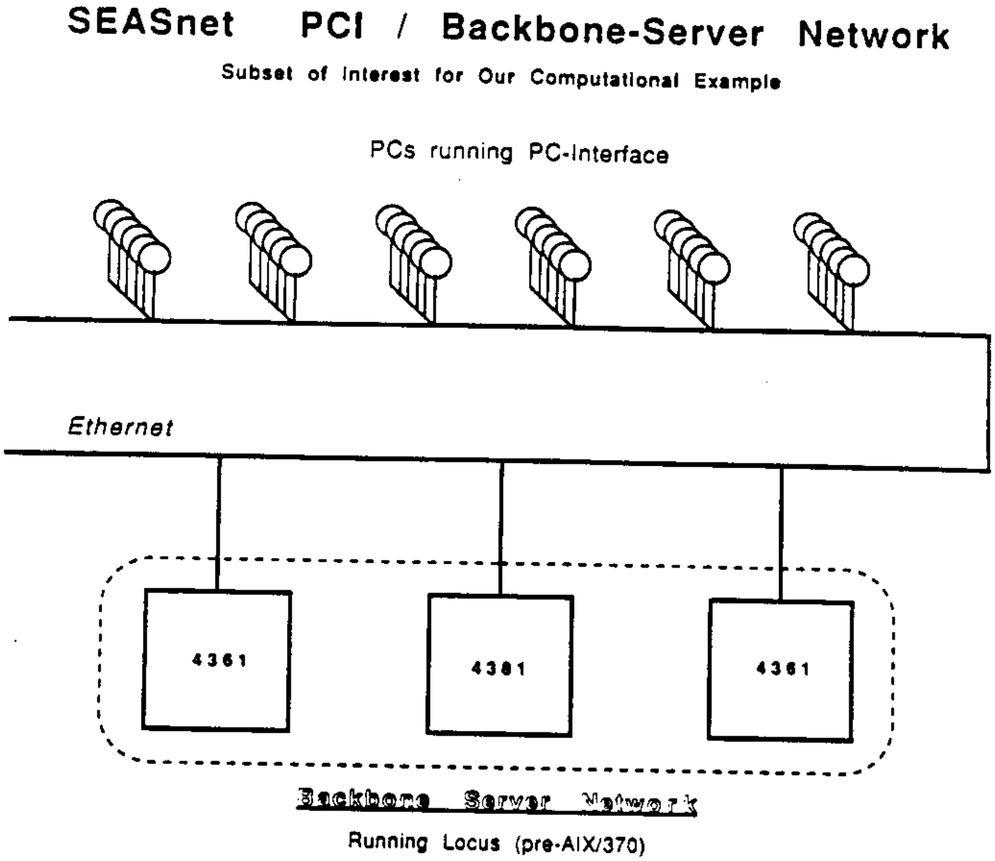

SEASnet PCI / Backbone-Server Network
Subset of Interest for Our Computational Example
PCs running PC-Interface
Ethernet
4361
4381
4361
Backbone Server Network
Running Locus (pre-AIX/370)

Figure 4.1: SEASNET - PCI / Backbone-Server Network

Figure 4.1: SEASNET - PCI / Backbone-Server Network

PCI is doing its tasks under the covers. In order to accomplish this task it is imperative that we study the demands that the user community is exerting on the PCI machines.

4.1.2 Applications Used

SEASnet is a mirror image of the engineering computing scene in the real life. The various classes taught are a representative sample of the various engineering specialties. Figure 4.2 shows a typical list of instructors that participated in SEASnet supported classes recently. It should be noted that a diversified course discipline is represented in this partial list. The associated Faculty are from Civil, Electrical, Mechanical, Aerospace Engineering, as well as from Computer Science. Additional pertinent information appears in Appendix B.

Therefore, there are many software packages which are licensed to SEASnet, and are used by the various Faculty during instruction. These would typically be PC-DOS compatible applications, which would be run on the PCs in the PC classrooms. The applications would be stored on the Backbone Servers, and brought over via PCI. Sometime these applications would use large amounts of data which would also be retrieved off the Backbone Servers.

A different task class that student were asked to perform involved not only use of canned applications, but also writing their own software. This could be either a stand-alone program or a program which is part of yet another large application provided by SEASnet. In this case the students would use software engineering

Martin	EE115A	Fall 1987	
Jacobsen	EE236A	Fall 1987	^
Mortensen	EE241A	Fall 1987	*
Dhir	MA105D	Fall 1987	
Stephenson	MA133A	Fall 1987	
Karagozian	MA150A	Fall 1987	*
Meyer	MA150P	Fall 1987	
Miu	MA162A	Fall 1987	
Forouhar/Youn(TA)	MA171A	Fall 1987	*
Wagner	MSE145A	Fall 1987	
Neethling	CE15A	Winter 1988	^
Neethling	CE15B	Winter 1988	^
Pfeiffer	CE106A	Winter 1988	
Fourney	CE130F	Winter 1988	
Selna	CE142	Winter 1988	
Felton	CE180	Winter 1988	^
Felio	CE221	Winter 1988	
Dong	CE235B	Winter 1988	
Felton	CE240	Winter 1988	^
Dracup	CE250A	Winter 1988	
Dracup	CE250B	Winter 1988	
Stenstrom	CE253	Winter 1988	
Neethling	CE255A	Winter 1988	^
Manousiouthakis	CH138	Winter 1988	%
Manousiouthakis	CH239AI	Winter 1988	
Kay/Wong(TA)	CS12	Winter 1988	*
Jefferson	CS111	Winter 1988	%
Bargodia	CS131	Winter 1988	
Cardenas	CS141	Winter 1988	
Rennels	CS152A	Winter 1988	
Klinger	CS161	Winter 1988	
Dyer	CS163	Winter 1988	
Aoki	CS170	Winter 1988	
Parker	CS239	Winter 1988	
Tyree	CS258BC	Winter 1988	
Flowers	CS264A	Winter 1988	*
Hillestad	EE103	Winter 1988	
Jacobsen	EE136	Winter 1988	^
Jacobsen	EE236B	Winter 1988	^
Wilcox	MA150A	Winter 1988	
Dinavari	MA154A	Winter 1988	
Yam/Youn(TA)	MA171A	Winter 1988	*
Friedmann	MA269B	Winter 1988	

* - Instructor or TA was interviewed w.r.t. this course.
^ - Instructor was interviewed but not w.r.t. this course.
% - SEASnet accounts were not used for this course.
The PC's wrer used as stand alone devices.

Figure 4.2: SEASnet - Instructors and Courses for Winter 1988

packages, such as compilers, linkers, debuggers etc. This was a predominant mode of operation for Computer Science classes. Appendix A contains a list of software packages licensed to SEASnet.

4.2 User Load Patterns

User loading is very dependent upon the engineering discipline, class taken, software package nature, as well as user education and computer literacy. The scope of user application is extremely wide, and ranges from Scheme programming to Propulsion Analysis. In fact, there is so much variation in the nature of the applications themselves, that we had to make some important technical judgements at this point to obtain a correct approach that would render our problem tractable. Since the computations themselves are done by the PCs, we focused our interest on the services provided by the network, namely remote file service.

4.2.1 Local and Remote Resources

PCI users use the PCs for all their computation needs, as DOS applications are exclusively utilized. Hence, local CPU use by the PCs is similar to other standalone DOS settings. Our interests are focused on the network service that PCI is rendering to the PCs. In fact, when users access their Network Drive (E:), is when our system becomes interesting. In these instances the network is accessed, and central resources at the backbone servers are put to work.

Whenever user obtain software from central resources, central service and network traffic are generated. In order to be able to account for user behavior, we had to get a realistic picture of the actual user demands of the central resources. These demands depend not only on the specific application, but also on user techniques and familiarity with the system.

4.2.2 User Skill and Literacy

The degree of user familiarity with SEASnet is variable. Many users are familiarizing themselves with DOS as well as with PCI. Such users are far less aware of remote resources as opposed to local resources. They use drive E: transparently, and do not obtain local copies of software and data onto their local hard disk (drive C:).

More experienced users realize that applications get run slower when used from remote resources, especially under high system load in peak times. These users localize data as much as possible, so they become less dependent on central service.

4.3 Estimation of User Load

In order to assess what particular users are demanding of SEASnet's PCI service, we devised several approaches. On one hand, we wanted a global picture of the tasks that are to be handled by SEASnet as an instructional tool. On the other hand we wanted to sample actual network traffic that is handled by

SEASnet at various times. We have decided to combine these two approaches, as they are complementing in nature. In this section we will describe the global approach. In the following section we will emphasize the measurement approach.

4.3.1 Variations in Computing Needs

This part of the work consists of an in depth understanding of the computing assignments that the faculty of the school have assigned to their students. This work is described in more detail in [Segal88]. The idea was to inquire the various instructors as to the nature of their teaching and required projects and assignments.

It turns out that the various disciplines within the schools have different requirements. Let us look at the various applications studied:

1. Water Resources classes - involved solution of field problems, described by Partial Differential Equations. Specialized software packages as well as spreadsheets were used for solutions of water quality problems.

2. Artificial Intelligence - classes used Scheme programming and Lisp. Major software construction effort. PC-Scheme obtained from servers.

3. Jet Propulsion Classes used canned software to evaluate Propulsion projects. Programs were stored on the servers.

4. Fluid Mechanics - mostly field problems, canned software was used. Distributed by Floppy discs.

5. Stochastic Control - Matrix algebra manipulation. Canned programs were stored on SEASnet servers. These were large executables, about 360K.

6. Dynamic System Control - parametric evaluation of dynamic systems were carried out by large canned software programs.

7. Structural Design - use of canned software to evaluate stress, strength and other design parameters. Canned software stored on Backbone Servers. Compiler and linker stored locally. Small assignments were given as well as a large project. Short routines written by students.

8. Introduction to Computer Science - C and PC-Scheme used to write programs. C compiler resides locally on PC, Scheme is on net. Most students worked on Network drive. Intensive writing and debugging efforts.

9. Data Structures - most of the work here was on Turbo Pascal. This software is locally installed on the PCs, and the network was utilized primarily for assignment and date distribution as well as e-mail.

10. Numerical Computing - Large object modules were provided through the Backbone Servers. Students compiled and linked short procedures to larger Linear Solution packages. For the large projects 300-500 lines of code were written by the students.

The actual information obtained was far more detailed, and Appendix B contains some more of this detail. We present in Figure 4.3 a smple of the data more of which was collected during the course of this work.

We shall now concentrate our attention to the derivation of the user model from such data.

4.3.2 Instructors and Students

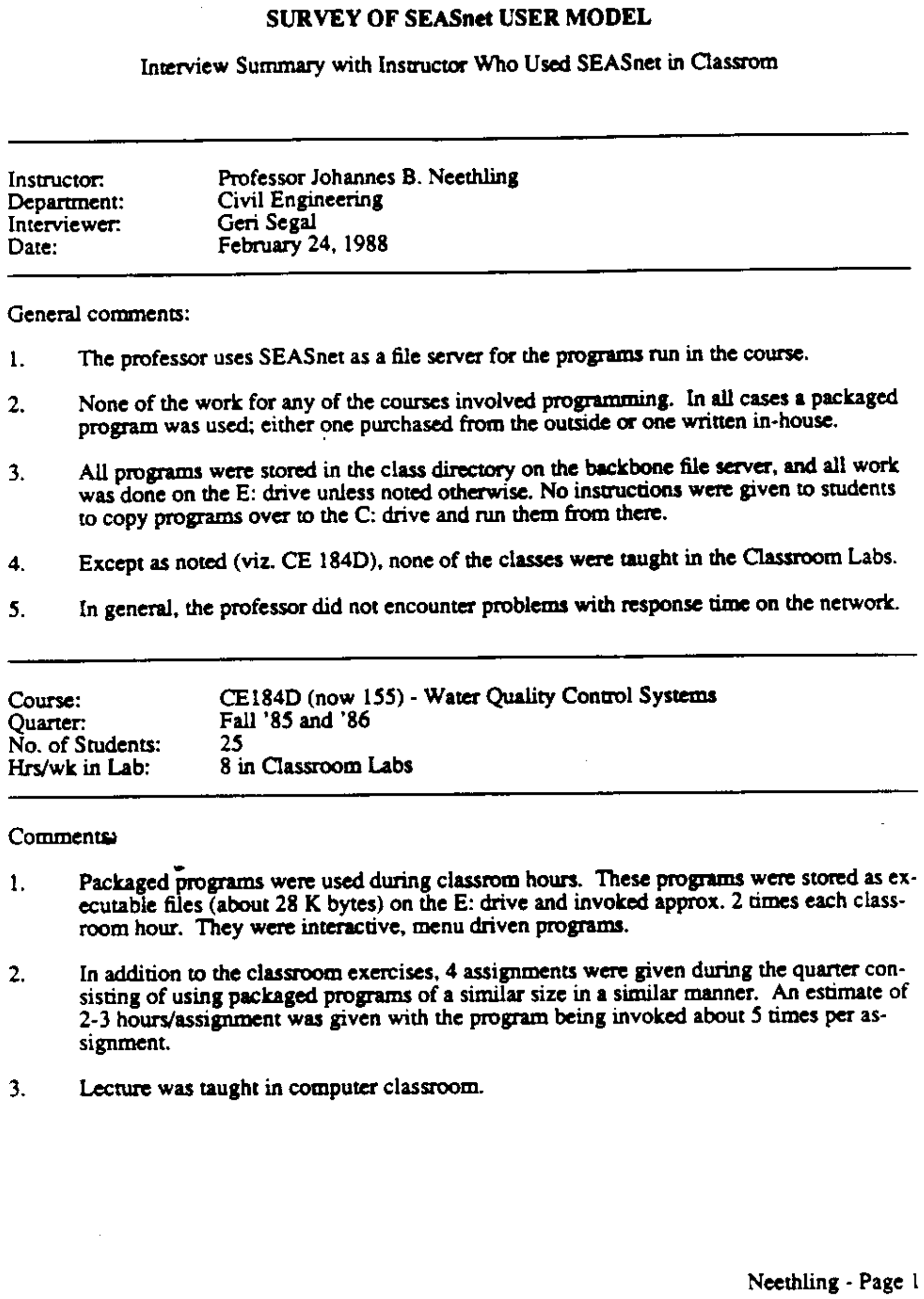
The instructors in the school were surveyed with respect to the workload, and access frequency that the users were using the central resources. The corresponding file sizes, the user literacy and resource localizing were taken into account, as well as course frequency and user population.

In the process of data gathering, we addressed not only instructors, but also Teaching Assistant and some Students as well. This was done to obtain additional insights and data points to tune our assessments.

We have reduced the detailed data to obtain average expected rates of network communication and central server demands. Figure 4.4 shows us the kind of data gathering that was required in order to obtain our global traffic requirement numbers. In fact, the average expected traffic according to instructor's expectation was in excess of 20 K/Byte sec. We shall see that this is higher than actually measured off the network.

4.4 Measurements of Communication Traffic

In addition to the ongoing efforts of global data gathering from the users, we decided to sample SEASnet traffic at various points in time. The measuring



SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor Johannes B. Neethling
Department:	Civil Engineering
Interviewer:	Geri Segal
Date:	February 24, 1988

General comments:

1. The professor uses SEASnet as a file server for the programs run in the course.
2. None of the work for any of the courses involved programming. In all cases a packaged program was used; either one purchased from the outside or one written in-house.
3. All programs were stored in the class directory on the backbone file server, and all work was done on the E: drive unless noted otherwise. No instructions were given to students to copy programs over to the C: drive and run them from there.
4. Except as noted (viz. CE 184D), none of the classes were taught in the Classroom Labs.
5. In general, the professor did not encounter problems with response time on the network.

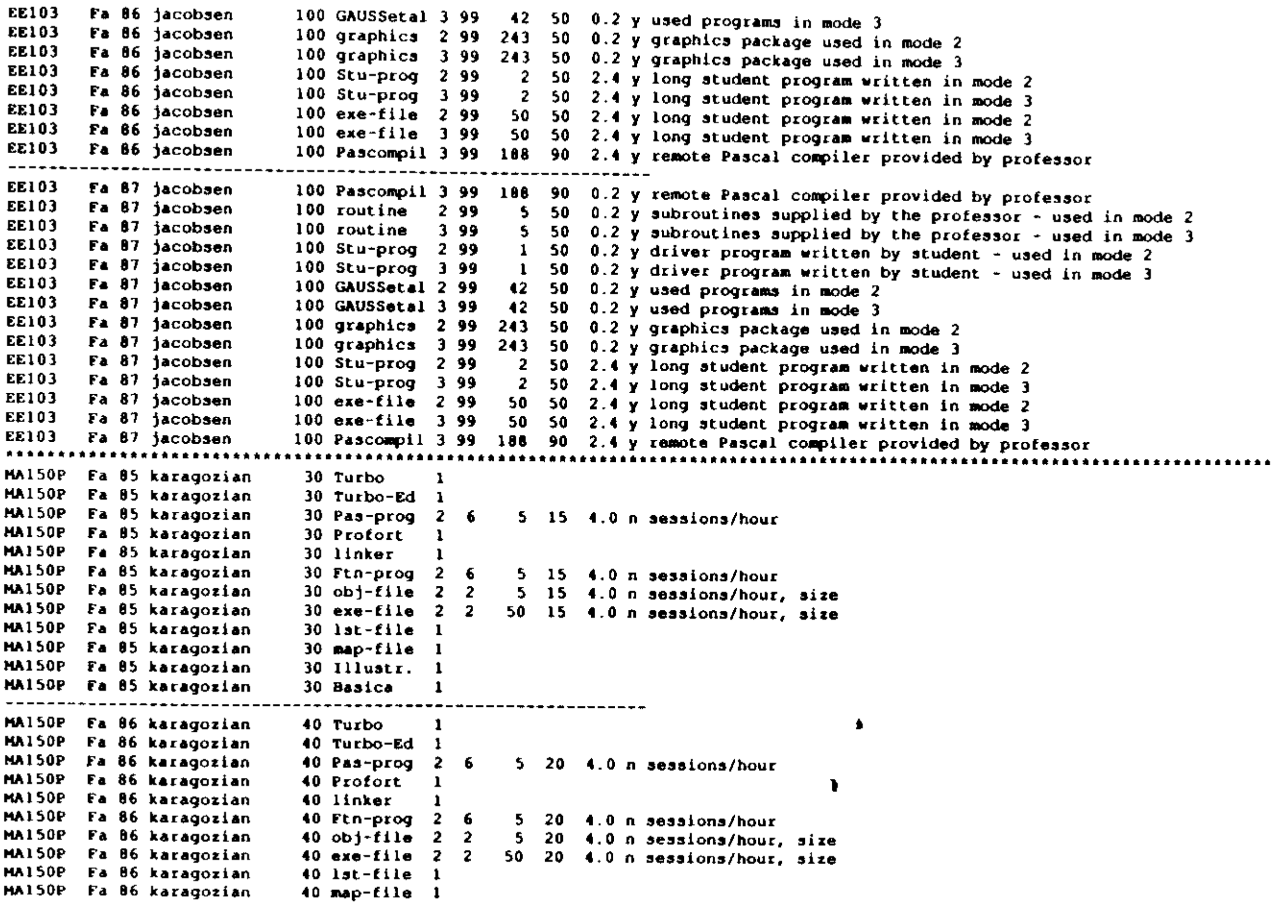
Course:	CE184D (now 155) - Water Quality Control Systems
Quarter:	Fall '85 and '86
No. of Students:	25
Hrs/wk in Lab:	8 in Classroom Labs

Comments:

1. Packaged programs were used during classrom hours. These programs were stored as executable files (about 28 K bytes) on the E: drive and invoked approx. 2 times each classroom hour. They were interactive, menu driven programs.
2. In addition to the classroom exercises, 4 assignments were given during the quarter consisting of using packaged programs of a similar size in a similar manner. An estimate of 2-3 hours/assignment was given with the program being invoked about 5 times per assignment.
3. Lecture was taught in computer classroom.

Neethling - Page 1

Figure 4.3: Sample Data Obtained from Instructor Regarding Expected SEASnet Usage



```
EE103    Fa 86 jacobsen      100 GAUSSetal 3 99   42  50  0.2 y used programs in mode 3
EE103    Fa 86 jacobsen      100 graphics  2 99  243  50  0.2 y graphics package used in mode 2
EE103    Fa 86 jacobsen      100 graphics  3 99  243  50  0.2 y graphics package used in mode 3
EE103    Fa 86 jacobsen      100 Stu-prog  2 99    2  50  2.4 y long student program written in mode 2
EE103    Fa 86 jacobsen      100 Stu-prog  3 99    2  50  2.4 y long student program written in mode 3
EE103    Fa 86 jacobsen      100 exe-file  2 99   50  50  2.4 y long student program written in mode 2
EE103    Fa 86 jacobsen      100 exe-file  3 99   50  50  2.4 y long student program written in mode 3
EE103    Fa 86 jacobsen      100 Pascompil 3 99  188  90  2.4 y remote Pascal compiler provided by professor
-----------------------------------------------------------------------
EE103    Fa 87 jacobsen      100 Pascompil 3 99  188  90  0.2 y remote Pascal compiler provided by professor
EE103    Fa 87 jacobsen      100 routine   2 99    5  50  0.2 y subroutines supplied by the professor - used in mode 2
EE103    Fa 87 jacobsen      100 routine   3 99    5  50  0.2 y subroutines supplied by the professor - used in mode 3
EE103    Fa 87 jacobsen      100 Stu-prog  2 99    1  50  0.2 y driver program written by student - used in mode 2
EE103    Fa 87 jacobsen      100 Stu-prog  3 99    1  50  0.2 y driver program written by student - used in mode 3
EE103    Fa 87 jacobsen      100 GAUSSetal 2 99   42  50  0.2 y used programs in mode 2
EE103    Fa 87 jacobsen      100 GAUSSetal 3 99   42  50  0.2 y used programs in mode 3
EE103    Fa 87 jacobsen      100 graphics  2 99  243  50  0.2 y graphics package used in mode 2
EE103    Fa 87 jacobsen      100 graphics  3 99  243  50  0.2 y graphics package used in mode 3
EE103    Fa 87 jacobsen      100 Stu-prog  2 99    2  50  2.4 y long student program written in mode 2
EE103    Fa 87 jacobsen      100 Stu-prog  3 99    2  50  2.4 y long student program written in mode 3
EE103    Fa 87 jacobsen      100 exe-file  2 99   50  50  2.4 y long student program written in mode 2
EE103    Fa 87 jacobsen      100 exe-file  3 99   50  50  2.4 y long student program written in mode 3
EE103    Fa 87 jacobsen      100 Pascompil 3 99  188  90  2.4 y remote Pascal compiler provided by professor
*****************************************************************************************************************************
MA150P   Fa 85 karagozian     30 Turbo     1
MA150P   Fa 85 karagozian     30 Turbo-Ed  1
MA150P   Fa 85 karagozian     30 Pas-prog  2  6    5  15  4.0 n sessions/hour
MA150P   Fa 85 karagozian     30 Profort   1
MA150P   Fa 85 karagozian     30 linker    1
MA150P   Fa 85 karagozian     30 Ftn-prog  2  6    5  15  4.0 n sessions/hour
MA150P   Fa 85 karagozian     30 obj-file  2  2    5  15  4.0 n sessions/hour, size
MA150P   Fa 85 karagozian     30 exe-file  2  2   50  15  4.0 n sessions/hour, size
MA150P   Fa 85 karagozian     30 lst-file  1
MA150P   Fa 85 karagozian     30 map-file  1
MA150P   Fa 85 karagozian     30 Illustr.  1
MA150P   Fa 85 karagozian     30 Basica    1
-----------------------------------------------------------------------
MA150P   Fa 86 karagozian     40 Turbo     1
MA150P   Fa 86 karagozian     40 Turbo-Ed  1
MA150P   Fa 86 karagozian     40 Pas-prog  2  6    5  20  4.0 n sessions/hour
MA150P   Fa 86 karagozian     40 Profort   1
MA150P   Fa 86 karagozian     40 linker    1
MA150P   Fa 86 karagozian     40 Ftn-prog  2  6    5  20  4.0 n sessions/hour
MA150P   Fa 86 karagozian     40 obj-file  2  2    5  20  4.0 n sessions/hour, size
MA150P   Fa 86 karagozian     40 exe-file  2  2   50  20  4.0 n sessions/hour, size
MA150P   Fa 86 karagozian     40 lst-file  1
MA150P   Fa 86 karagozian     40 map-file  1
```

Figure 4.4: Computerized Course Work Data Gathering for the Purpose of Studying Network Traffic

software was placed centrally, at the backbone server. This way we obtained traffic information generated by the entire user population.

The collected data contained information about specialized pci packets that were transmitted over the PCI links. That way we could study actual network use by the user population.

4.4.1 The PCI Protocol Logging

Logging the operation of the DOS-SERVER on the backbone server machine produces information regarding the type, size, and frequency of the PCI messages that are traveling on the network. Figure 4.5 is an example of such logging information.

Collecting this type of information at various times in the quarter has led us to the following key observations:

1. Users transmit large files sporadically, mainly at session start

2. Some users carry sporadic communication throughout the session

3. Some users continue large file transfers

4. Many users have long idle periods with respect to network service. This might be also due to intensive local computation on the PC.

5. Traffic has high variance, with average of 4-6 Kbyte/sec.

```
Got request=2                    ------------------2-CREATE ----
hdr: res 0; req 2; stat 41; seq 144; drvNum 0; mode: 0
...: t_cnt 33; b_cnt 16350; inode 0; date 0; time 0; pid 11076
...: fdsc 0; f_size 77; offset 608; pattern 0x20103; attr 0
ascii: \U\SEASGRAD\CS\BETSER\TIMEEC.BAT<00>
create: \U\SEASGRAD\CS\BETSER\TIMEEC.BAT pid 0x2b44 mode 0 attr 0
hdr: res 0; req 2; stat 41; seq 144; drvNum 0; mode: 1
...: t_cnt 0; b_cnt 0; inode 278; date 4222; time 30501; pid 0
...: fdsc 0; f_size 0; offset 0; pattern 0; attr 0x20

Sending >>                       >>>>CREATE Response 0 SUCCESS <<<<

Got request=5                    ------------------5-NEW_CLOSE ----
hdr: res 0; req 5; stat 41; seq 145; drvNum 0; mode: 0
...: t_cnt 0; b_cnt 16350; inode 0; date 0; time 0; pid 11076
...: fdsc 0; f_size 77; offset 608; pattern 0x20103; attr 0
new_close: fdsc 0 size 4d
hdr: res 0; req 5; stat 41; seq 145; drvNum 0; mode: 0
...: t_cnt 0; b_cnt 0; inode 0; date 0; time 0; pid 0
...: fdsc 0; f_size 0; offset 0; pattern 0; attr 0

Sending >>                       >>>>NEW_CLOSE Response 0 SUCCESS <<<<

Got request=18                   ------------------18-NEW_OPEN ----
hdr: res 0; req 18; stat 41; seq 146; drvNum 0; mode: 1
...: t_cnt 33; b_cnt 16350; inode 0; date 0; time 0; pid 11076
...: fdsc 0; f_size 77; offset 608; pattern 0x20103; attr 0
ascii: \U\SEASGRAD\CS\BETSER\TIMEEC.BAT<00>
new_open: \U\SEASGRAD\CS\BETSER\TIMEEC.BAT mode 0x1 pid 0x2b44 attr 0
hdr: res 0; req 18; stat 41; seq 146; drvNum 0; mode: 1
...: t_cnt 0; b_cnt 0; inode 278; date 4222; time 30501; pid 0
...: fdsc 0; f_size 0; offset 0; pattern 0; attr 0x20

Sending >>                       >>>>NEW_OPEN Response 0 SUCCESS <<<<

Got request=40                   -------------------40-DEVICE_INFO_C ----
hdr: res 0; req 40; stat 41; seq 147; drvNum 0; mode: 1
...: t_cnt 33; b_cnt 16350; inode 0; date 0; time 0; pid 11076
...: fdsc 0; f_size 77; offset 608; pattern 0x20103; attr 0
ascii: \U\SEASGRAD\CS\BETSER\TIMEEC.BAT<00>
deviceinfo: fdsc 0
hdr: res 0; req 40; stat 41; seq 147; drvNum 0; mode: 64
...: t_cnt 0; b_cnt 0; inode 0; date 0; time 0; pid 0
...: fdsc 0; f_size 0; offset 0; pattern 0; attr 0

Sending >>                       >>>>DEVICE_INFO_C Response 0 SUCCESS <<<<
```

Figure 4.5: Example of Collected PCI Logging Information

4.4.2 Assessment and Evaluation of Network Utilization

Combining the information obtained through both techniques, we obtain the following conclusions:

1. Traffic on SEASnet is highly variable
2. A tendency for specific behavior patterns was observed
3. Lower utilization than predicted was observed. Users tend to depend more on own media.
4. Centralized service could be utilized to a higher degree

4.5 Derivation of the Three User Classes

The study and the measurements we performed, led us into the definition of three distinct user classes that together have the ability to describe most usage scenarios. The user classes describe the range of very intensive PCI service to very light traffic usage. The idea is that the combination of users of the various classes will describe the usage patterns that are of interest to us, and thus could be used as inputs to the performance model. Various combinations of the user classes could also be used to simulate user load on SEASnet for the purpose of conducting measurements under changing user loads.

4.5.1 Communication Intensive (CI)

This is an intensive data transfer from the server to the PC, at the maximum rate that the system will permit. Technically this is accomplished by continuous (infinite) transfer of large files. This is the most a user can get out of PCI service. Surely there could be only very few users with such high demands, but is is an important limiting situation to study. We shall elaborate on this in Chapter 5 and the rest of this Dissertation.

4.5.2 High Interactive (HI)

This is a limiting situation for a super intensive interactive session. Such session consists of a 20K file transfer back and forth, 3 remote directory lists executed continuously, without local user think time. This again is somewhat unrealistic user load, but it can test the system capacity under such heavy load, and be used as a building block for given user loads.

4.5.3 Interactive (Int)

This is a reasonable session with operations identical to the High Interactive class. The big difference is that think times of 60 seconds are added to the remote operations described above. These think times correspond to actual user think time, as well as PC-local operations such as compilation, execution, etc.

The actual definition of class Int is as follows :

1. List remote directory

2. Read 20K file from backbone server

3. 60 sec think time

4. List remote directory

5. Write 20K file on backbone server

6. List remote directory

7. 60 sec think time

Class HI is similar, but without the think times.

4.6 Summary

In this Chapter we have reviewed user operations and behavior with respect to PCI service on SEASnet. We have described our observations and studies of the user community, and how it can be characterized for the purpose of modeling. We have shown what measurement on SEASnet have obtained, in the way of actual traffic and demands experienced by the central resources.

We then combined the various observations into a dynamic and modular approach that allows us to compose various user loads, with 3 types of user classes we have derived.

We shall use these results in the rest of the dissertation for the purposes of conducting further measurements, constructing a general SEASnet model, and synthesis of future networks.

CHAPTER 5

System Performance Measurements and Observations

This chapter contains the bulk of the experimental contribution of our work. The process of analyzing and modelling requires technical knowledge of the system at hand. Not all parts and modules of the system wher open and available to us. We did conduct an extensive measurement plan, that evolved with the model design, as well as system growth and enhancement. We note that this general methodology applies to other instances of experimental performance work, particularly in the cases where a a real system is analyzed.

In order to build a performance model for SEASnet, system parameters needed to be measured, and system behavior had to be monitored. We have conducted several experimentation sets prior to and during model construction. First, we measured a homogeneous O/S - Locus exclusively across the net. We then proceeded to our target heterogeneous system - SEASnet, consisting of Locus backbones and DOS clients. This chapter reports these measurement results, and provides thrust and motivation for the analysis that follows.

5.1 Measurement Approach

The initial set of measurements was designed to give us performance measures and a good design intuition for the system under investigation. We therefore devised a set of operation scenarios, since it was important for us to acquire performance bounds in order to proceed to the modeling phase of the work.

It is important to understand that the presentation follows the chronological development of the project as we witnessed it evolve. At first we started with a system that was most familiar to us, i.e. the AT/VAX network. This was the first instance of demonstration of Locus operating across very different machine architectures. The experimentation was carried out on a research network, rather than on a stable production net. These initial measurements gave us an interesting set of results, as well as a feel for some of the problems at hand. However, we progressed on to the more interesting setting of heterogeneous operating systems on top of machine heterogeneity. This presented far higher potential for scaling and growth, as well as a forward step with the evolving technology supporting SEASnet. This will also coincide with future research directions, that are evolving, as SEASnet and other environments are continuously enriching their hardware and software capabilities. We note here that major modeling emphasis was given to the heterogeneous SEASnet paradigm.

It should be emphasized that the process of modeling evolved as the measurements progressed, and our knowledge and understanding were growing. In fact the iterative process of model design and refinement took place as the experimental part of our work was proceeding. Hence, for presentation purposes we separated measurement from modeling, but we keep in mind their close relationship in our work.

5.2 Measurements on PCs/Backbones Running Locus

This is an introductory phase of the measurement phase. The hardware used was heterogeneous as both PC/ATs and VAX750 backbone machines were used. However, the operating systems were homogeneous, as Locus was used across the network. We shall first present results pertaining to Locus/AT performance as supporting up to seven terminals of interactive work. We than present performance of networked ATs and VAXen running homogeneous Locus.

5.2.1 The Load Presented to the System

Various benchmarks have been constructed to test system performance. The common denominator to all is that they simulate user behavior in an intensive fashion. If people are convinced that the system under examination was subject to load patterns which are characteristic to the environment, they will trust the measurement results.

Indeed, the load configuration is a key issue in the composition of a reliable system model. We feel confident here at UCLA in representing a typical academic load. In fact, we have instrumented a LOCUS environment to monitor and

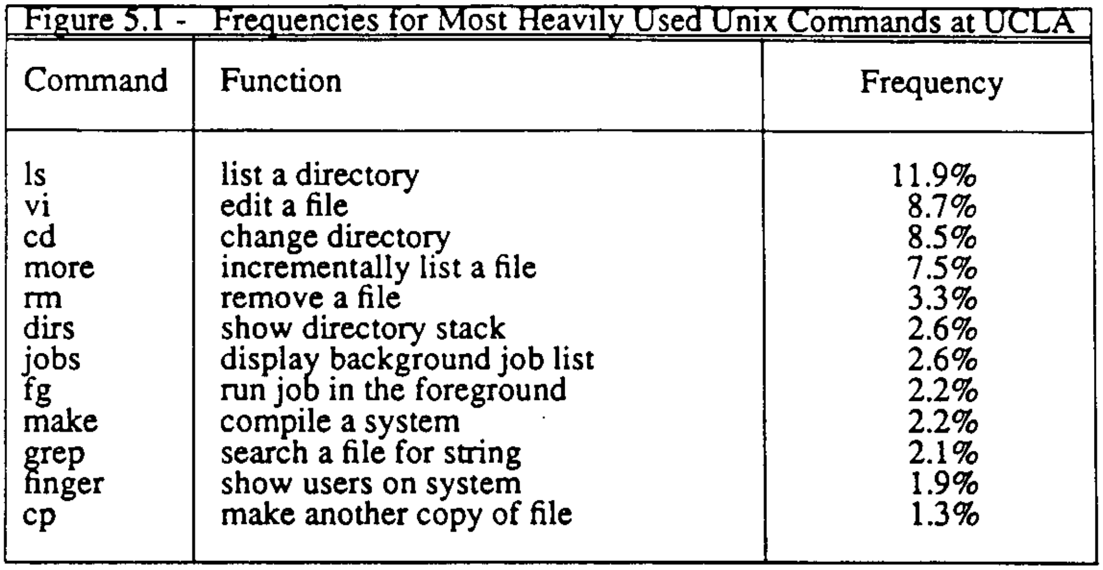


Figure 5.1 - Frequencies for Most Heavily Used Unix Commands at UCLA

Command	Function	Frequency
ls	list a directory	11.9%
vi	edit a file	8.7%
cd	change directory	8.5%
more	incrementally list a file	7.5%
rm	remove a file	3.3%
dirs	show directory stack	2.6%
jobs	display background job list	2.6%
fg	run job in the foreground	2.2%
make	compile a system	2.2%
grep	search a file for string	2.1%
finger	show users on system	1.9%
cp	make another copy of file	1.3%

Figure 5.1: Frequencies for Most Heavily Used Unix Commands at UCLA

measure user activity [ShelPope86]. We have found that the user community has a strong tendency to use a specific set of commands and programs. This general result is a classic one, and has been shown in [Zipf49]. For a LOCUS operating system we have experimentally found the frequencies shown in Figure 5.1 under daily operation.

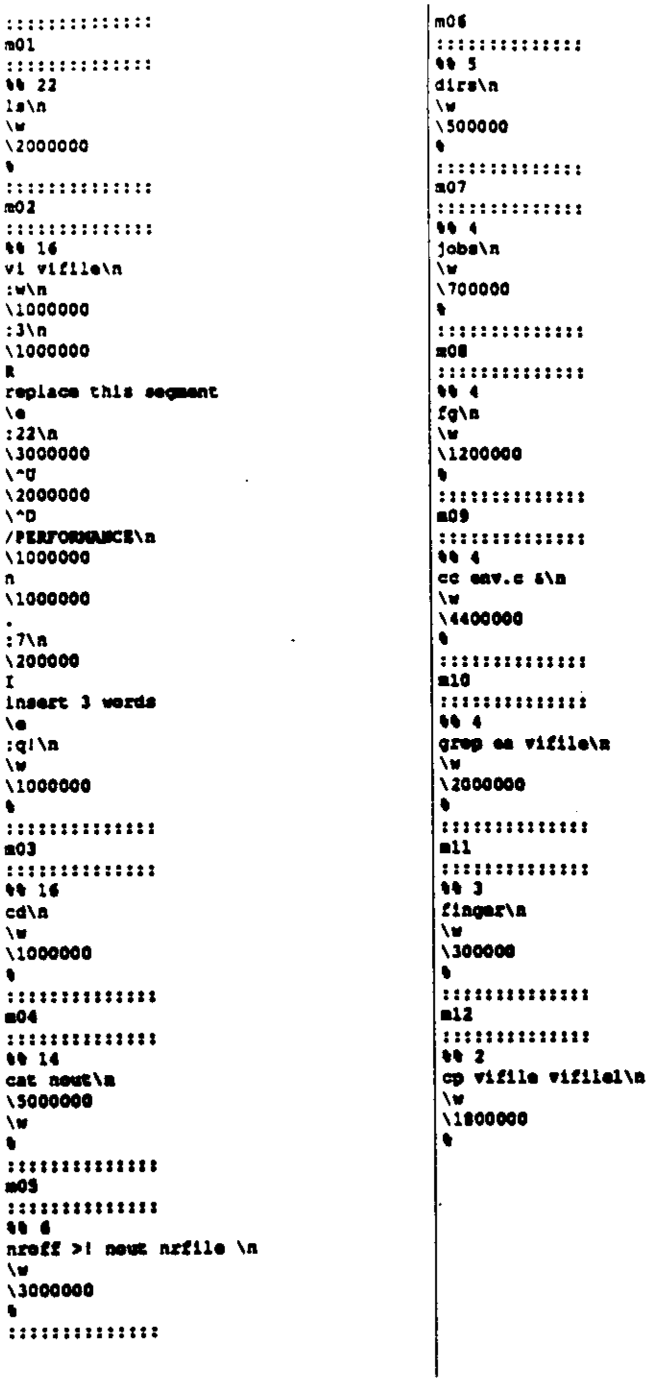
Based on our measurements, we have constructed a set of command-scripts as our base benchmarks for the current study (Figure 5.2). We have used these in our initial measurements with a single PC/AT and an AT backed up by backbone processors. We have also developed some I/O intensive scripts (Fig 5.7,5.8). These were composed as we believe that I/O performance is an important system metric. We have used these scripts for AT to backbone measurements, as well as to backbone to backbone measurements. We shall present these results in section 5.2.3 below.

5.2.2 PC/AT + Terminal Traffic

For the initial measurements, we have compiled user execution scripts. These scripts are to be executed at the corresponding execution probabilities presented in section 2.1 above. The user scripts include usage frequencies, and user think times, in addition to the actual commands to be invoked. Figure 4 lists the 12 scripts we used. The scripts can be executed at variable user typing speed. For the results reported below we used a typing speed of 10 characters per second. This is the simplest configuration possible. The initial measurements were completed on an IBM PC/AT running LOCUS. The AT was equipped with an 8-port Hostess Board[1]. The 8 users were simulated by a DEC Vax 11/750, which was running 8 user simulation processes (Figure 5.3). The user processes communicate (through the Vax's multiple serial ports) with the ports on the AT, thus simulating users on terminals. Each user process invokes user command scripts in assigned frequencies by the respective user processes.

Figure 5.4 below shows the elapsed time for each of the 12 scripts executed. This result is shown for 1,2,4, and 7 users on an AT. We note the increase in elapsed time with the increase in user population. This increase varies from

[1] Made by Control Systems, Minnesota and attaches to a serial board slot



```
::::::::::::::
m01
::::::::::::::
%% 22
ls\n
\w
\2000000
%
::::::::::::::
m02
::::::::::::::
%% 16
vi vifile\n
:w\n
\1000000
:3\n
\1000000
R
replace this segment
\e
:22\n
\3000000
\^U
\2000000
\^D
/PERFORMANCE\n
\1000000
n
\1000000
.
:7\n
\200000
I
insert 3 words
\e
:q!\n
\w
\1000000
%
::::::::::::::
m03
::::::::::::::
%% 16
cd\n
\w
\1000000
%
::::::::::::::
m04
::::::::::::::
%% 14
cat nout\n
\5000000
\w
%
::::::::::::::
m05
::::::::::::::
%% 6
nroff >! nout nrfile \n
\w
\3000000
%
::::::::::::::
```

```
m06
::::::::::::::
%% 5
dirs\n
\w
\500000
%
::::::::::::::
m07
::::::::::::::
%% 4
jobs\n
\w
\700000
%
::::::::::::::
m08
::::::::::::::
%% 4
fg\n
\w
\1200000
%
::::::::::::::
m09
::::::::::::::
%% 4
cc env.c &\n
\w
\4400000
%
::::::::::::::
m10
::::::::::::::
%% 4
grep ee vifile\n
\w
\2000000
%
::::::::::::::
m11
::::::::::::::
%% 3
finger\n
\w
\300000
%
::::::::::::::
m12
::::::::::::::
%% 2
cp vifile vifile1\n
\w
\1800000
%
```

Figure 5.2: Command Scripts Executed by Simulated Users

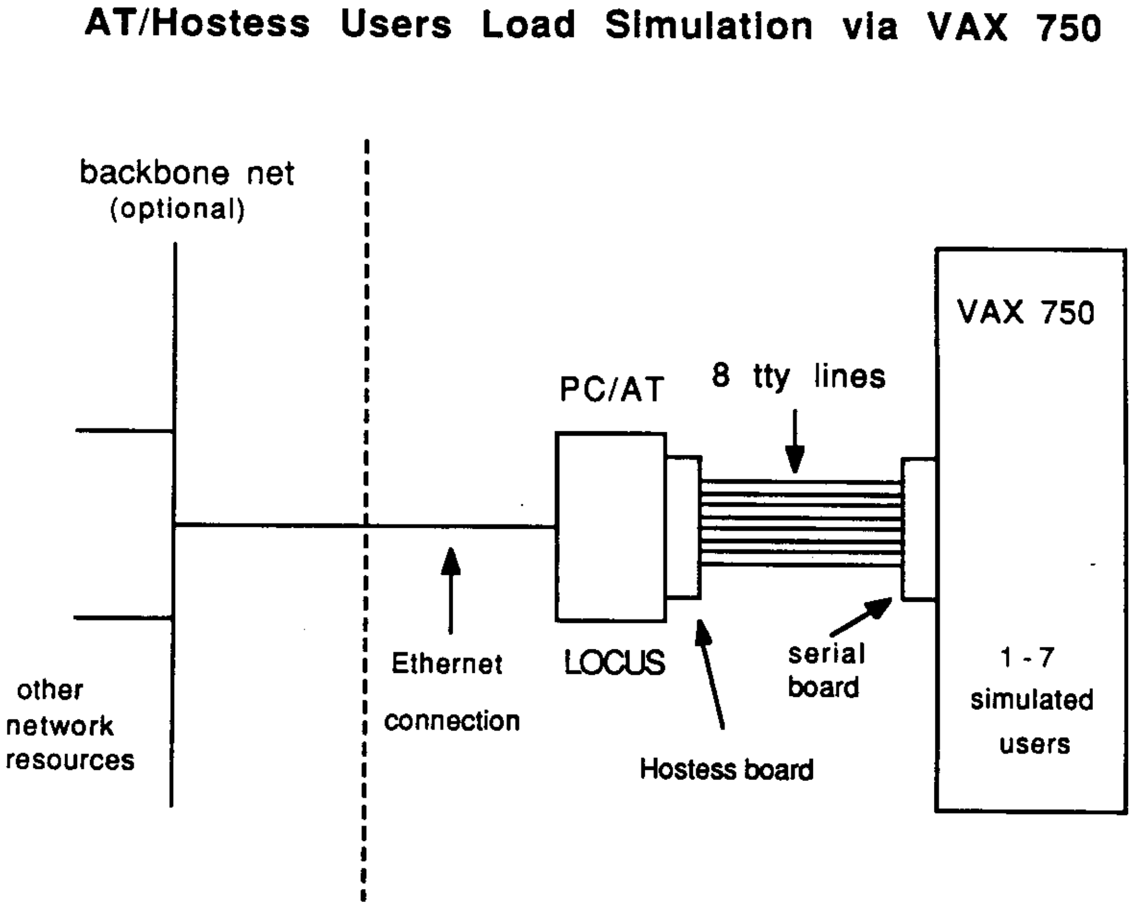


Figure 5.3: AT/Hostess Users Load Simulation via VAX 750

twice to three times of elapsed time for 7 users compared with a single user. We note that these times *include* user think time and typing time as per Figure 5.2 above. In Figure 5.5 we have aggregated the elapsed time to execute a suite of command scripts consisting of 100 randomly selected script executions, at the specified execution probabilities. We note that an average increase of 2.5 in elapsed time is incurred, when population increases from 1 to 7 users.

The last set of measurements performed on a single AT consisted of I/O bound experiments, namely file duplication (cp). These were done directly on the console of the AT involved using the csh *time* facility. Files of 670 KBytes were copied on the ATs hard disk (MAXTOR 1085). Figure 5.6 shows the nearly linear increase in response time of these non-interactive (batch) tasks. When 4 interactive users are added to the machine load, execution time increases, especially for the highly loaded situations, when many batch jobs run concurrently on that machine. These measurements provided a baseline for the I/O intensive experiments that were run on the backbone supported configuration.

We can summarize our results as follows: it is relatively easy to load an AT with computationally intensive jobs, such that interactive response time increases beyond several seconds, for the multiuser situation being studied. This indicates that addition of backbone servers carries potential merit.

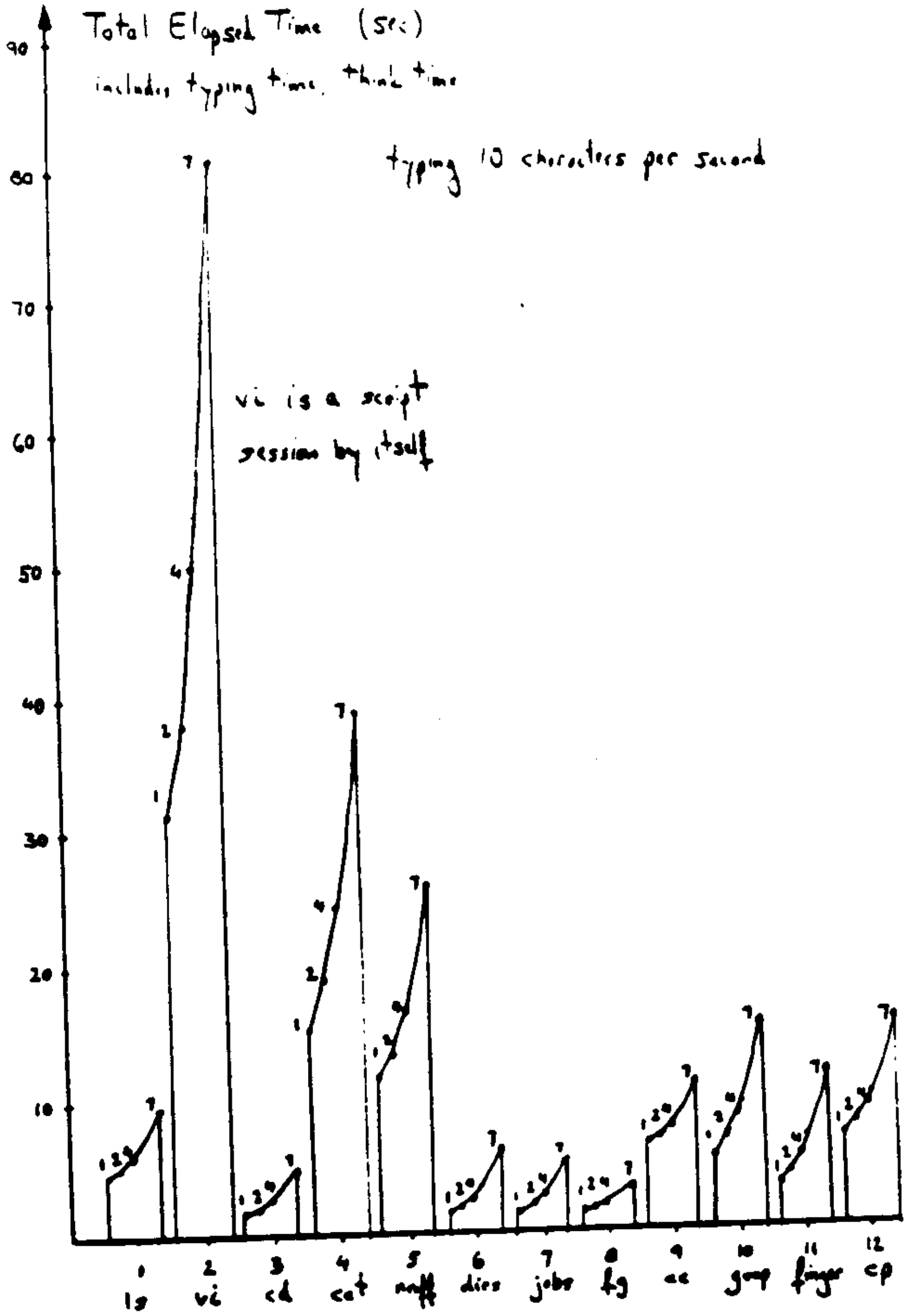


Figure 5.4: Elapsed Time vs. Number of Users (Individual Command Scripts)

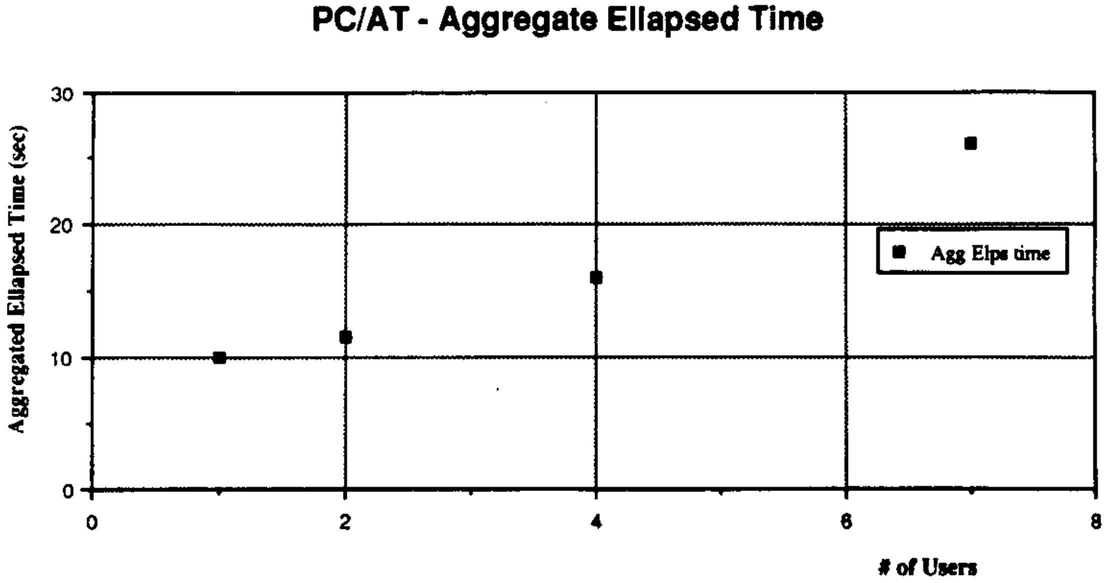


Figure 5.5: Aggregated Elapsed Time vs. Number of Users (Suite of Scripts)

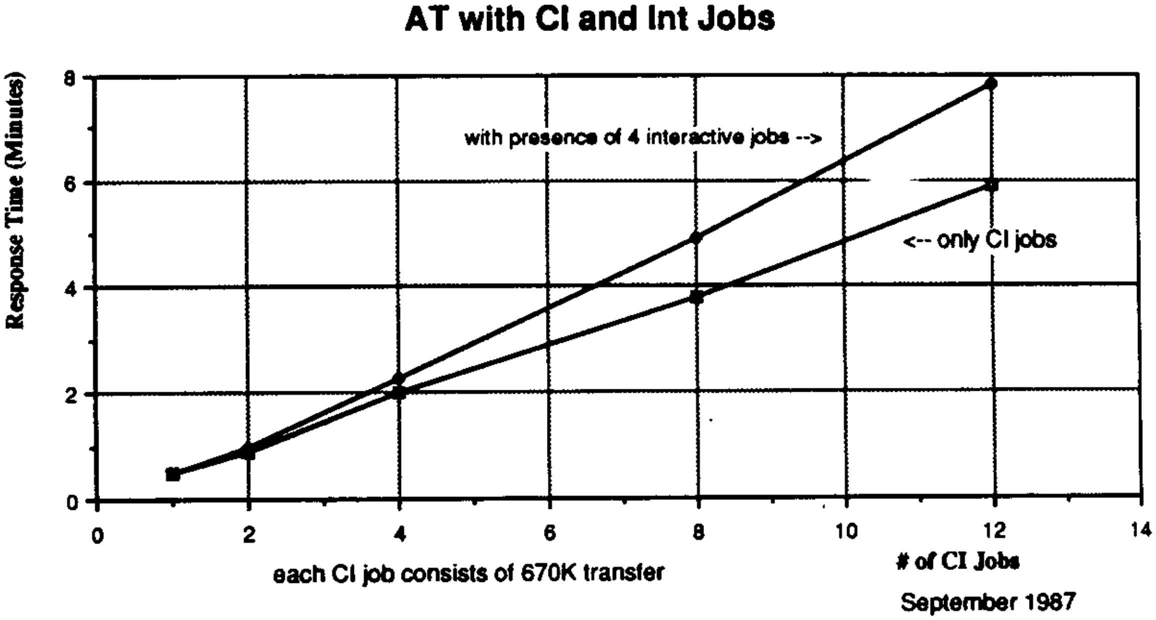


Figure 5.6: Response Time vs. Number of I/O Intensive Jobs

5.2.3 Multiple ATs and Backbone Servers

The study of this configuration gives us an interesting view of the computation support that a backbone servers provide to PCs. Many load balancing issues are involved [SouzGerl84], as well as tasking protocols between workstations and servers. There have been ongoing debates on the advantages and disadvantages of independent but slower local storage devices versus fast centralized ones [LazZahZwa85]. Our experimental work started with the addition of a VAX 11/750 and/or an IBM 4361 server to the multiuser LOCUS PC/AT configuration.

We conducted the I/O intensive experiments on the backbone and AT configurations. We were submitting copy file commands to various permutations of Storage Sites and User Sites. Load on all the machines was *low* during experimentation. Examining Figure 5.7 below, we note that there is a distinct difference between process execution on the originating site or receiving site. Execution on the receiving site is always faster, as read ahead and caching demand driven protocols are being utilized. Transfer of 670K files among machines took between 12 seconds and 43 seconds depending on machine speed and I/O devices. In some extreme cases this transfer takes several minutes. That happened only for some cases were the originating site runs the process. Similar behavior was observed when running equivalent experiments on IBM backbone servers on SEASnet. Figure 5.8 demonstrates file transfers of 1Mbyte in size. We were using 4361 (Thor) and 4381 (Odin) for these experiments. For the 4361 −− > 4381

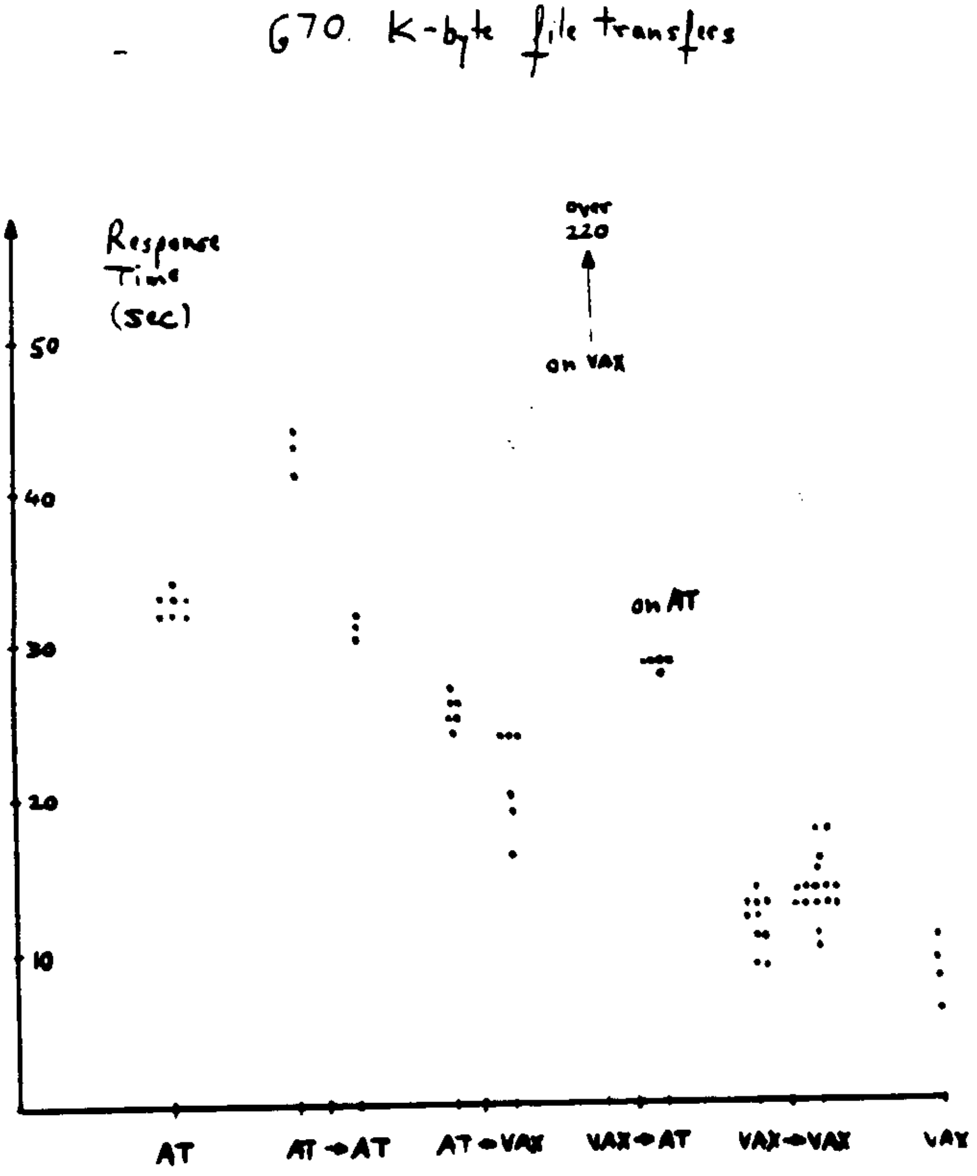


Figure 5.7: AT< −− >VAX Confs - Response Time for an I/O Intensive Task

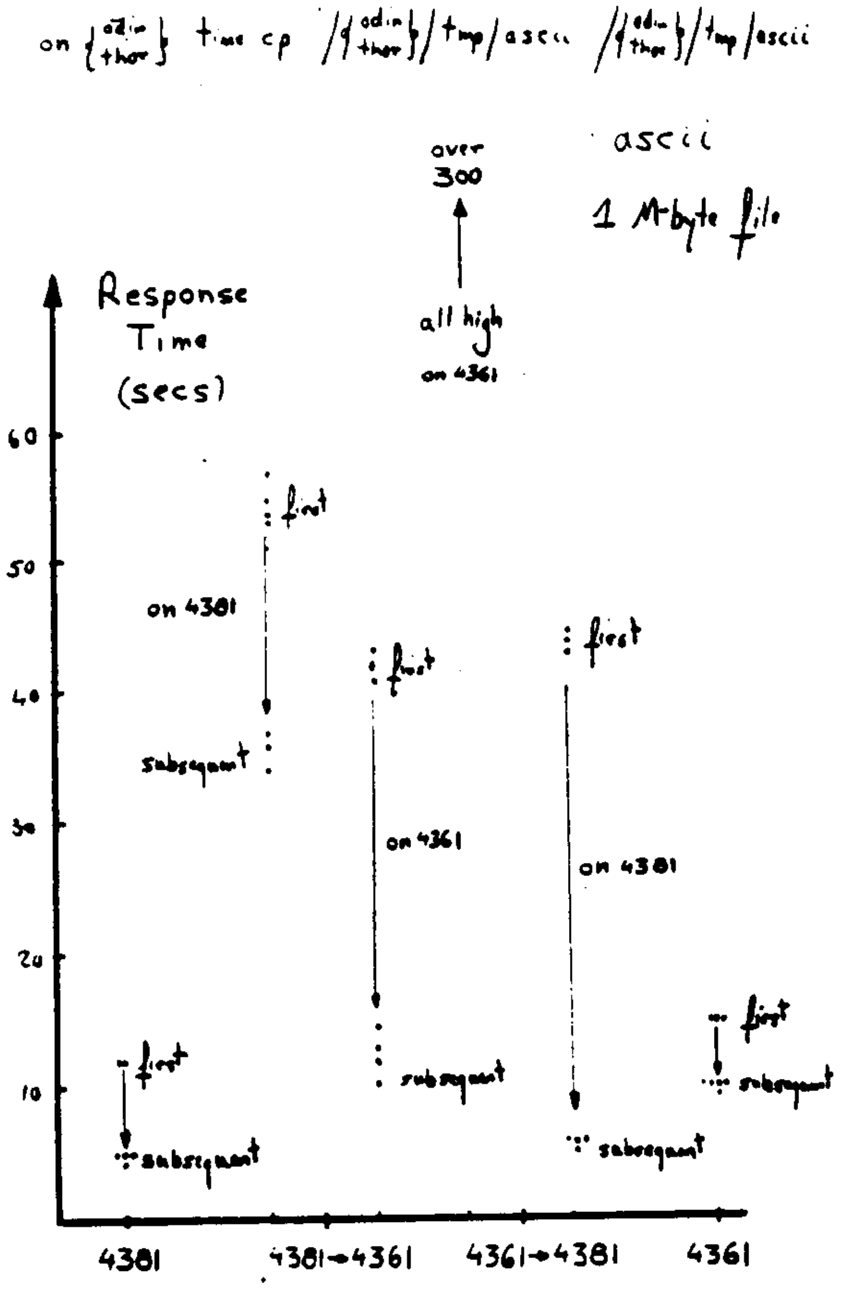


Figure 5.8: 4381< — — >4361 Confs - Response Time for an I/O Intensive Task

route, transfer completed in 5 minutes while executing[2] on the 4361. When executing the exact same command on the 4381, the transfer would complete in 44 seconds for the first time, and in a swift 5 seconds thereafter. This phenomenon was eliminated in 1988 release of Locus, as network protocols were enhanced.

5.3 Measurements on Heterogeneous H/W and O/S Configuration

This phase embarks on the more challenging aspects of studying heterogeneity. We have here not only different hardware on which the servers and clients run, but we also have heterogeneous operating systems running on the heterogeneous hardware. This environment is encapsulated by SEASnet, the School of Engineering and Applied Sciences Network. It is important to note that SEASnet is an ever evolving environment, and undergoes numerous improvements, expansions, and enhancements throughout our measurement phase. Some of these changes will be described as their effect on performance is observed. Also, SEASnet is our largest distributed network. It allows us to scale up to several dozen ATs (DOS/PCI) supported by 9 backbone machines running Locus. This architecture is described in Figure 5.9.

The user model was applied as input to SEASnet in several ways. As the heterogeneous loading to SEASnet occurs primarily by the PCs running DOS, and communicating with the LOCUS backbone servers. These PCs are used for course work by the students. Based on the fact that students access the

[2] on thor time cp /thor/tmp/ascii /odin/tmp/ascii

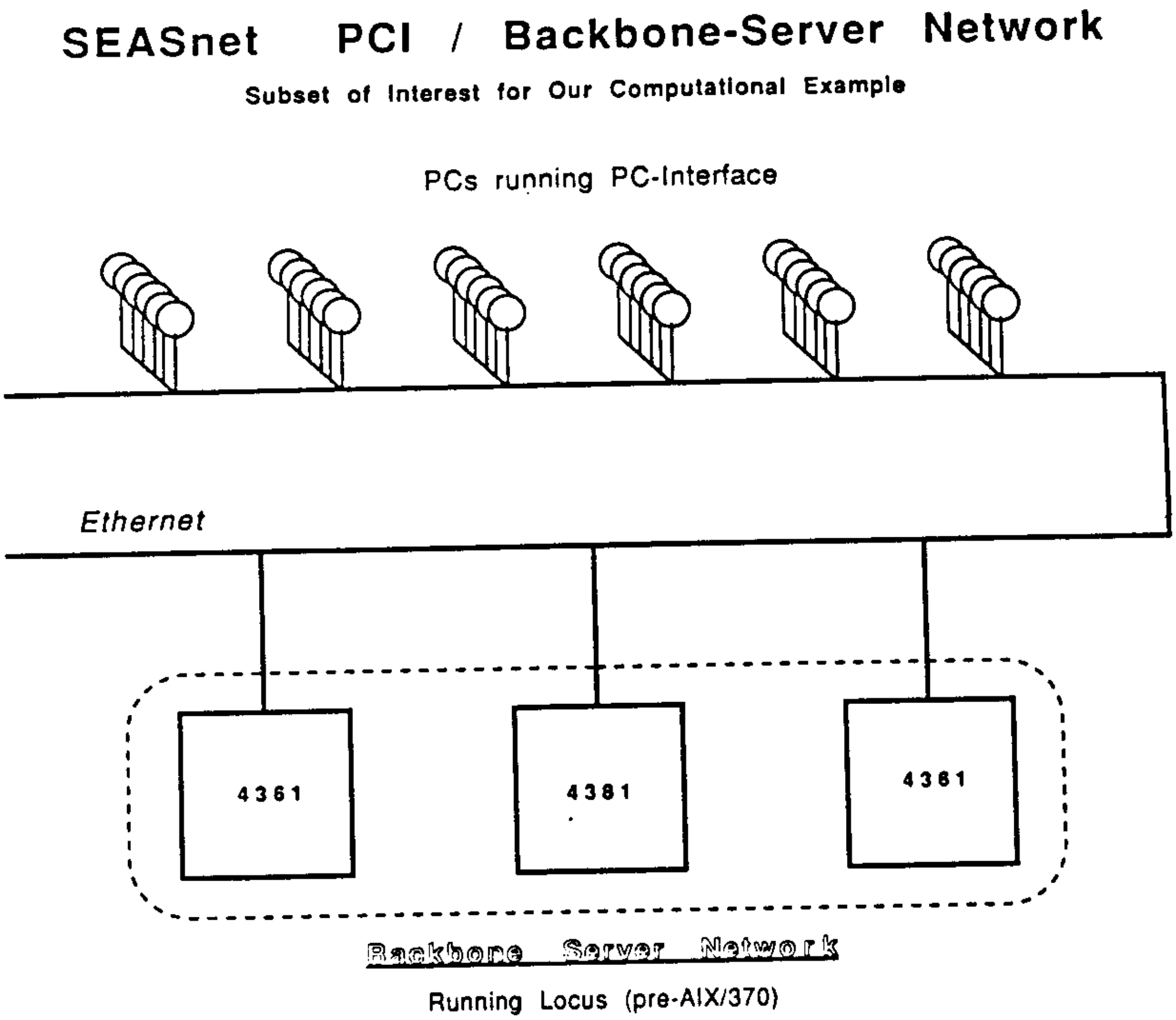

SEASnet PCI / Backbone-Server Network
Subset of Interest for Our Computational Example
PCs running PC-Interface
Ethernet
4361
4381
4361
Backbone Server Network
Running Locus (pre-AIX/370)

Figure 5.9: SEASnet - PCI / Heterogeneous Backbone-Server Network

central files through the network periodically, we have composed the user classes. Class Interactive (Int), for example, simulates sporadic access, common command execution, along with user think time.

At the same time, bounds on performance where sought, and Communication Intensive (CI) work was studied. these bounds teach us about system bottlenecks, and facilitate parameter extraction for the purpose of model construction. Benchmark files with 1MBytes of data were used to do most of these measurements. Measurements were done off hours, when no class service was being rendered and no other system load was present. This insured accuracy and reproducibility of testing conditions.

Since our main concern was in the heterogeneous part of the system, we were most interested in evaluating the operating system bridge. In our case we used the PC-Interface, although the methodology generally applies. We connected as many sessions as we liked to the network resources in question. Connections were made to specific backbone machines depending on the desired system configuration we wished to study.

5.3.1 Loading of Individual Backbone Servers

The goal in this phase of the measurements was to quantify the quality of file service that a backbone can deliver. We defined three classes (categories) of measurements in accordance with the results in Chapter 4. The Communication Intensive (CI) category consists of massive data transfer across the network. This heavy pounding on the operating system bridge teaches us about the ultimate bounds we might expect in the way of inter O/S transfer rates and overall system throughput. The High Interactive (HI) class consists of continuous executions of interactive commands such as list directory and copy medium sezed file. The Inteactive (Int) class consists of user think time (60 sec) interleaved with interactive commands such as in HI class. This is explained in more detail in section 4.5.

We have tested 3 server types under these conditions. Two of the servers were IBM4381 and IBM4361. The third server was a VAX 750. All servers were running Locus. All clients were ATs running DOS. Connection was accomplished through the PCI protocol.

5.3.1.1 Communication Intensive (CI)

The sessions were transferring 1MBytes files from a disk on the Backbone to the hard disk on the PC. This was done in a continuous fashion, such that infinite capacity data transfer would result. The transfer time for a 1MByte file was used as a standard performance yardstick. In fact, we have standardized this benchmark to measure system response in other load situations (interactive and HI sessions, for example).

The results are presented as sets of 3 plots for each experiment set. As a function of the number of sessions we describe the following :

1. Transfer time for 1MByte of data (seconds)

2. Individual session throughput (KBytes/sec)

3. Global network throughput (KBytes/sec)

The IBM4381 Demonstrated the best maximum global throughput capacity, as is demonstrated in Figure 5.10. The Vax 750 demonstrated a 77 KByte/sec max throughput (Fig 5.11), and the IBM4361 supported 51 KBytes/sec (Fig 5.12).

Other important observations are that the VAX 750 had a fast rise into its maximum global throughput, whereas the IBM machines had a continuous longer rise with respect to the number of sessions. The maximum number of supported sessions was 20 for the 4381, 18 for the 750, and 10 for the 4361.

5.3.1.2 Interactive Sessions (Int)

In this set of measurements we have introduced session load that represents actual student load. The student light interactive load was simulated by the following command script:

1. list remote directory

2. get 20KByte file from server

3. wait 60 seconds (Student think time, local PC processing)

4. list remote directory

5. wait 60 seconds (Student think time, local PC processing)

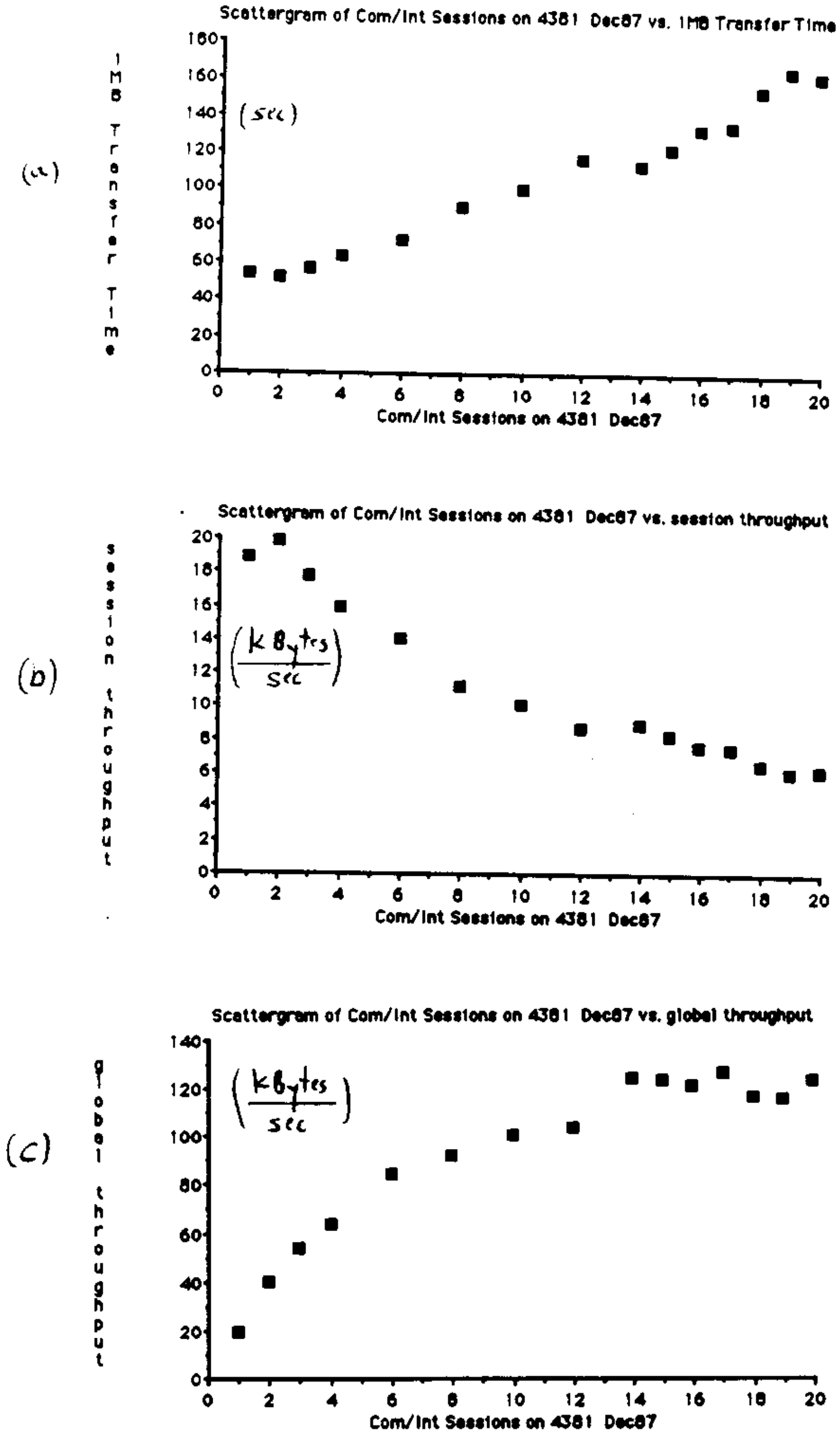


Figure 5.10: 4381 - Communication Intensive Sessions

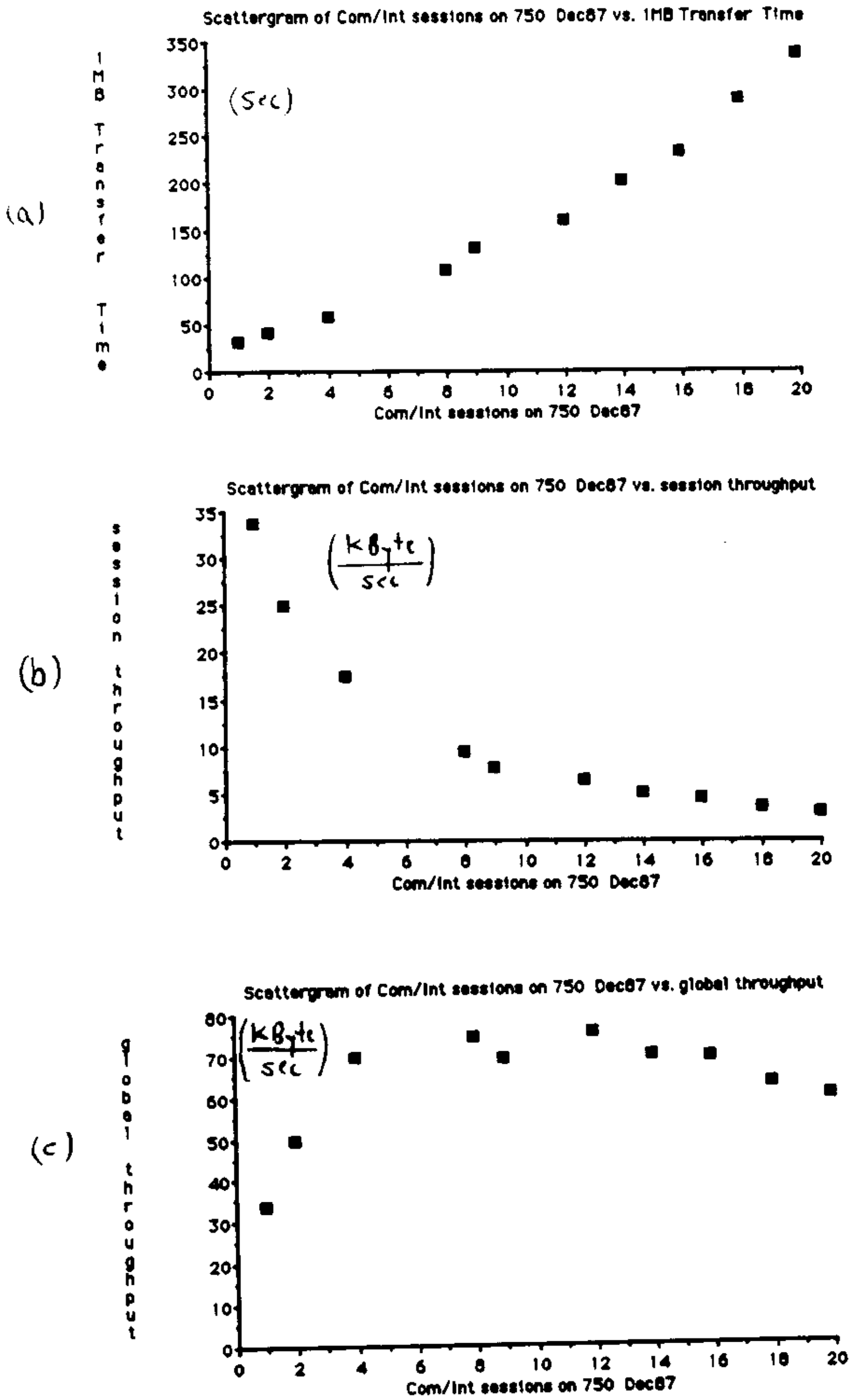


Figure 5.11: 750 - Communication Intensive Sessions

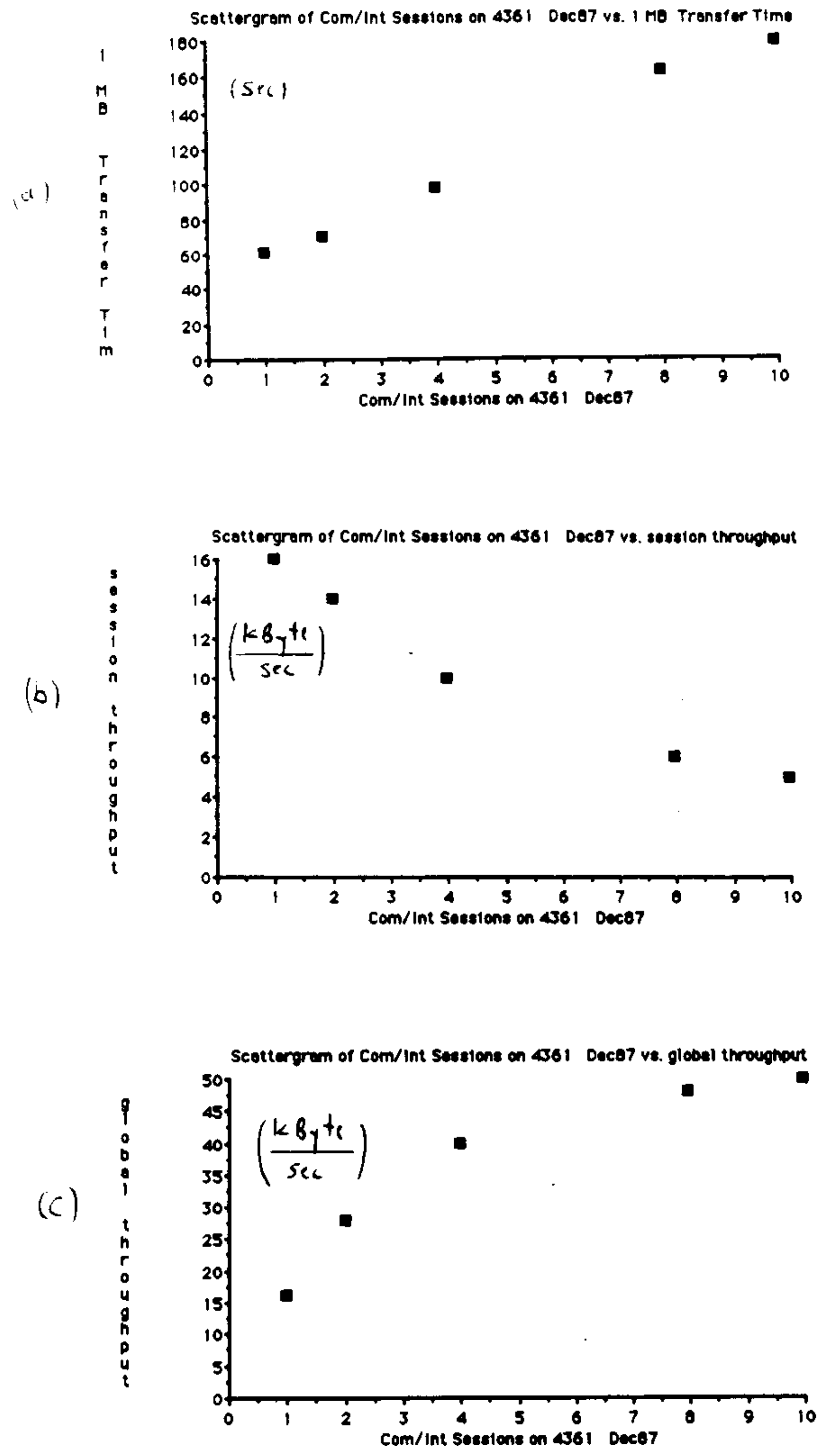


Figure 5.12: 4361 - Communication Intensive Sessions

6. send 20KByte file to server

7. list remote directory

Several sessions were operated on the machine under consideration, and a single session of the 1MByte benchmark was run to sample system performance. Clearly, the number of sessions that any particular machine could support is higher than with respect to the communication intensive case. The corresponding numbers for 4381 750 and 4361 are 38, 28, and 10 sessions. The results are presented in Figures 5.13, 5.14, and 5.15 below.

5.3.1.3 High Interactive Sessions (HI)

Experiments along the same lines were conducted after the new TCP/IP PCI software was installed. More demanding interactive tasks were composed. These tasks did not have the wait cycles, such that the central resources were heavily taxed. These highly interactive sessions exhibited no user think time, such that continued command execution resulted in. Three parameters were measured as the number of active sessions on the 4381 increased. Namely, we have measured the suite execution time, the 1MByte transfer on a single session, and the time to "cat" the password file (108KBytes) to the PC screen. Figures 5.16, 5.17, and 5.18 display these results, along with the maximum obtainable session throughput for the last two measures.

It is easily noted that the system behaves very well for less than 30 users. At

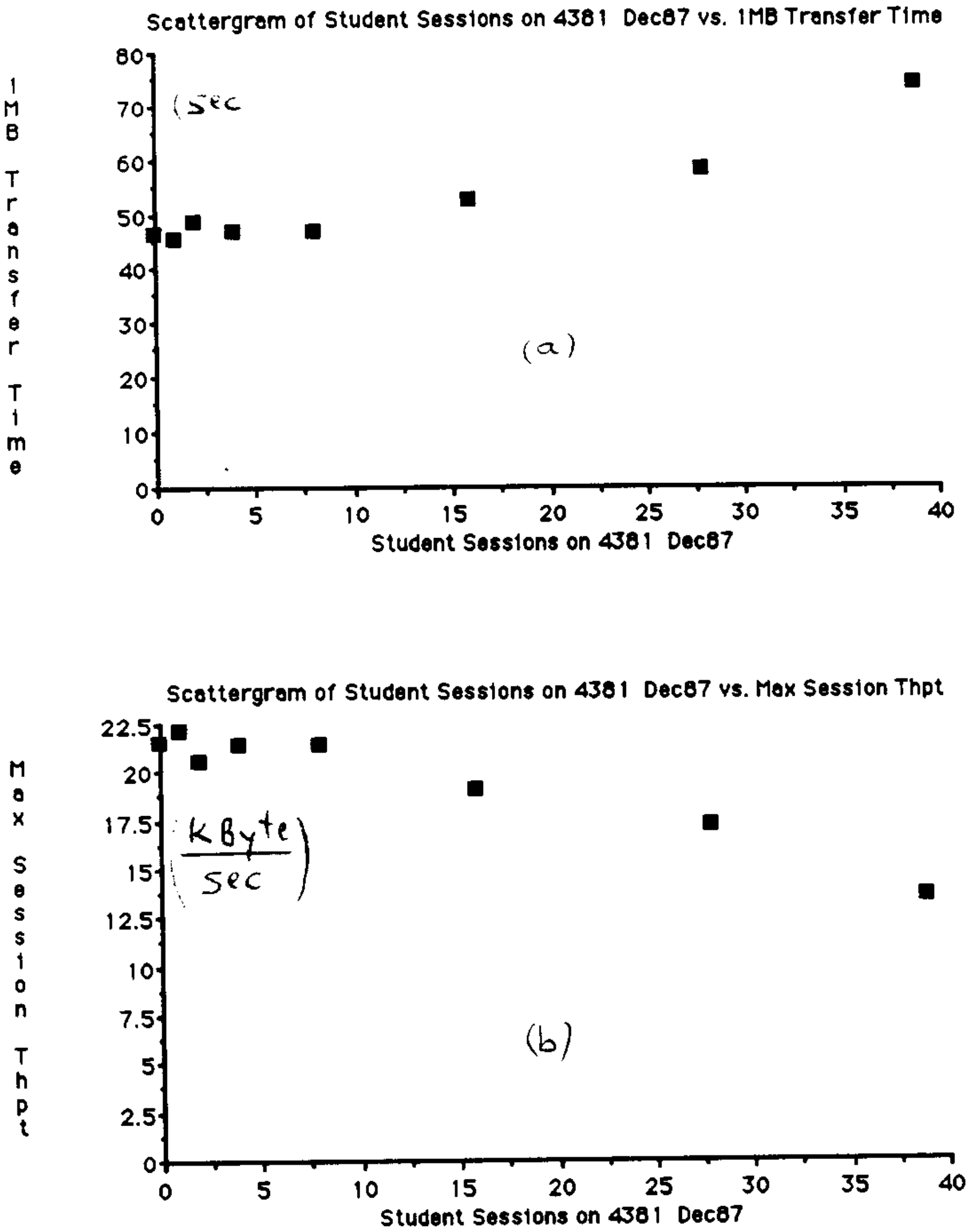


Figure 5.13: 4381 - Interactive (Int) Student Sessions - 1MB Benchmark

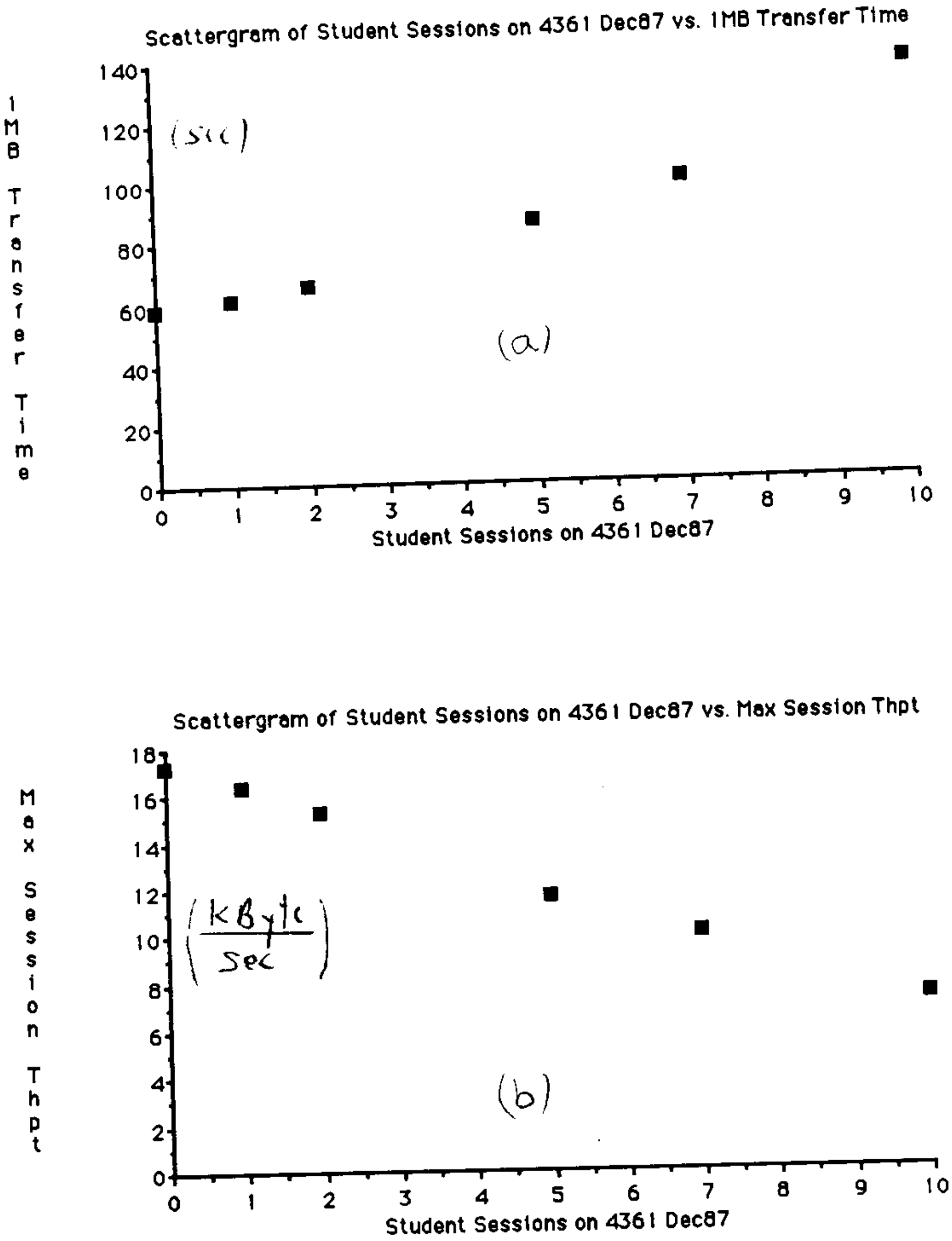


Figure 5.14: 750 - Interactive (Int) Student Sessions - 1MB Benchmark

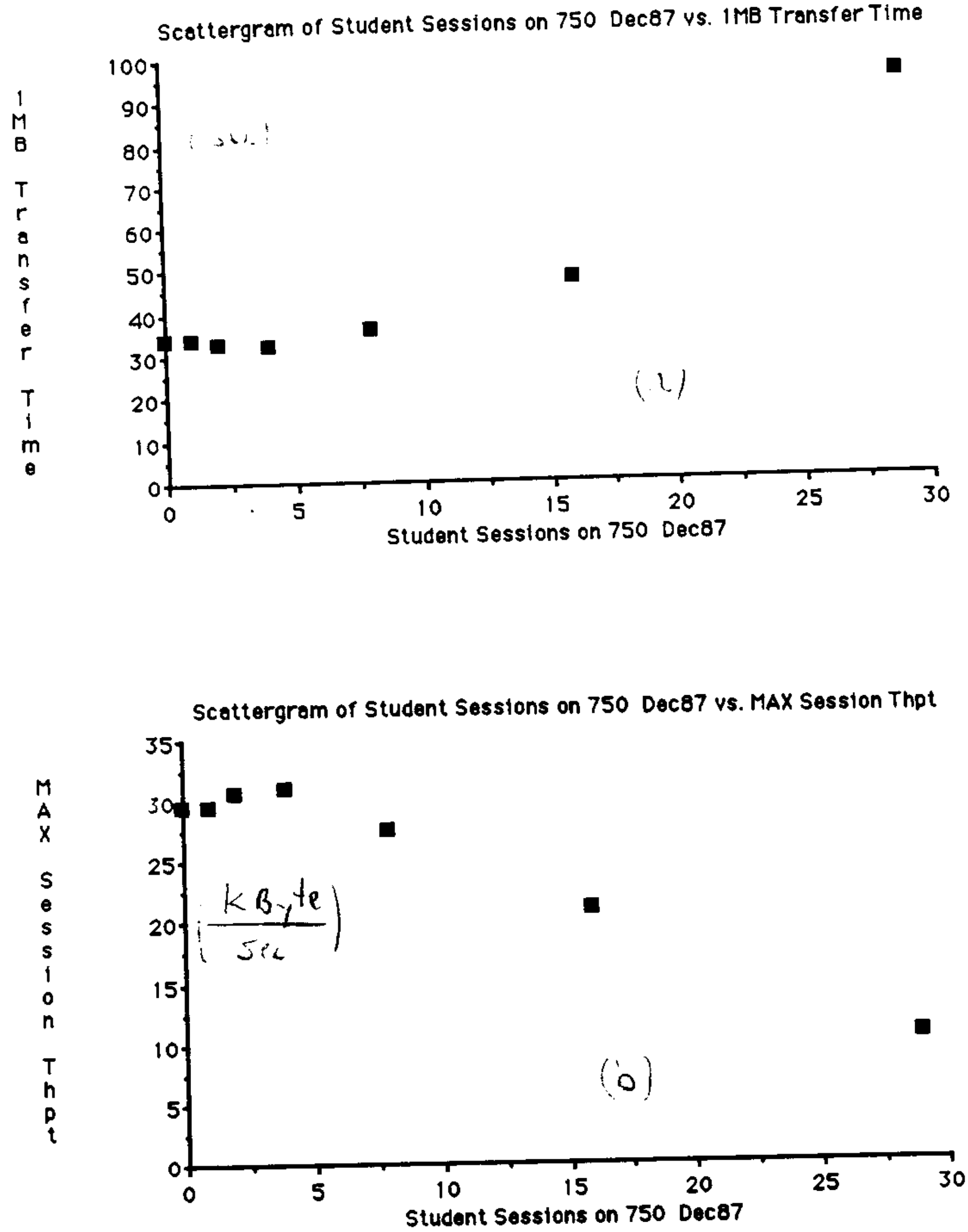


Figure 5.15: 4361 - Interactive (Int) Student Sessions - 1MB Benchmark

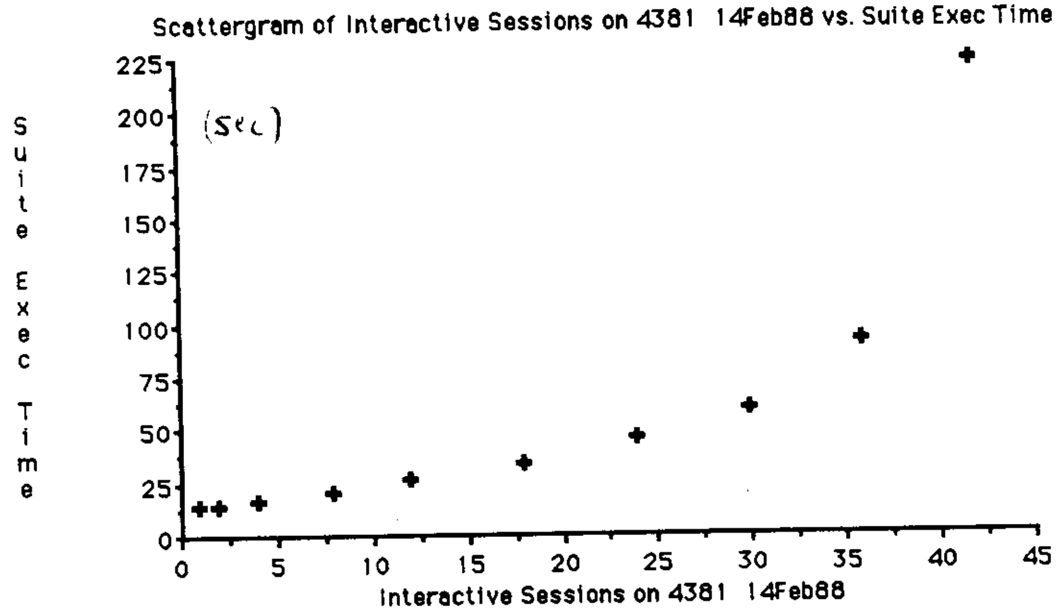


Figure 5.16: 4381 - High Interactive Sessions - Command Suite Execution Time

about 35-40 users there is a very substantial performance wall in all categories. response goes way up, available throughput diminishes. It was noted at the time of measurement, that the 4381 had a substantial number of jobs in the cpu queue. It is quite clear, that the problem results from memory contention, swapping, and thrashing, that are consuming cpu and I/O cycles.

5.3.1.4 Idle Sessions

In the most recent version of TCP/IP PCI, good support for idle sessions has been demonstrated. The IBM4381 was able to carry 60 sessions logged on simultaneously. This is in contrast with earlier versions which had a tendency to lose sessions and sometimes crash the backbone machine.

The IBM4361 and the VAX750 could support 12 and 34 idle sessions respec-

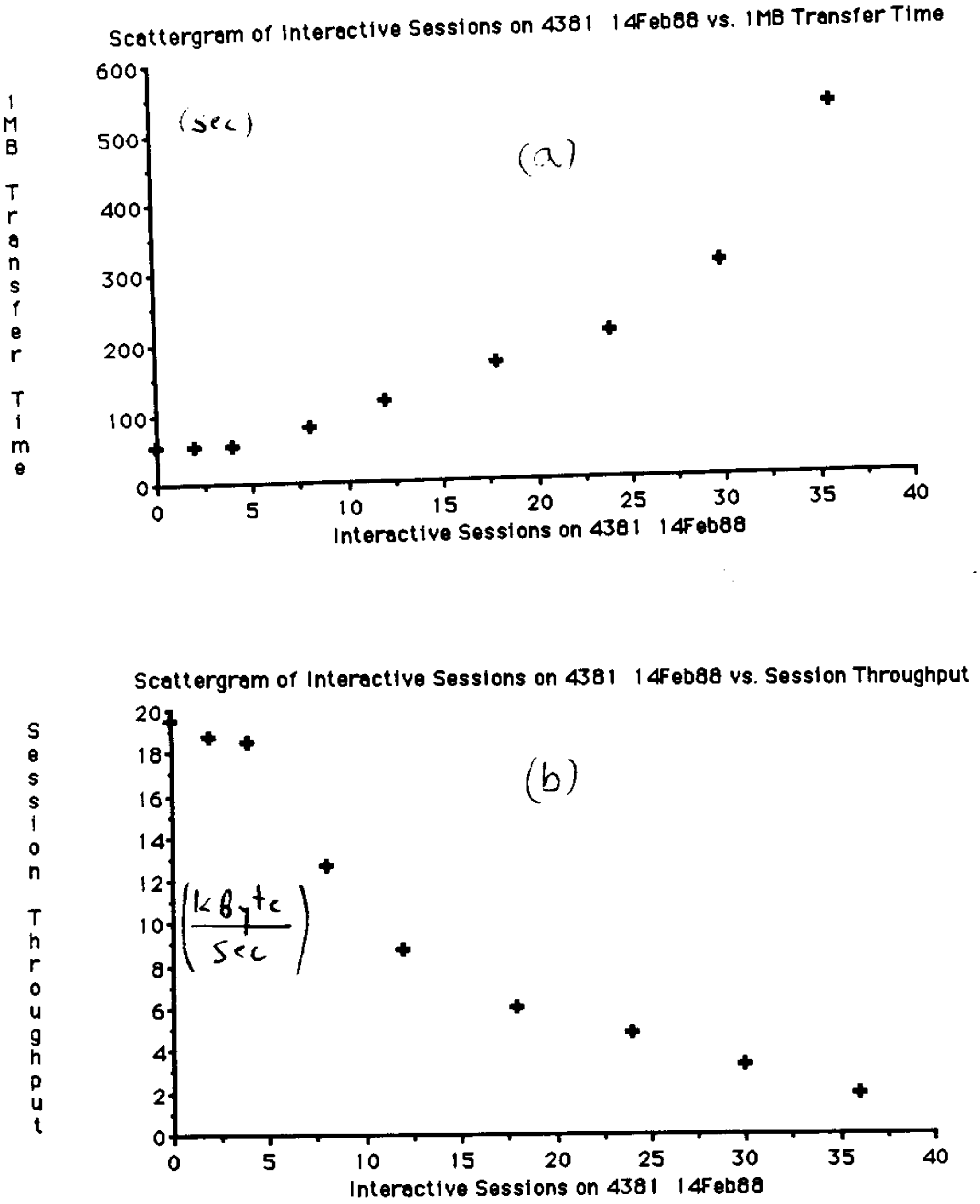


Figure 5.17: 4381 - High Interactive Sessions - 1MB Benchmark

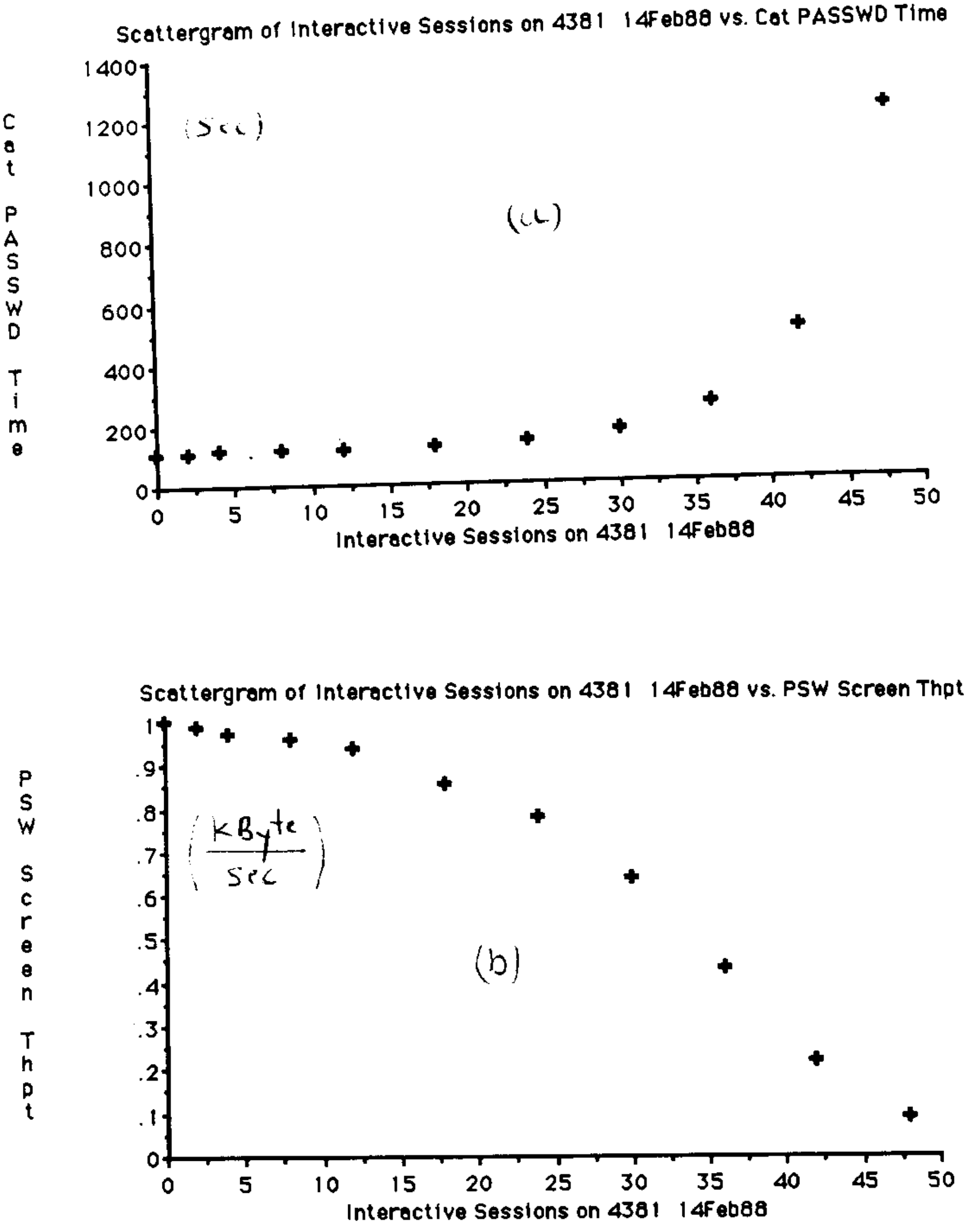


Figure 5.18: 4381 - High Interactive Sessions - Screen Throughput Benchmark

tively.

5.3.2 Loading of Several Backbone Servers

In this phase we have examined the support that could be offered by joining several backbone servers, to alleviate contention phenomena reported above. The idea is to use several less loaded backbone machines, and allocate a part of the work to those machines. In this particular example, 6 VAX 750 were used, and were each loaded with up to 10 sessions, totaling 60 sessions. We tested two cases, namely communication intensive and student session load.

5.3.2.1 Communication Intensive

In this case the objective was to find the worst case bound. The results are presented in Figure 5.19 below. It is evident that the central resources are delivering far better service. While any single machine could not support more than 20 communication intensive sessions, we easily achieved 60 sessions. In fact the total network throughput was over 240 KBytes/sec which is very impressive considering the fact that it is close to the practical limit of the Ethernet. No sessions were disconnected and no machine has crashed.

5.3.2.2 Interactive Sessions

The demands from the network/server resources were substantially reduced, as traffic requirements are very low in the commands suites which include think

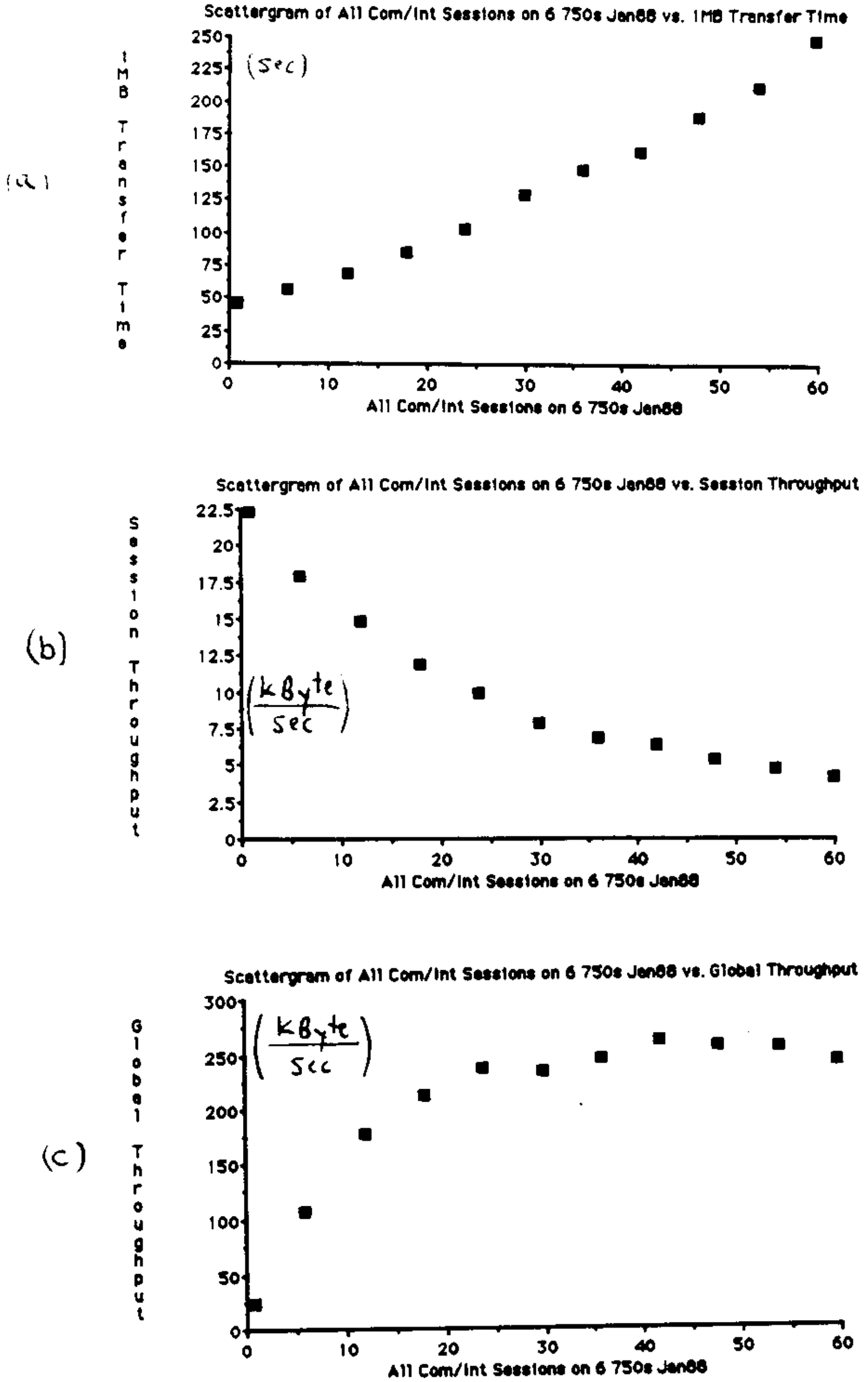


Figure 5.19: 6 x 750 - Communication Intensive Sessions Equally Distributed

time as well. The results are plotted below in Figure 5.20. Here, too, the 1Mbyte benchmark was used to measure system performance. It is evident that the backbone machines can easily support the interactive student session load without noticeable degradation in performance.

5.4 Measurements of Internal Performance Parameters

In this phase the communication overhead was monitored for the backbone servers. The communication delay for two communicating processes was measured. We present results for processes on either same local server, or communicating via Ethernet with a remote process. The results in Figure 5.21 are the *round trip* delay for a packet containing no data (pure overhead).

It is interesting to note that these pure delays indicate that the single layer of Locus on the 750 has a light weight overhead that performs better than the 4300 machines in the remote cases. for the local cases the 4381 exhibits faster process communication, as is expected due to its superior raw computing power. These communication figures explain many of the global results reported in this chapter.

We also note that for the first time we present a comparison with more recent results, measured after May 1988. These results correspond to AIX/370 release 0.0. The earlier results are designated as pre-0.0. It is apparent that the 4300 architectures have increased performance for the new version of the operating

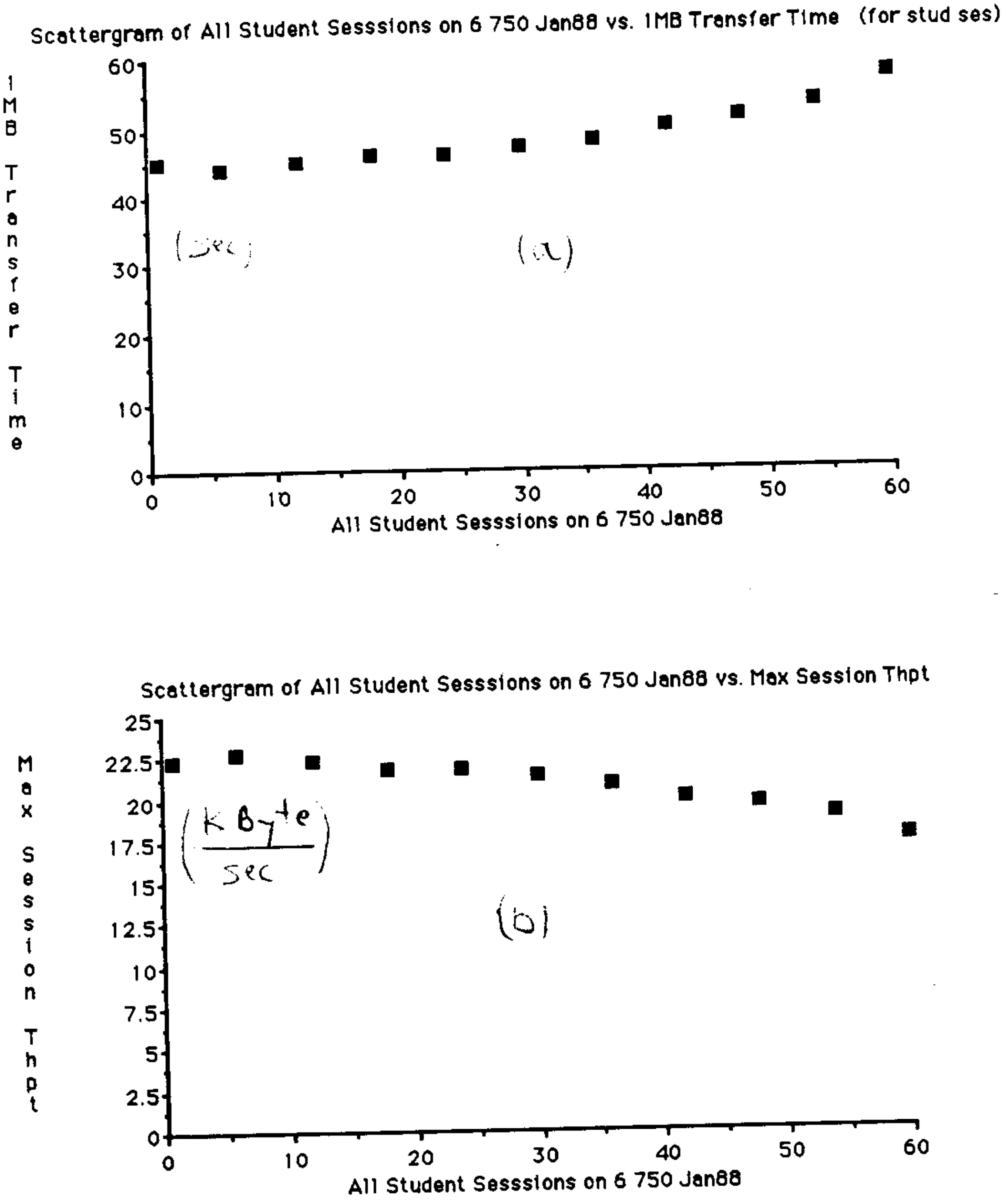


Figure 5.20: 6 x 750 - Interactive Sessions Equally Distributed

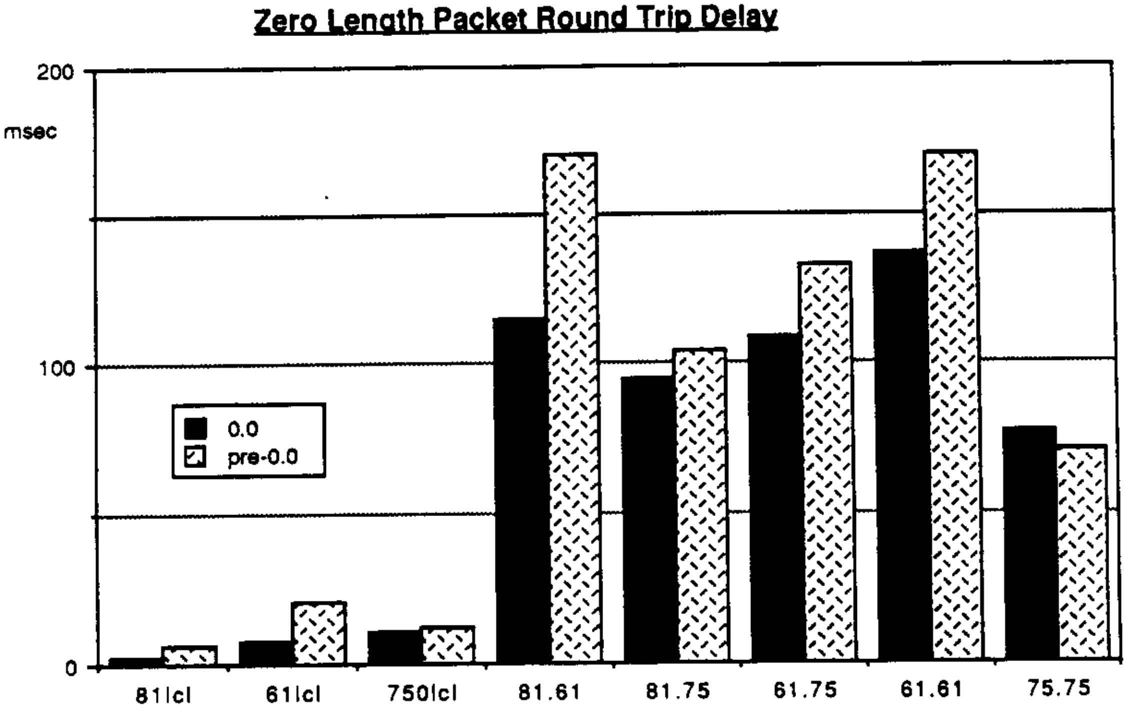


Figure 5.21: Round Trip Delay for a Packet containing no Data

system.

In the next section we will revisit additional pre-0.0 measurements and compare them to the recent AIX/370 0.0 performance.

5.5 Comparisons with a New pre-Release of AIX/370

In the process of conducting our research, the system was evolving, as it has been under intensive development effort by the developers[3] We have noticed earlier in the chapter that some changes corresponded to Ethernet interfaces that were changed early in 1988. In this section we report how the introduction of a new release affected performance figures we have measured.

5.5.1 Communication Intensive (CI)

This user class received considerable attention, as we use it as a general yardstick for performance measurements among other classes. Hence we shall look at a case of backbone to backbone CI performance, as well as the traditional PCI work.

5.5.1.1 Backbone Internal

Here we have measured the maximum throughput that was obtained, as 1MB files were transferred among various backbones. These results are presented in Figure 5.22. The results are presented in terms of cold and warm caches, as read-

[3] Locus Computing Corporation under contract from IBM

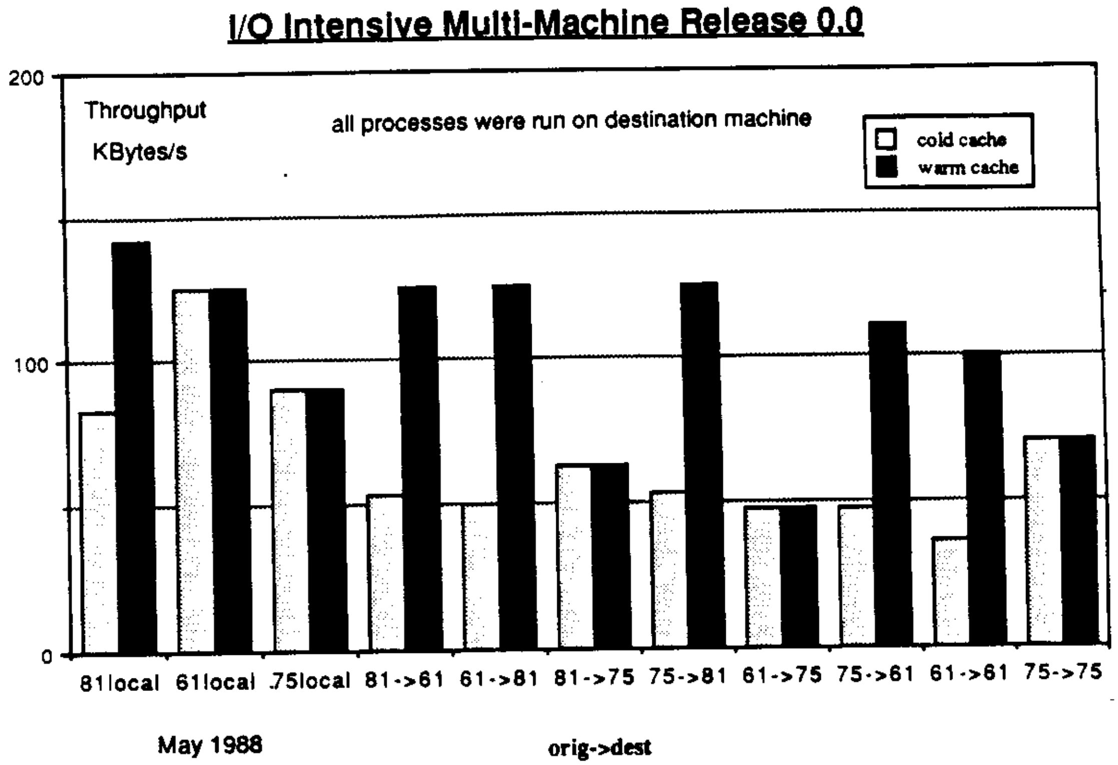


Figure 5.22: Communication Intensive Multi-Machine Release 0.0

ahead and other sequential optimizations result in better throughputs than the first time around. We note that throughputs of 50-120 KB/sec were obtained for the 4300 machines. Previous problems with hanging and exceedingly low throughputs were indeed eliminated.

5.5.1.2 PCI-to Backbone

This set of experiments presents a wide variety of comparisons among the three backbone architectures under consideration. We present here only the final throughput figures, skipping the intermediate derivations. in Figure 5.23 we include the maximum CI throughput for the 4300s and the 750. We note increased stability and capacity to support sessions on the part of the 4300. The 750 suffered decrease in performance which is attributed to the tuning of AIX to support the 4300 architectures. Figure 5.24 presents the relative performances of the three machines before and after the introduction of release 0.0. It is evident that the 4361 is now superior to the 750, as the new release supports better the 370 architecture. We conclude this section with an overlay of all 6 measurement sets (Figure 5.25), to present the relative behavior of all experiments with respect to each other.

5.5.2 Interactive (Int)

In this part of the measurement effort we study the Interactive (Int) class of users, before and after the introduction of release 0.0. The interactive class

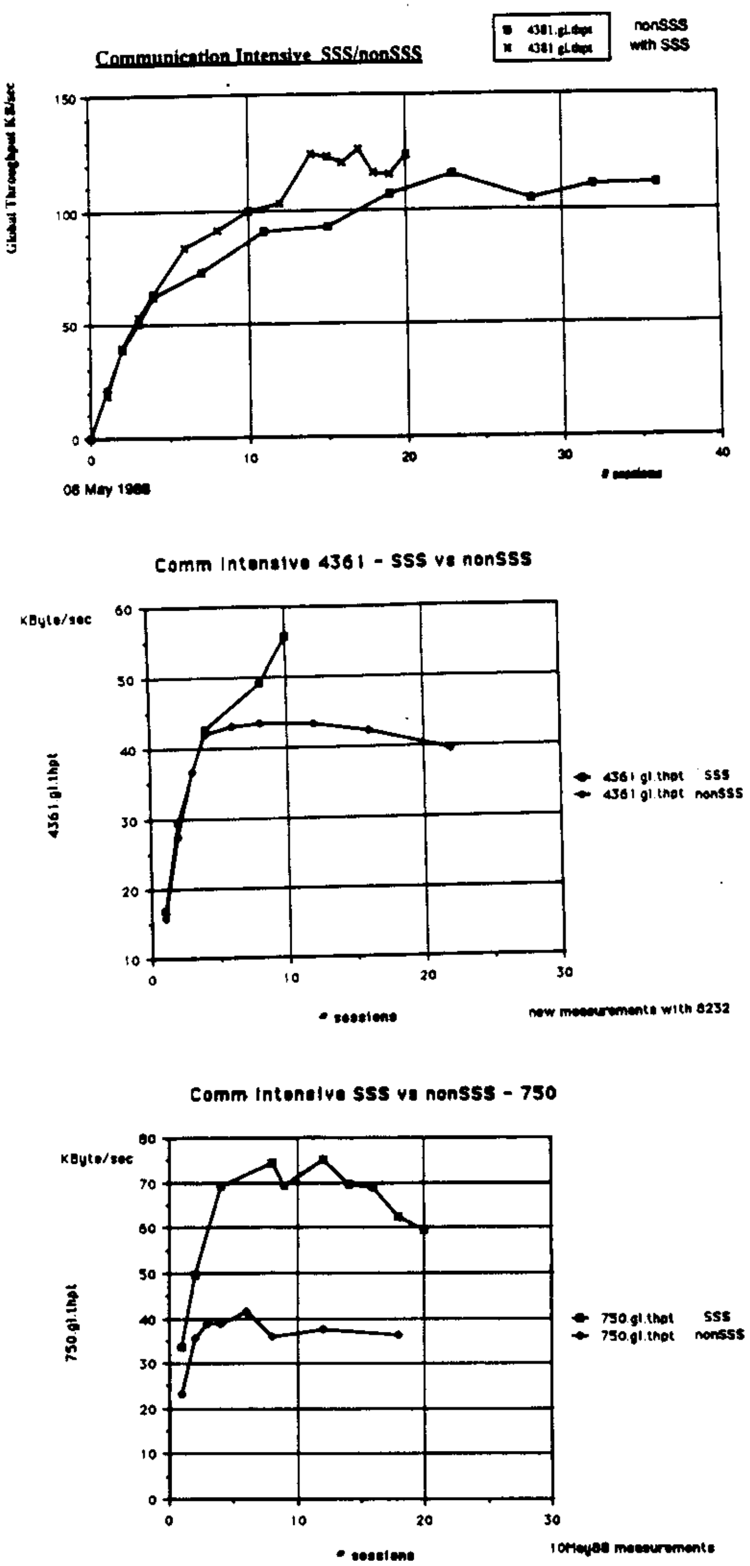


Figure 5.23: Individual Machines - Communication Intensive Multi-Machine Release 0.0 (nonSSS) and pre-0.0 (SSS)

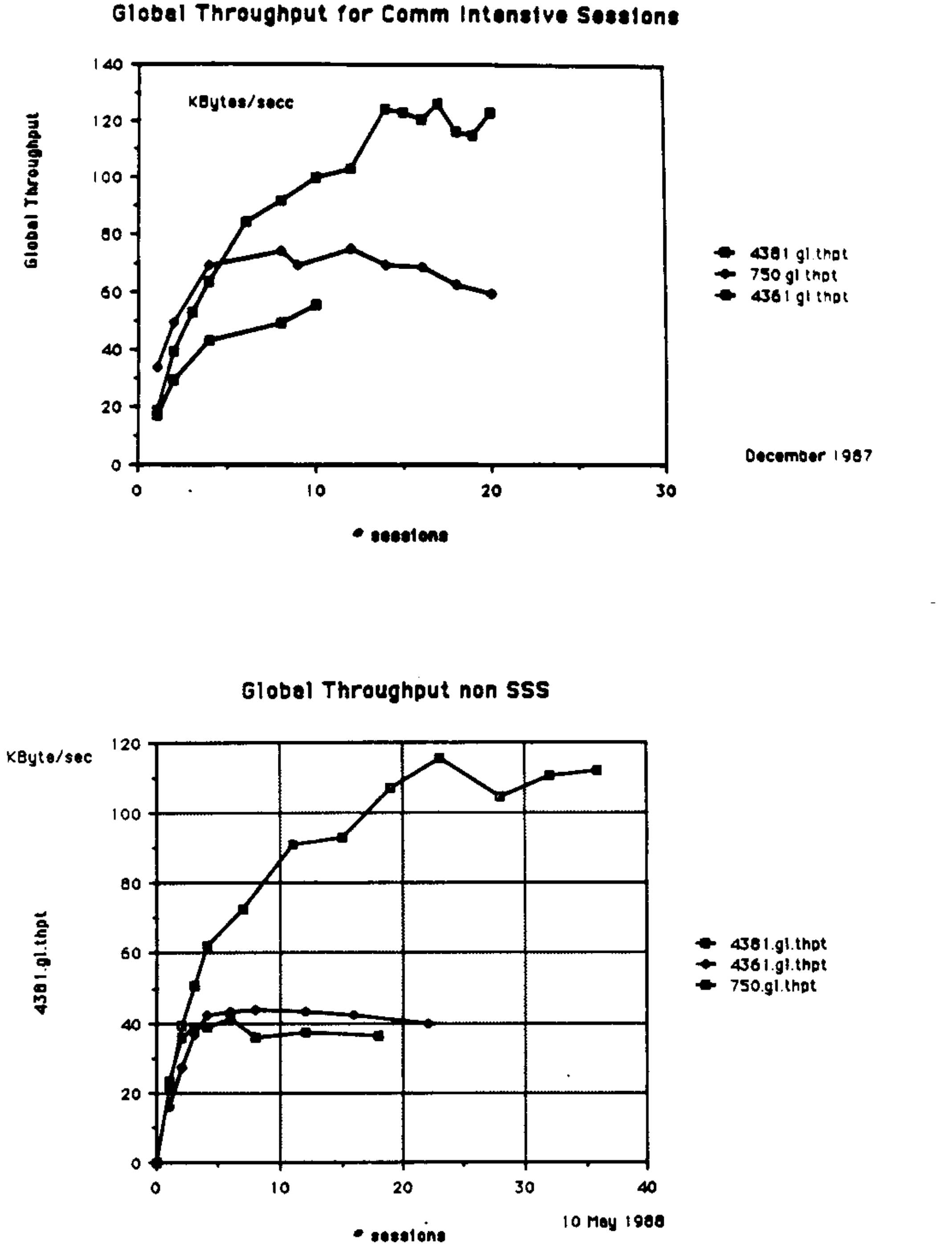


Figure 5.24: Communication Intensive Multi-Machine Release 0.0 (nonSSS) and pre-0.0 (SSS)

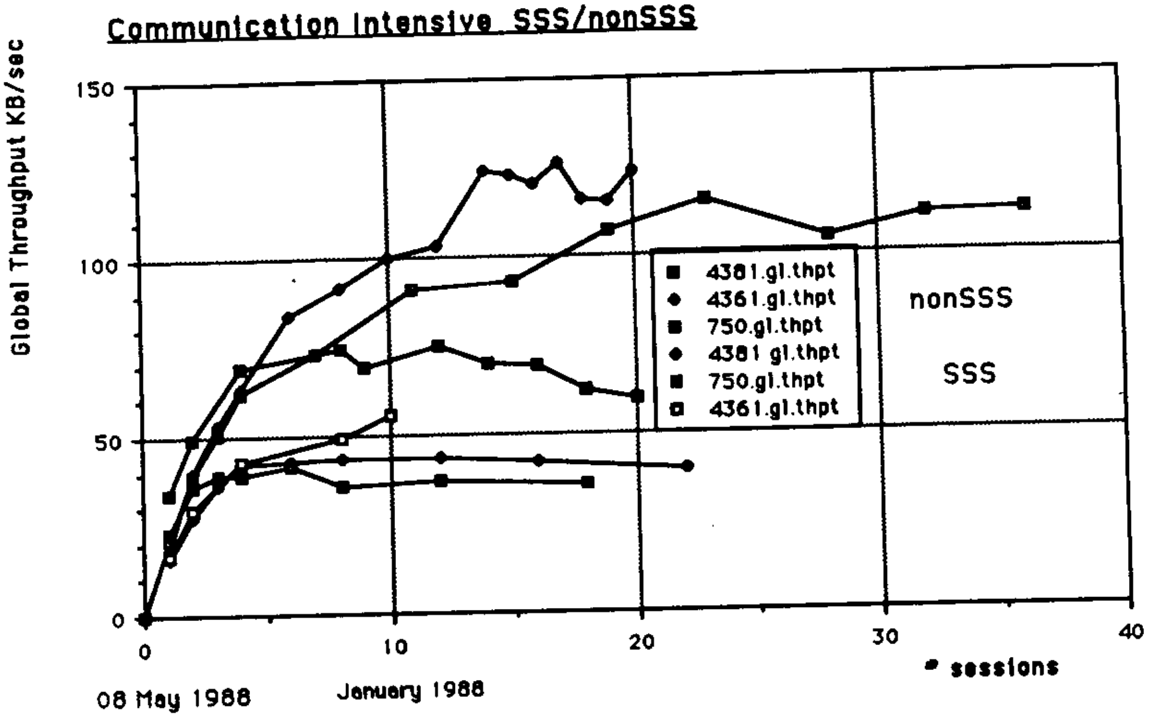


Figure 5.25: Communication Intensive Multi-Machine Release 0.0 (nonSSS) and pre-0.0 (SSS) - Overlay

is a very typical as far as SEASnet users are concerned. The metric we employ here is the delay incurred in a 1MB transfer - our global yardstick. We present in Figure 5.26 the comparisons for all three machines. We note that the 4300 exhibit increased stability at high session numbers. The 750 performance has diminished, similarly to the behavior of the CI sessions in the previous subsection. For completeness, we present in Figure 5.27 the overlay of all interactive session experiments.

5.5.3 High Interactive (HI)

This middle-ground class represents an over-zealous interactive user, that stretches the system to some bounds, less than CI and more than Int. We shall present here comparison results for the major PCI server, the 4381. Figure 5.28 demonstrates a comparison between pre0.0 and 0.0 performance. We note that similar to the observations from class (Int), we found better overall stability as well as multiple session support for a higher number of sessions.

5.5.3.1 HI with CPU Loading at Backbone

Another recent measurement that we performed with respect to HI sessions was a combination of High Interactive multiple PCI sessions, concurrently with backbone loading with non-PCI CPU Intensive jobs. The purpose of such an experiment is to learn about the cpu demands of the HI job class. We present in Figure 5.29 the suite execution time with an unloaded CPU, and with a cpu with

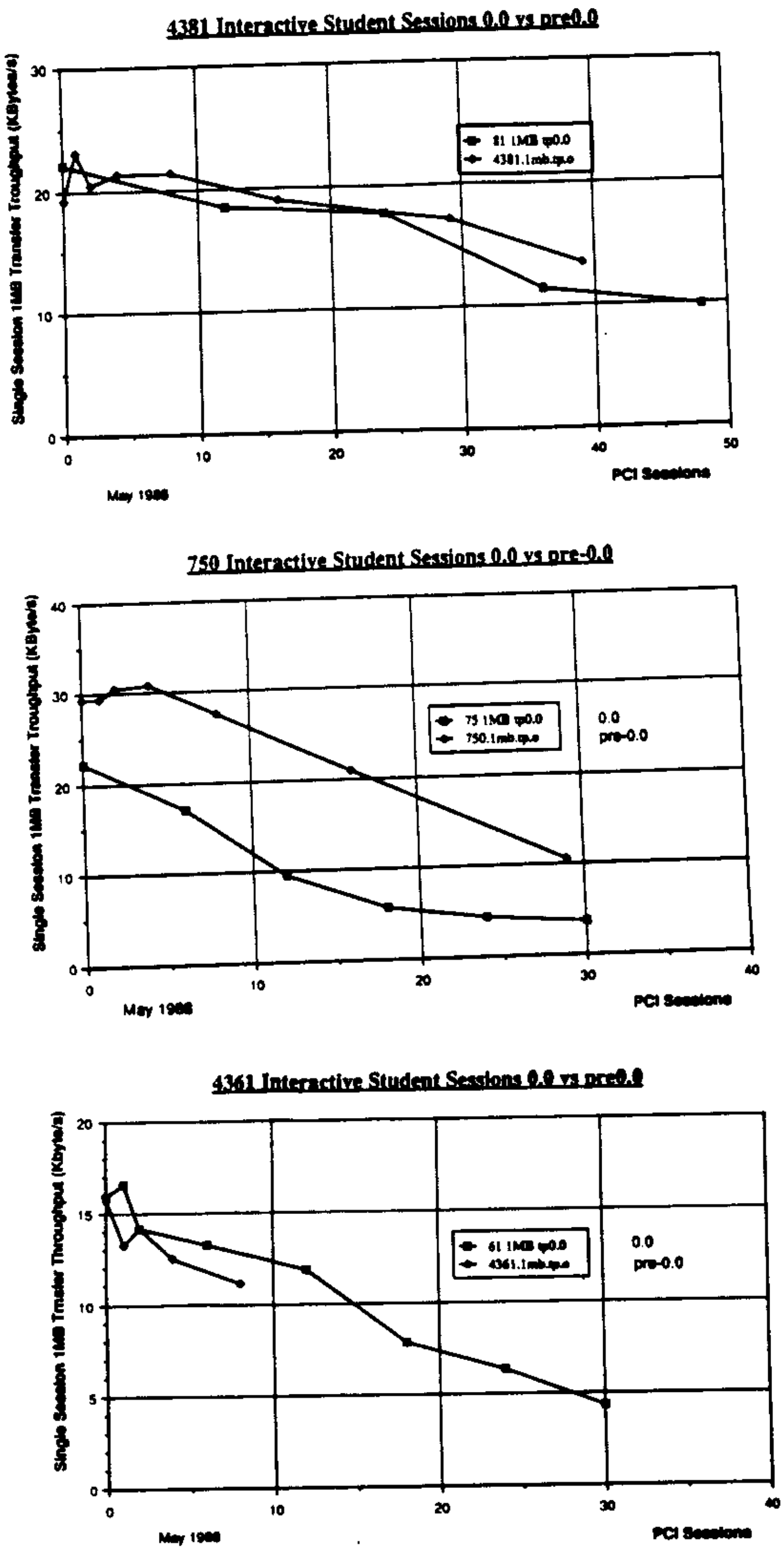


Figure 5.26: Interactive (Int) Multi-Machine Release 0.0 (nonSSS) and pre-0.0 (SSS)

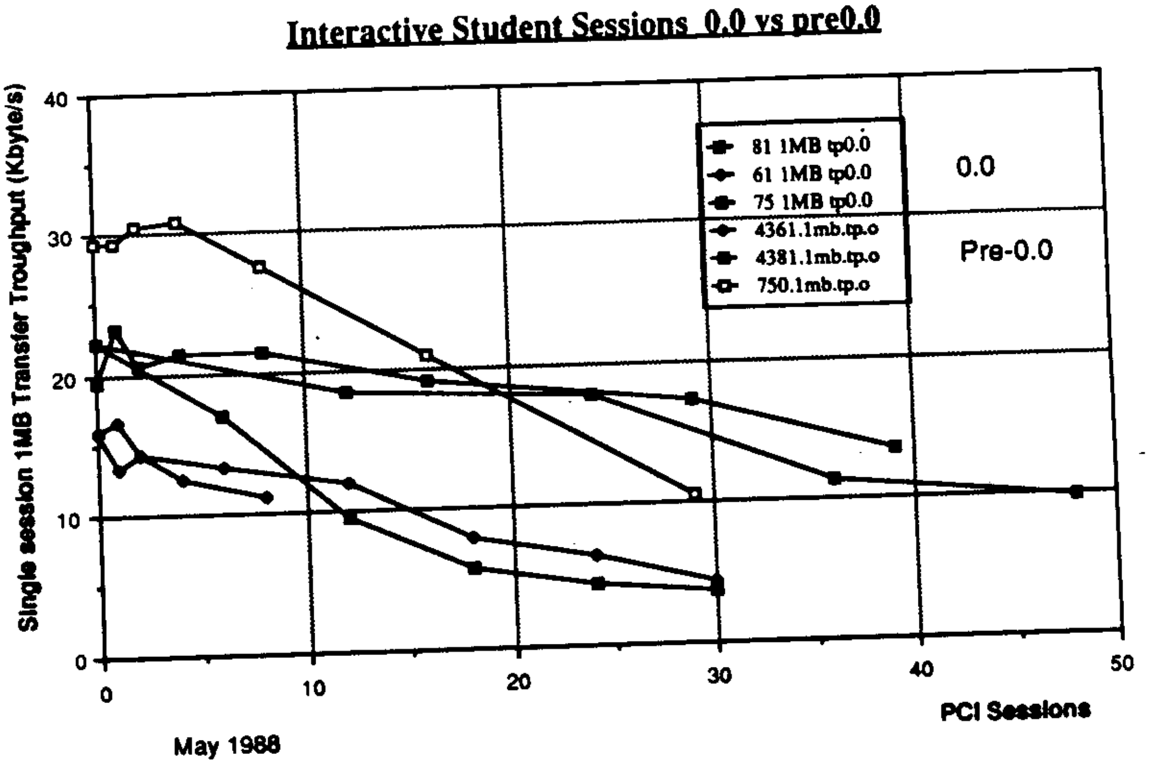


Figure 5.27: Interactive (Int) Multi-Machine Release 0.0 (nonSSS) and pre-0.0 (SSS) - Overlay

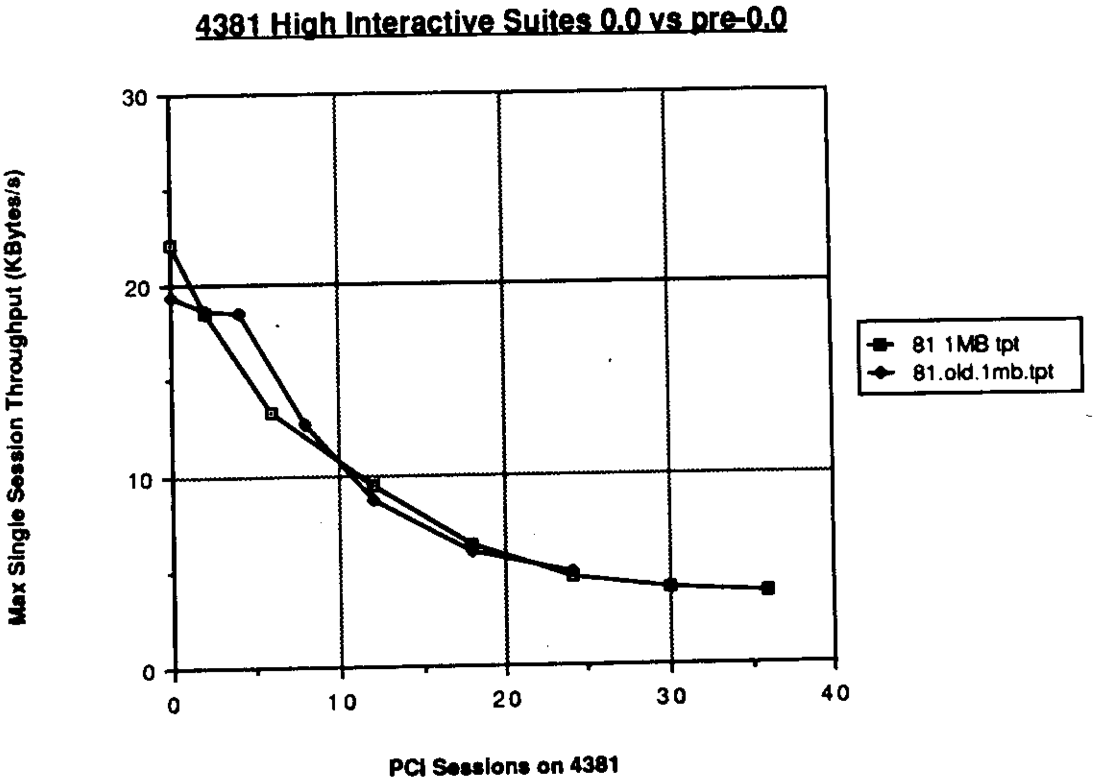


Figure 5.28: High Interactive (HI) Release 0.0 (nonSSS) and pre-0.0 (SSS) - 4381

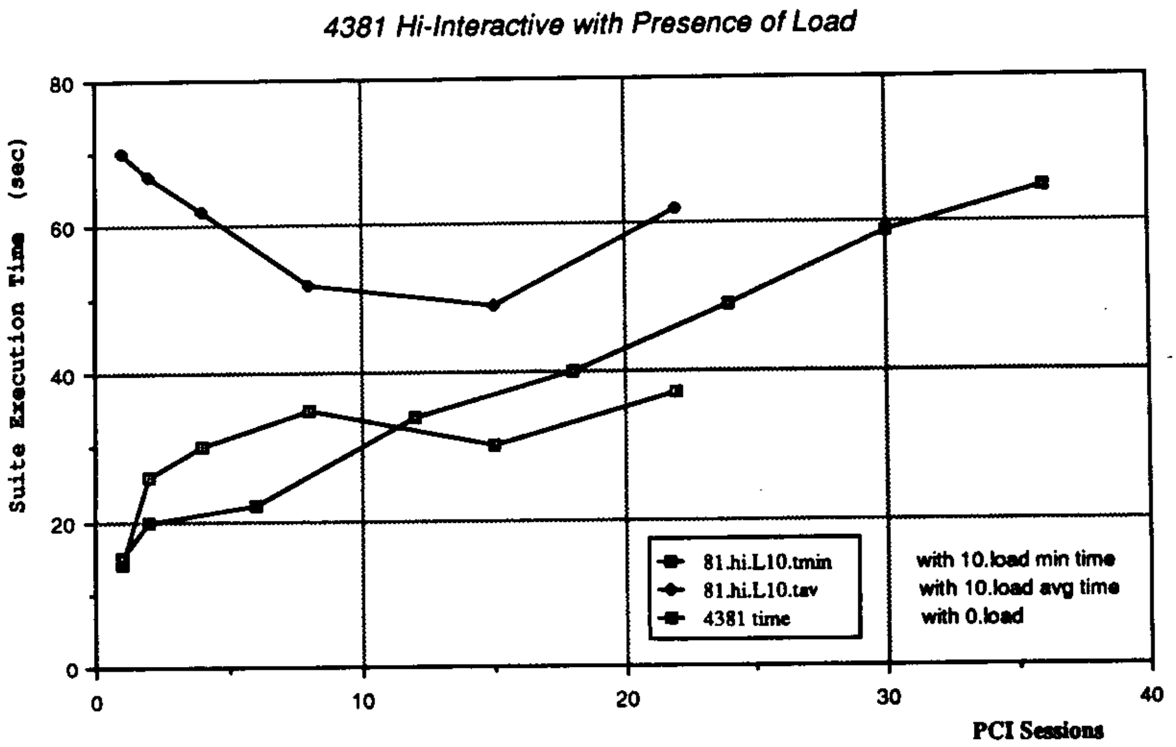


Figure 5.29: High Interactive (HI) Release 0.0 Effects of CPU Loading - 4381

a load factor of 10 (10 jobs in the ready queue). We note that the high cpu load has a substantial effect on suite execution, with a high variance introduced to that time. This indeed demonstrates that there exists system overhead consisting of significant instruction execution by the cpu. This will be reflected in the observations and modeling efforts that will follow.

5.6 Performance Observations

Keeping the heterogeneous systems focus in mind, the array of measurements reported in this chapter, conveyed to us the necessary intuitive feel for the systems under consideration. We have identified the major contention sources, and the way in which they affect service to the user of the system. Let us elaborate on some of the important observations :

5.6.1 Process Space Contention

This is a very important phenomenon for any active session. Each connected session has a PCI-Server module active on the backbone server. This process is about 110 KBytes in size. Whenever there is any communication activity, this process needs the memory space to function. If there is no space available, another PCI-server will be swapped out to make space. Given that our available memory is about 4 MBytes on the 4381, this means that only 35 sessions or so could be supported simultaneously in an effective manner. This is in agreement with measured results (Figs 5.16, 5.17, and 5.18), and will be a key factor to consider in our modeling effort.

5.6.2 Layered O/S Overhead

The Locus operating system was integrated onto IBM Hardware in an incremental fashion. Many of the lower level functions are still handled by CP-SYS. which is the low level Control Program that VM370 uses. On top of VM370 there is the SSS Layer. Only then comes LOCUS, on top of SSS. This Layered structure causes many interrupts to be spawned, whenever O/S functions such as I/O or communication are requested. This means that processes have to wait longer for completion of O/S functions. During that time Processor utilization might be low, as time is spent in idle waits for an interrupt to complete. This is demonstrated best in The Communication Intensive results for the 4381 in Figure 5.10 above. We note that the Global throughput rise is flatter than that of the 750. Only when several sessions are connected does the global throughput reach a maximum. This is due to the fact that when several processes are around, there is higher readiness of data in buffers, and there is some process that can consume I/O and communication data most of the time.

We note that the process communication times clearly indicate that the layered O/S results in significant communication overhead. Indeed we have seen in the course of this work that the SSS layer was eliminated, which drastically improved performance for AIX/370.

5.7 PC Ethernet-Interface Limit

This is a factor introduced on the PCs end. Our equipment is fitted with a 3Com Interface board, through which all Ethernet communication is done. This hardware exhibits a bottleneck of roughly 25 KBytes/sec. This is, differently said, the maximum throughput for a single session. This limit can be approached by intensive file transfer, or intensive communication embedded in an application. Routine command execution on the network disk or file listings to the screen amount to only a fraction of this available throughput.

5.7.1 Backbone Server I/O Speed

This factor is influenced more than anything else by the layered O/S structure eluded to above. We note that even local I/O speed does not exceed 120 KBytes/sec on a 4381, which is less than the raw performance of the 3380 disk drive.

5.7.2 Ethernet Contention

Rarely could we approach an Ethernet contention situation. That situation has occurred when several backbone servers were pumping data into communication intensive sessions. In this case a measured net traffic of 240KBytes/sec was present. This extreme case tells us that as a first low traffic approximation, we might fare well by considering the Ethernet to be transparent.

5.7.3 Backbone Communication Speed

The backbone communication speed was shown to be dependent primarily on Operating System architecture (Fig 5.21). In fact, a light-weight O/S on a slower processor performs faster than a fast processor that is covered by a multitude of O/S layers. The numbers obtained for communication performance, would be valuable in constructing the details of the queueing network model.

5.7.4 Backbone Processor Speed

This turns out to be a contributing factor in the global performance picture when considered together with the system overhead. The main reason being that raw processing power is an upper bound that the user could hope to get. The services provided by the operating system that make the machine useful to the user do not come for free. This cost is substantial, and our interest is mostly in the net performance delivered to the user.

5.7.5 Reliability

Reliability of the heterogeneous system considered here has improved substantially with the development of the operating system support. At first there was a noticeable tendency of sessions to timeout and disconnect from the server. Other times the backbone server would crash altogether. This undesirable behavior improved significantly with newer versions of the system and PCI. in 1988 we have experienced more graceful degradation under heavy load for the 4300 machines. It should be noted that user confidence in the system is directly related to the reliability and performance as perceived by the user. The user will demand more of a reliable performing system and vice versa. One of the main modeling goals is to guarantee that load is prevented from increasing to a point where system stability is questionable.

5.7.6 Impact on Modeling Phase

The performance measurements conducted helped us focus on the most significant parameters of the heterogeneous system. We are ready to develop a model that will emphasize first order effects.

Only once coarse grain results are satisfactory, can subparts of the model be refined, and more accurate description of the system may be obtained.

We will first model the most significant sources of contention, namely the layered operating system, CPU and I/O overheads, and process space contention for PCI-Servers on the backbone machines. The PC customers will be modeled by their Ethernet interface bottleneck. Other measured effects might be added in further refined models.

5.8 Summary

We have conducted two sets of measurements on two distinct architectures. Both architectures were heterogeneous in terms of hardware, as they consisted of PCs and backbone servers. However, the first system was running homogeneous operating systems - Locus for all machines. The measurements on this system gave us some indication of the performance of an AT running Locus, and led us to the target and focus of our study - heterogeneity. Indeed, the second system - SEASnet - consisted of heterogeneous operating system, Locus and DOS, as well as heterogeneous hardware. This system will draw most of our attention in the future chapters of this dissertation.

The second set of measurements was used to profile the heterogeneous system under investigation. We have also acquired a good intuition for the governing parameters of SEASnet, and we are ready to pursue the next logical step of the research.

We note that the measurement base that was provided can serve as a basis for many other modeling efforts. Indeed, we have set out to produce a wide set of measurement results that can seed other research efforts in the future, In addition to using many of the results presented in the remainder of the dissertation.

5.8.1 Correlating Experimentation with Analysis and Modeling

The next logical step was to enter an iterative cycle of correlation and inference drawing from results with respect to model predictions. We then modify our model and conduct an iterative research cycle to achieve good convergence. We make notice here that chronologically these steps were overlapping to some extent, such that the iterative modeling effort recieved continuous feedback from our measurement effort. Since this task was closely knit with the construction of the model, the accuracy of the model was successively improved. We note that our ultimate goal is the demonstration of predictive capacity accurate enough to perform system configuration synthesis.

CHAPTER 6

Model Construction and Validation

In this chapter we shall gather the user parameters we have obtained, along with system parameters we have available, and construct a model for SEASnet. The goal is to demonstrate consistency between the model predictions and the actual system behavior. This task is difficult, based on the fact that the information we have on the system being modeled is not complete. We nevertheless demonstrate first-order compatibility for the level of granularity under consideration.

6.1 The Governing Factors

The system under study, namely SEASnet, is complex, and presents a large degree of heterogeneity. This is the first attempt to model a SEASnet-family distributed system to date. Subparts of the Locus[1] distributed system were modeled in the past [GolLavPop83], [BetGerPop84]. However, our present work involves the concepts of heterogeneous node architectures as well as heterogeneous operating systems. In order to get a handle on such a complex set of issues, we reduce the problem to the minimum bare-bone size set of assumptions and parameters,

[1] Locus is the predecessor to AIX/370. It was originally developed on a homogeneous environment, running exclusively on DEC equipment (PDP-11 or VAX 750)

that still describe the behavior of interest in an acceptable fashion.

In the PC-Interface configurations, we consider the backbone clusters, as giant file servers. Since that is their primary designation, we chose to model factors which are crucial in that capacity. for example, we consider the file service rate a primary factor in the modeling scheme. These rates were determined via measurement reported earlier for the various servers under consideration.

6.2 Available Modeling Techniques

The queueing network is a natural way to describe a network of computation, communication and I/O resources. There can be several ways to map a specific system onto a queueing network. Additionally, once a model is established, there could be several ways to go about solving for performance parameters. We elaborate on this further when we discuss related research in chapter 8. However, we provide here the necessary context to facilitate following our work.

It is an important model design problem to establish a queueing network that represents the system under study in a way acceptable to the modeler. It is important to identify the principal acting forces within the system, and represent them as servers, with queueing disciplines and service rates that match the original system. Second order effects are less important, and might be incorporated in later, refined, models. We shall describe our particular modeling effort later in this chapter.

The solution of a queueing network model consists of several alternatives.

They are analytical methods, and simulation methods.

Analytical methods require that the system adhere to a set of restrictions and assumption. For example, the network should be product form for some analyses. Other assumptions with respect to arrival/service distributions might apply, also depending on the analytic method applied.

Analytical methods are divided into several subset methods. The method could be exact, i.e. the solution obtains the precise solution for the system. On the other hand we could be dealing with approximate methods. This is done when exact solutions are impossible, or computationally impractical. Simplifying approximations are made, and a more tractable solution is obtained. One should be aware that that solution deviates from the exact solution. It is nice to be able to bound the resulting deviation.

Another modeling enhancement involves decomposition. This is a way to redefine a complex model into several simpler sub-models. This results in more tractable sub-solutions. From these sub-solutions the solution for the original model can be recomposed.

Simulation solutions have the distinct advantage that they are much less restrictive. One can incorporate non-product form networks, with much more general customer and server behavior. Solutions can be obtained, without reliance on approximations or computationally complex analyses. Accuracy can be bounded by a confidence interval as desired.

The price for this convenience is in the computational cost of the simulation.

When the system becomes more complex, and required the accuracy is high, simulation costs increase rapidly. Since discrete event simulation uses event generators to simulate customer and service actions in the network, one has to repeat these random events many times over to obtain the required avarage performance values at the requested confidence intervals. We shall see below that we chose to use this technique to evaluate our model, since our model is relatively simple, and simulation was the most straight forward way to approach it.

6.3 Our Modeling Approach

We elected to simulate SEASnet as a queueing network. This choice makes it possible for us to employ tools, techniques, and modeling packages for the routine portion of the work. The difficult and creative part is to construct a good model, based on important behavior observations and assumptions we make.

The model should capture the key system behavior and be as simple as possible at the same time. Hence, we have the 3 classes of jobs we have defined in the user model. These are the Communication Intensive (CI), High Interactive (HI), and Interactive (Int) classes. These classes will define the chains of interest for the model. The service centers which are most crucial and most contended for will be the necessary servers that the model shall employ.

In constructing the user chains through the model, we will use knowledge we have regarding the communication protocols the PCI is using. Again, in the construction of this key queueing network, we shall emphasize as much knowledge

Server	*I/O Rate [KByte/sec]*
4381	120
4361	120
750	80

Table 6.1: I/O Service Rates for the Various Backbone Servers

that is obtainable on the system, and strive for as accurate a representation as feasible.

6.4 Resource Contention and Bound Determination

In our analysis, we emphasized the resources which had the highest degree of contention, since the bottlenecked areas of a system are the limiting performance factors. The areas of immediate interest are the cpu servers and the I/O devices at the backbone servers. These servers carry a crucial load during file service.

Figure 6.1 shows us the core SEASnet architecture of interest. The backbone server network serves the entire PC user community, and provides file service via PCI. The measured maximum file service rate for the particular servers are presented in Table 6.1.

Cpu rates were determined by cpu intensive measurements, as well as manufacturer's specifications. The values used are shown in Table 6.2.

The Ethernet card at the PC end was clocked at 25 KBytes/sec, and was a noticeable limit when only a few sessions were operational.

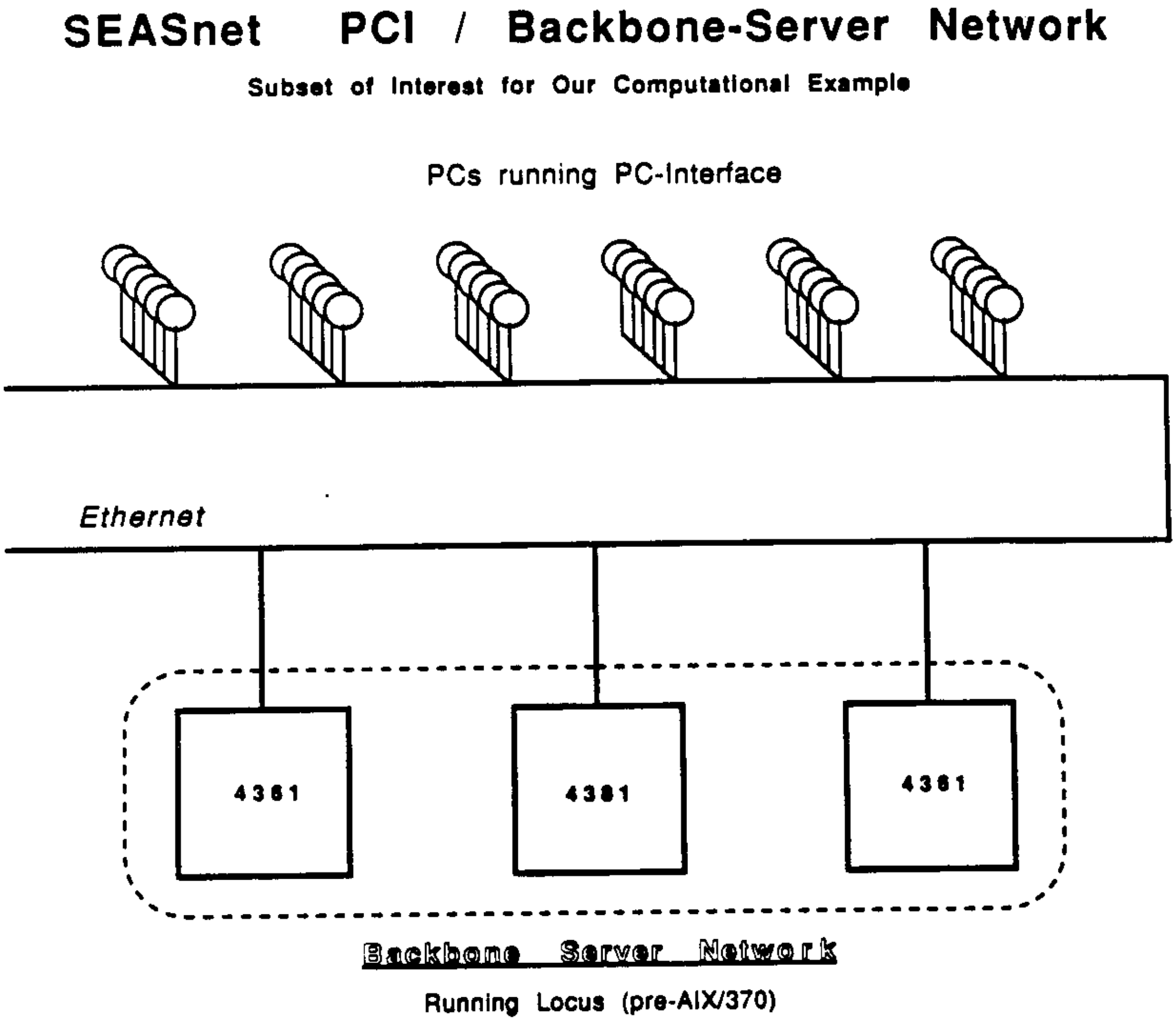

SEASnet PCI / Backbone-Server Network
Subset of Interest for Our Computational Example
PCs running PC-Interface
Ethernet
4361
4381
4361
Backbone Server Network
Running Locus (pre-AIX/370)

Figure 6.1: SEASnet - PCI / Backbone-Server Network

Server	*Rate [MIPS]*
4381	6.0
4361	2.5
750	0.8

Table 6.2: CPU Rates for the Various Backbone Servers

The Ethernet itself was not a limiting resource, as 10 Mbit/Sec is an ample throughput with respect to the 3 resources mentioned above.

6.5 Construction of the Performance Model

Simulation was used throughout the model construction and evaluation phase. The reason being that we wanted to be able to compute non-product form models, exhibiting several customer classes (CI, HI, Int).

To compute the key behavior of our system, we constructed the queueing network shown in Figure 6.2. This model encapsulates all the major contended-for resources that were presented above. A variable number of PCs is presented. This number corresponds to the number of active PCI sessions, and is also the number of jobs (customers) in the closed chain queueing network presented.

Communication between the PCs and the backbone takes place on the Ethernet, via Ethernet packets which are 1KB long for file transfer applications. In a typical file transfer, the PC will request the next packet, and the backbone will respond by sending the next packet over the Ethernet to the PC.

The operation of the backbone server consists of three sub-operations :

1. Receive request from PC

2. Verify page in memory (fault to I/O read if not in buffer)

3. Send page packet to PC

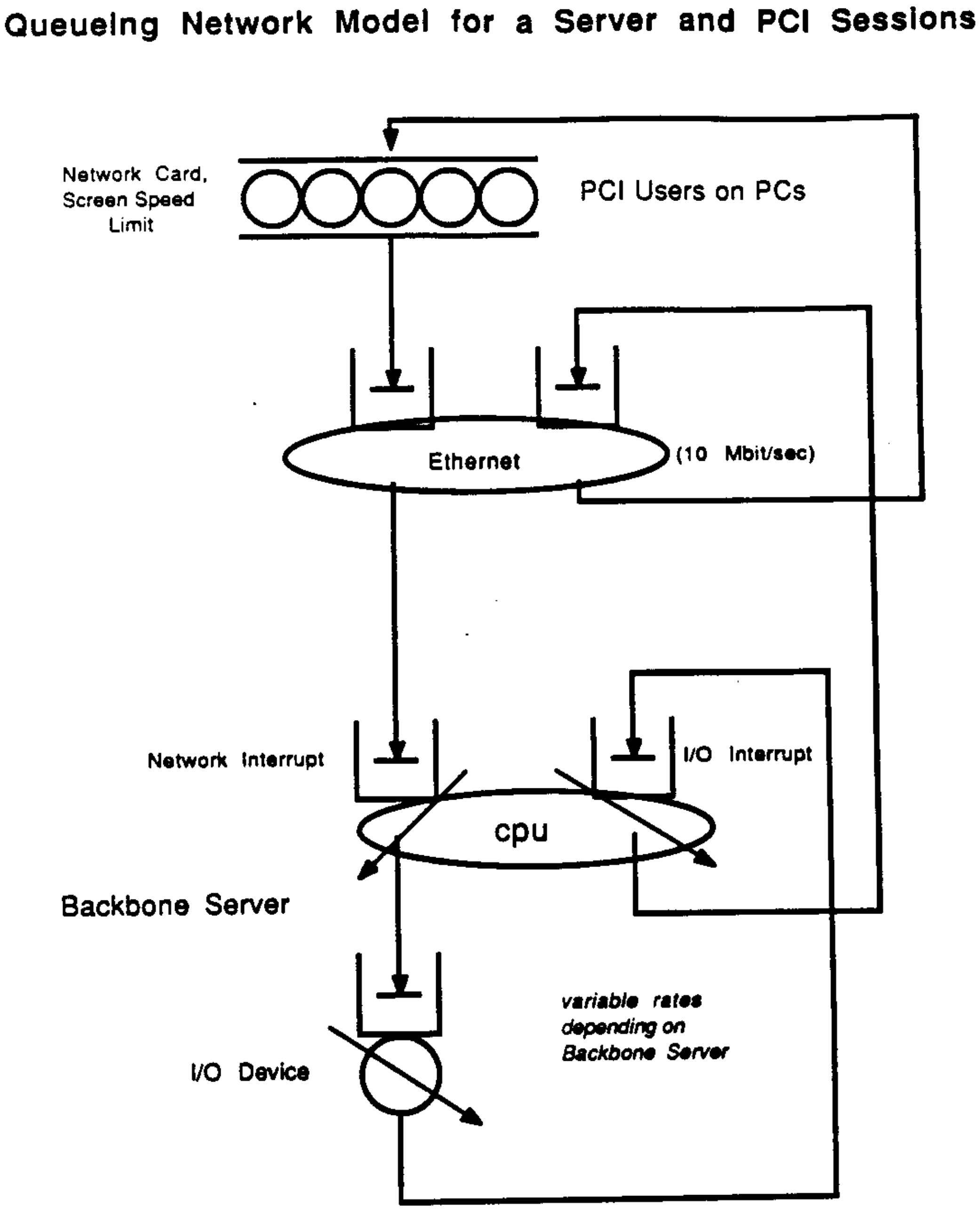


Figure 6.2: Queueing Network Model for a Backbone and PCI Sessions

The second sub-operation page faults once every four consecutive requests, as disk pages are 4K, and PCI packets contain 1K of data. We have also lumped together sub-operations 1 and 3 into an equivalent cpu service. As an example for the IBM machines we used 29200 cpu instructions. The VAX architecture requires fewer instructions, as Locus runs as a native operating system on the hardware.

Additional effects are added to accommodate HI and Int job classes. for these jobs we add user think time at the PCs. For HI we use 0.2 sec think time, and for Int we use 60 seconds.

Jobs of class Int also incur job swapping during their long think time periods. This overhead is considered as additional I/O overhead.

We also note that variable service rates are incorporated for the cpu and I/O service centers. This feature of the model supports heterogeneous sets of hardware such as the ones serving the SEASnet environment.

The service rate at the PC end is limited by the Ethernet Interface card to 25 KByte/sec. Additionally, if intensive screen output is required, 9600 baud (corresponding to about 1KByte/sec) becomes the effective bottleneck.

We proceed with a description of the model simulation and refinement processes.

6.6 Model Simulation and Refinement

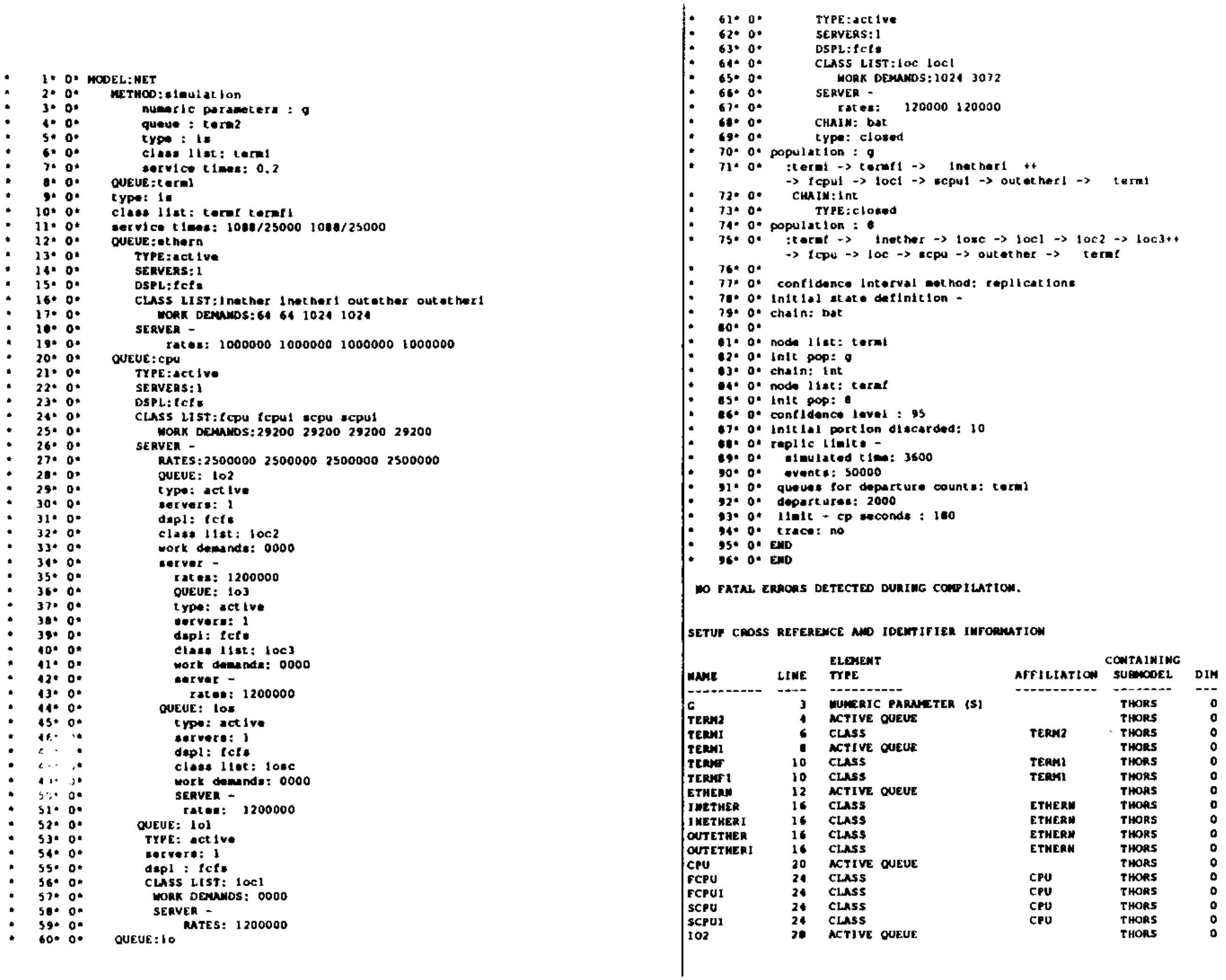
This queueing network model was simulated using RESQ2[2]. Confidence intervals of 95% were sought throughout the simulations. These evaluations were carried throughout the process of model refinement and validation. In Figure 6.3 we present a RESQ2 model description. In this example we show the model for a 4361 running jobs of classes CI and HI. It should be noted that the model uses two chains for the two user classes. Additional RESQ2 examples and evaluations are included later in the chapter and in Appendix C

The process of model refinement was lengthy and iterative. Assumptions were made, tested, and revisited as the model was being constructed. Given our limited detail as to the system internal components, we applied different load patterns to the system under study, and compared observed results to predicted figures.

As an example, the AIX/370 architecture required additional operating system layers to run on IBM 4300 machines. Initially we modeled this effect by tandem servers at the backbone machines, as well as attempting to balance the network resources. However, the best fit to measured results was obtained by increasing service requirements for the layered architecture.

Additional information which would be most desirable for further refinement would pertain to more precise cpu service requirements, process scheduling disci-

[2] RESQ2 (RESearch Queueing 2) is an enhanced performance modeling package developed by the IBM Research division



```
*    1* 0* MODEL:NET
*    2* 0*    METHOD:simulation
*    3* 0*        numeric parameters : g
*    4* 0*        queue : term2
*    5* 0*        type : is
*    6* 0*        class list: termi
*    7* 0*        service times: 0.2
*    8* 0*    QUEUE:terml
*    9* 0*    type: is
*   10* 0*    class list: termf termfi
*   11* 0*    service times: 1088/25000 1088/25000
*   12* 0*    QUEUE:ethern
*   13* 0*       TYPE:active
*   14* 0*       SERVERS:1
*   15* 0*       DSPL:fcfs
*   16* 0*       CLASS LIST:inether inetheri outether outetheri
*   17* 0*          WORK DEMANDS:64 64 1024 1024
*   18* 0*       SERVER -
*   19* 0*           rates: 1000000 1000000 1000000 1000000
*   20* 0*    QUEUE:cpu
*   21* 0*       TYPE:active
*   22* 0*       SERVERS:1
*   23* 0*       DSPL:fcfs
*   24* 0*       CLASS LIST:fcpu fcpui scpu scpui
*   25* 0*          WORK DEMANDS:29200 29200 29200 29200
*   26* 0*       SERVER -
*   27* 0*          RATES:2500000 2500000 2500000 2500000
*   28* 0*          QUEUE: io2
*   29* 0*          type: active
*   30* 0*          servers: 1
*   31* 0*          dspl: fcfs
*   32* 0*          class list: ioc2
*   33* 0*          work demands: 0000
*   34* 0*          server -
*   35* 0*            rates: 1200000
*   36* 0*            QUEUE: io3
*   37* 0*            type: active
*   38* 0*            servers: 1
*   39* 0*            dspl: fcfs
*   40* 0*            class list: ioc3
*   41* 0*            work demands: 0000
*   42* 0*            server -
*   43* 0*              rates: 1200000
*   44* 0*          QUEUE: ios
*   45* 0*            type: active
*   46* 0*            servers: 1
*   47* 0*            dspl: fcfs
*   48* 0*            class list: iosc
*   49* 0*            work demands: 0000
*   50* 0*            SERVER -
*   51* 0*             rates:  1200000
*   52* 0*       QUEUE: io1
*   53* 0*        TYPE: active
*   54* 0*        servers: 1
*   55* 0*        dspl : fcfs
*   56* 0*        CLASS LIST: ioc1
*   57* 0*         WORK DEMANDS: 0000
*   58* 0*         SERVER -
*   59* 0*             RATES: 1200000
*   60* 0*    QUEUE:io
*   61* 0*       TYPE:active
*   62* 0*       SERVERS:1
*   63* 0*       DSPL:fcfs
*   64* 0*       CLASS LIST:ioc iocl
*   65* 0*          WORK DEMANDS:1024 3072
*   66* 0*       SERVER -
*   67* 0*          rates:   120000 120000
*   68* 0*       CHAIN: bat
*   69* 0*       type: closed
*   70* 0* population : g
*   71* 0*   :termi -> termfi ->   inetheri  ++
           -> fcpui -> ioci -> scpui -> outetheri ->   termi
*   72* 0*    CHAIN:int
*   73* 0*       TYPE:closed
*   74* 0* population : 8
*   75* 0*   :termf ->   inether -> iosc -> iocl -> ioc2 -> ioc3++
           -> fcpu -> ioc -> scpu -> outether ->   termf
*   76* 0*
*   77* 0*  confidence interval method: replications
*   78* 0* initial state definition -
*   79* 0* chain: bat
*   80* 0*
*   81* 0* node list: termi
*   82* 0* init pop: g
*   83* 0* chain: int
*   84* 0* node list: termf
*   85* 0* init pop: 8
*   86* 0* confidence level : 95
*   87* 0* initial portion discarded: 10
*   88* 0* replic limits -
*   89* 0*   simulated time: 3600
*   90* 0*   events: 50000
*   91* 0*  queues for departure counts: terml
*   92* 0*  departures: 2000
*   93* 0*  limit - cp seconds : 180
*   94* 0*  trace: no
*   95* 0* END
*   96* 0* END

 NO FATAL ERRORS DETECTED DURING COMPILATION.


SETUP CROSS REFERENCE AND IDENTIFIER INFORMATION

                 ELEMENT                                CONTAINING
NAME       LINE  TYPE                       AFFILIATION  SUBMODEL   DIM
---------- ----  ----------                 -----------  --------   ---
G             3  NUMERIC PARAMETER (S)                    THORS      0
TERM2         4  ACTIVE QUEUE                             THORS      0
TERMI         6  CLASS                        TERM2       THORS      0
TERM1         8  ACTIVE QUEUE                             THORS      0
TERMF        10  CLASS                        TERM1       THORS      0
TERMF1       10  CLASS                        TERM1       THORS      0
ETHERN       12  ACTIVE QUEUE                             THORS      0
INETHER      16  CLASS                        ETHERN      THORS      0
INETHERI     16  CLASS                        ETHERN      THORS      0
OUTETHER     16  CLASS                        ETHERN      THORS      0
OUTETHERI    16  CLASS                        ETHERN      THORS      0
CPU          20  ACTIVE QUEUE                             THORS      0
FCPU         24  CLASS                        CPU         THORS      0
FCPUI        24  CLASS                        CPU         THORS      0
SCPU         24  CLASS                        CPU         THORS      0
SCPU1        24  CLASS                        CPU         THORS      0
IO2          28  ACTIVE QUEUE                             THORS      0
```

Figure 6.3: A RESQ2 Example for the Performance Model - 2 User Classes

plines and memory allocation strategies that are used by the Control Program[3] and AIX/370. Additional information regarding interrupt handling by the layered operating system would be useful as well.

We proceed with presentation of results obtained using our model.

6.7 Validation of Model Predictions with System Measurements

Model predictions for all classes of customers were carried out. We obtained sets of predicted plots for several measures of interest:

1. Delay to transfer 1MB of data
2. Single session throughput
3. Global system throughput

it is important to use measures that express the system's performance in its file service capacity, which is the central functionality of interest for our study. We present in Figures 6.4 - 6.9 some of the predictions of our model, along with measured results for several user classes under consideration.

We note that model predictions and results measured directly on the system are in good agreement. Demonstrated deviation is generally smaller than 25%. In particular, in Figure 6.4 we note that for Interactive (Int) sessions we obtained graceful system degradation for the 1MB benchmark. Results for the 4381 are

[3] The Control Program (CP) is the lower level operating system that controls the hardware. All communications from higher level operating systems go through CP.

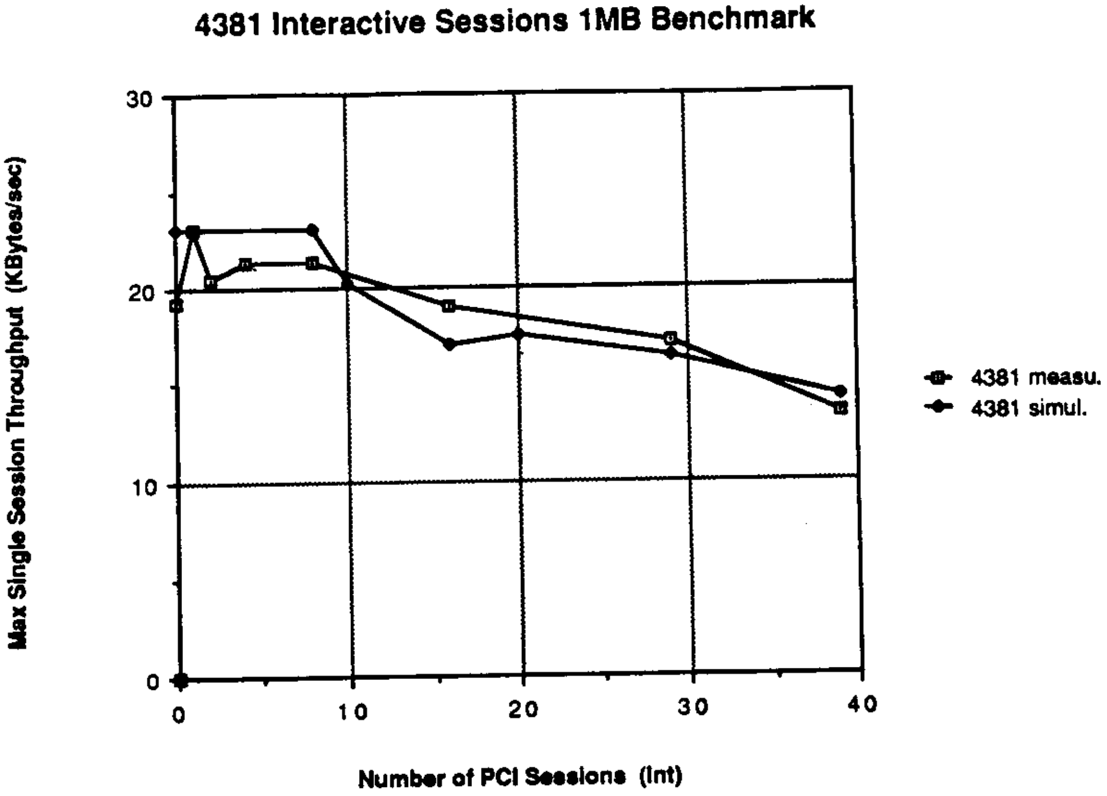


Figure 6.4: Model Predictions and System Measurements for User Load of Interactive (Int) Class (4381)

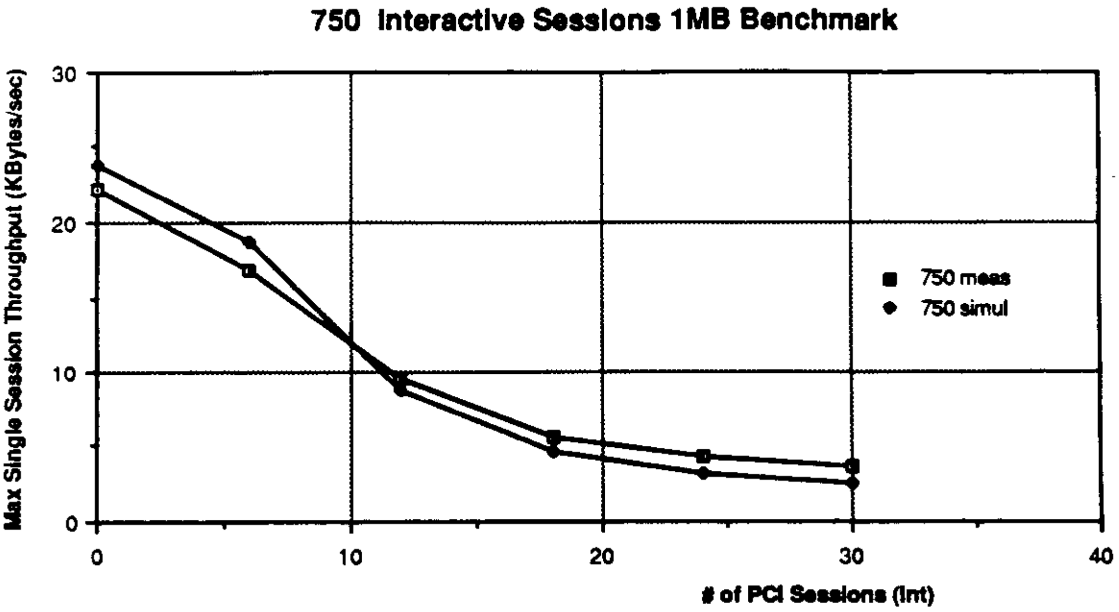


Figure 6.5: Model Predictions and System Measurements for User Load of Interactive (Int) Class (750)

shown, and similar results were reported for the 4361. We show in Figure 6.5 comparison results for the VAX 750 as well.

In Figure 6.6 we present results for High Interactive (HI) sessions on a 4381. Here we show very good agreement for the range of 4-20 sessions per backbone server. Degradation here is expectedly much quicker than for the Int sessions.

Figure 6.7 shows the global throughput of the system for a Communication Intensive (CI) setup. Under these trying conditions, we see impressive agreement in the work range of up to 20 sessions on the 4361 and over 30 sessions on the

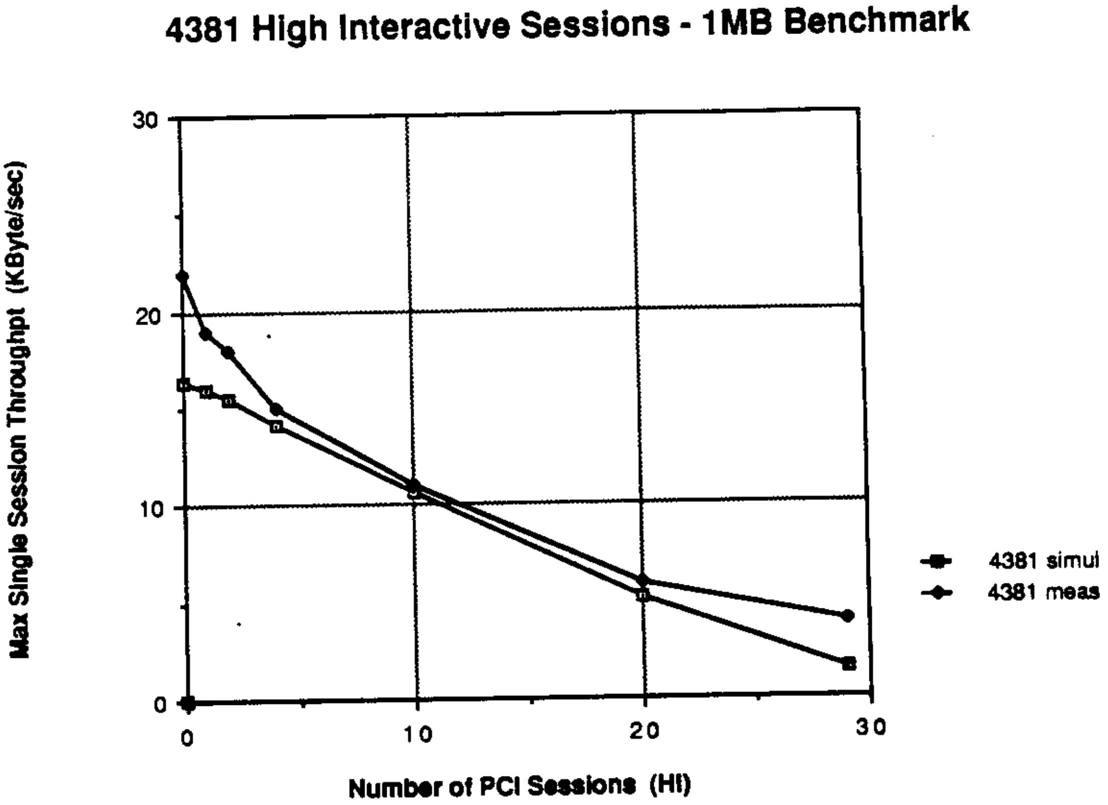
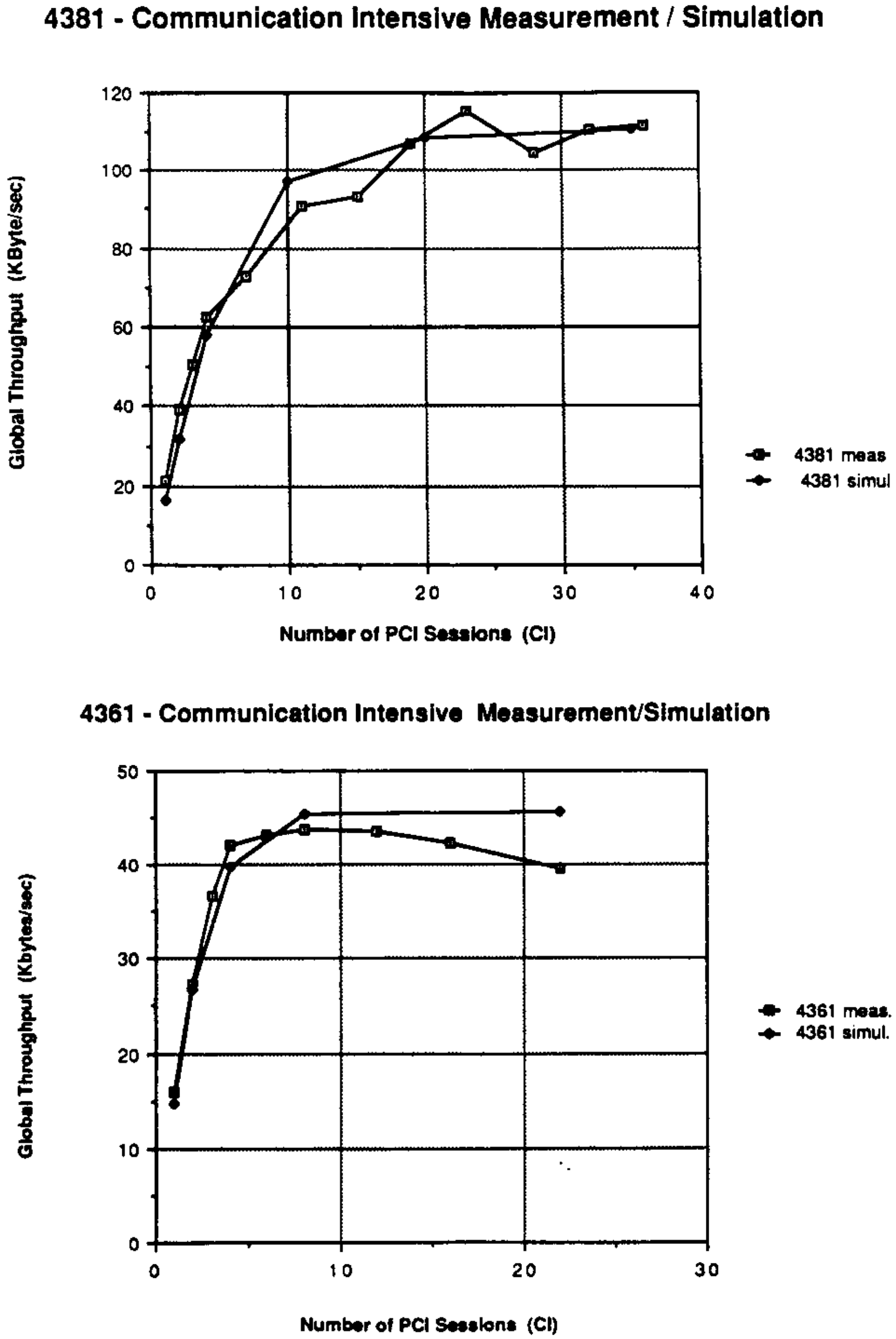


Figure 6.6: Model Predictions and System Measurements for User Load of High Interactive (HI) Class


4381 - Communication Intensive Measurement / Simulation

4361 - Communication Intensive Measurement/Simulation

Figure 6.7: Model Predictions and System Measurements for User Load of Communication Intensive (CI) Class

4381. Let us note that this traffic is far more intense than normal SEASnet file service requirements. It is highly unlikely that many users will require maximum throughput file transfer at the same time. However, this measure does give us indications about the limiting behavior of the system and the model, and thus its importance.

6.7.1 Multiple User Classes

In addition to the results reported above, we have been experimenting with combinations of user class load. This way we are putting under scrutiny all our assumptions and model components. In fact, we further show that simultaneous execution of jobs of different classes is correctly represented by the queueing network model.

In Figure 6.8 we show results for HI sessions with simultaneous CI sessions. All results correspond to the delay incurred for the 1MB transfer benchmark. We present results for both the IBM 4361 and the IBM 4381, for the range of 0-16 HI sessions and up to 4 CI sessions. Model predictions are within 25% of measured results throughout. It should be noted that for this complex workload combination we obtained good agreement between measurements and model evaluation.

In Figure 6.9 we extend our validation test space to combinations of all three classes, CI, HI, and Int. We present a comparison of predicted delays and measured delays for the 1MB transfer benchmark. This range of session combination

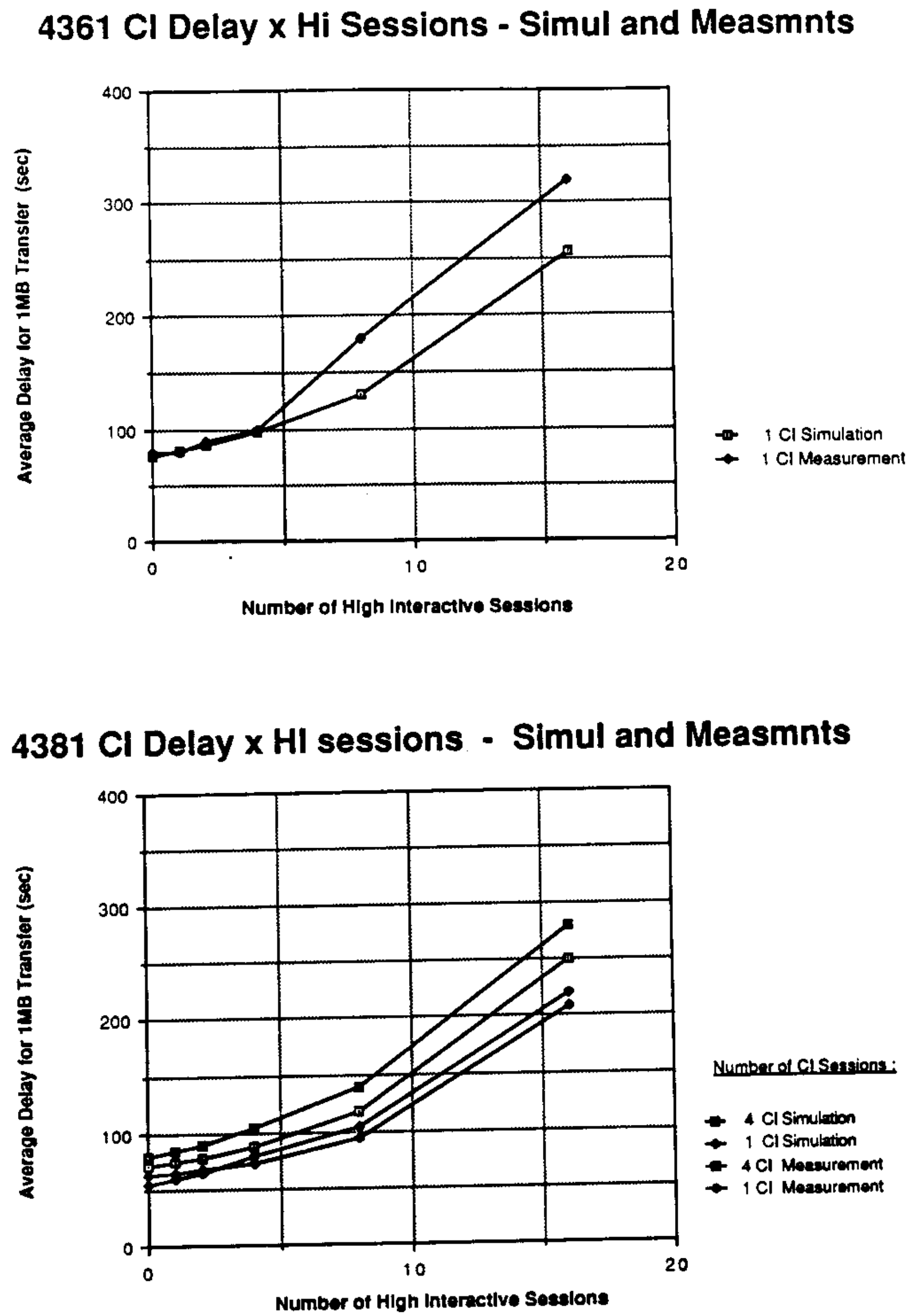


Figure 6.8: Model Predictions and System Measurements for User Load Combination of Classes CI and HI

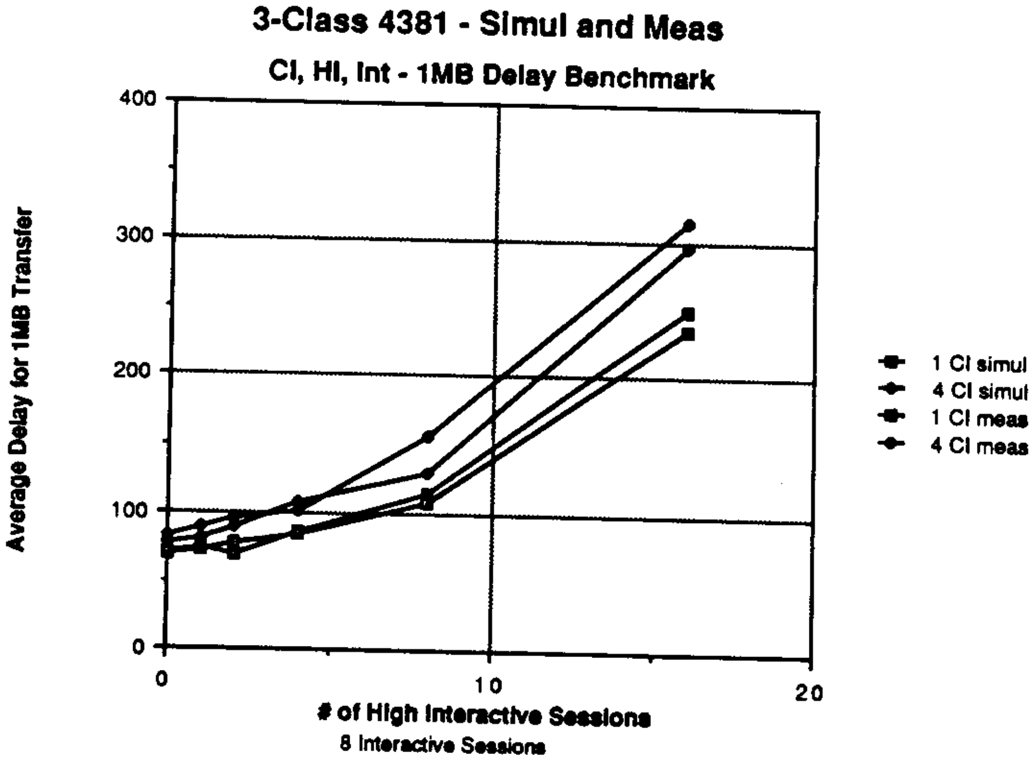


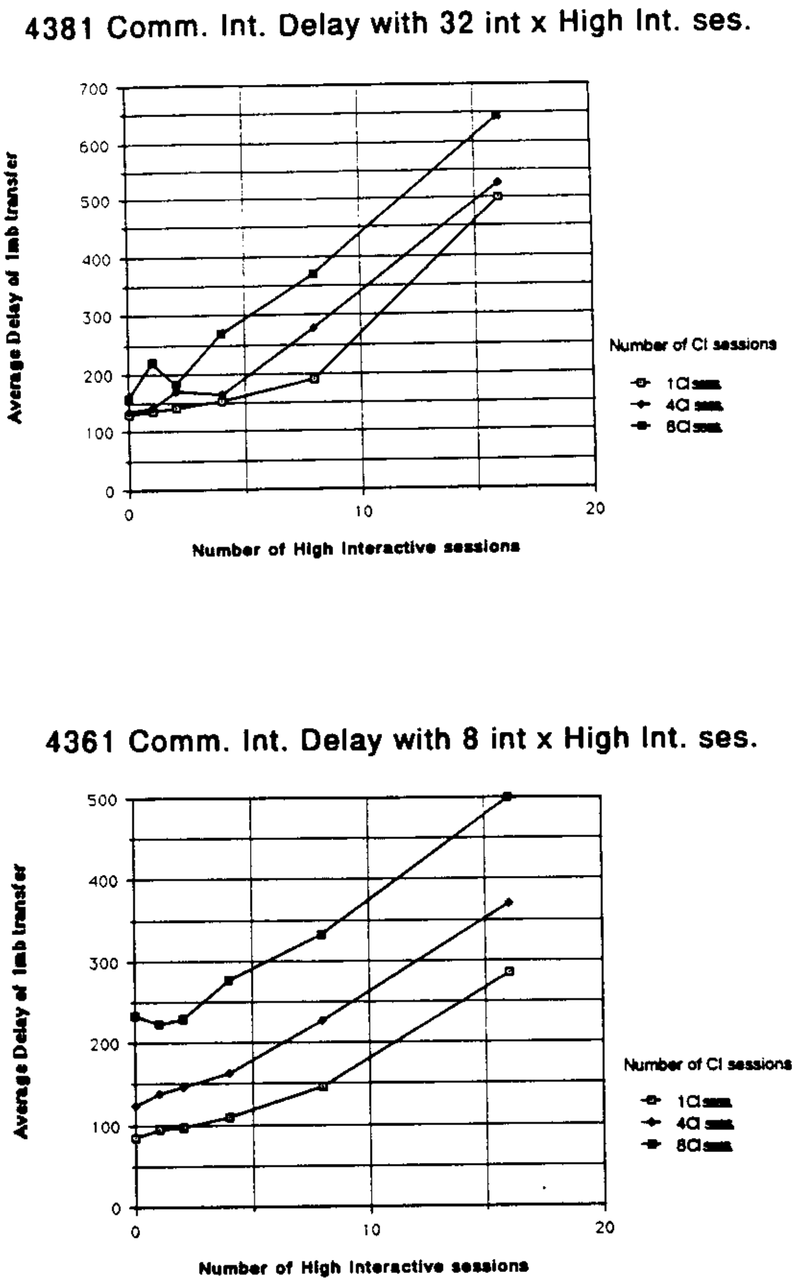
Figure 6.9: Model Predictions and System Measurements for User Load Combination of Classes CI, HI, Int

from one through twenty eight sessions covers a significant working range for the backbone server. The accuracy for this case falls within 25% as well. It is important to note that we characterize our user model as a combination of these three classes (Chapter 4). Therefore, we consider our description of SEASnet acceptable for further performance predictions.

Combined with results presented in the rest of the chapter, an array of load conditions and system functionality is covered. The agreement we obtained across the board gives us increased confidence in our modeling effort.

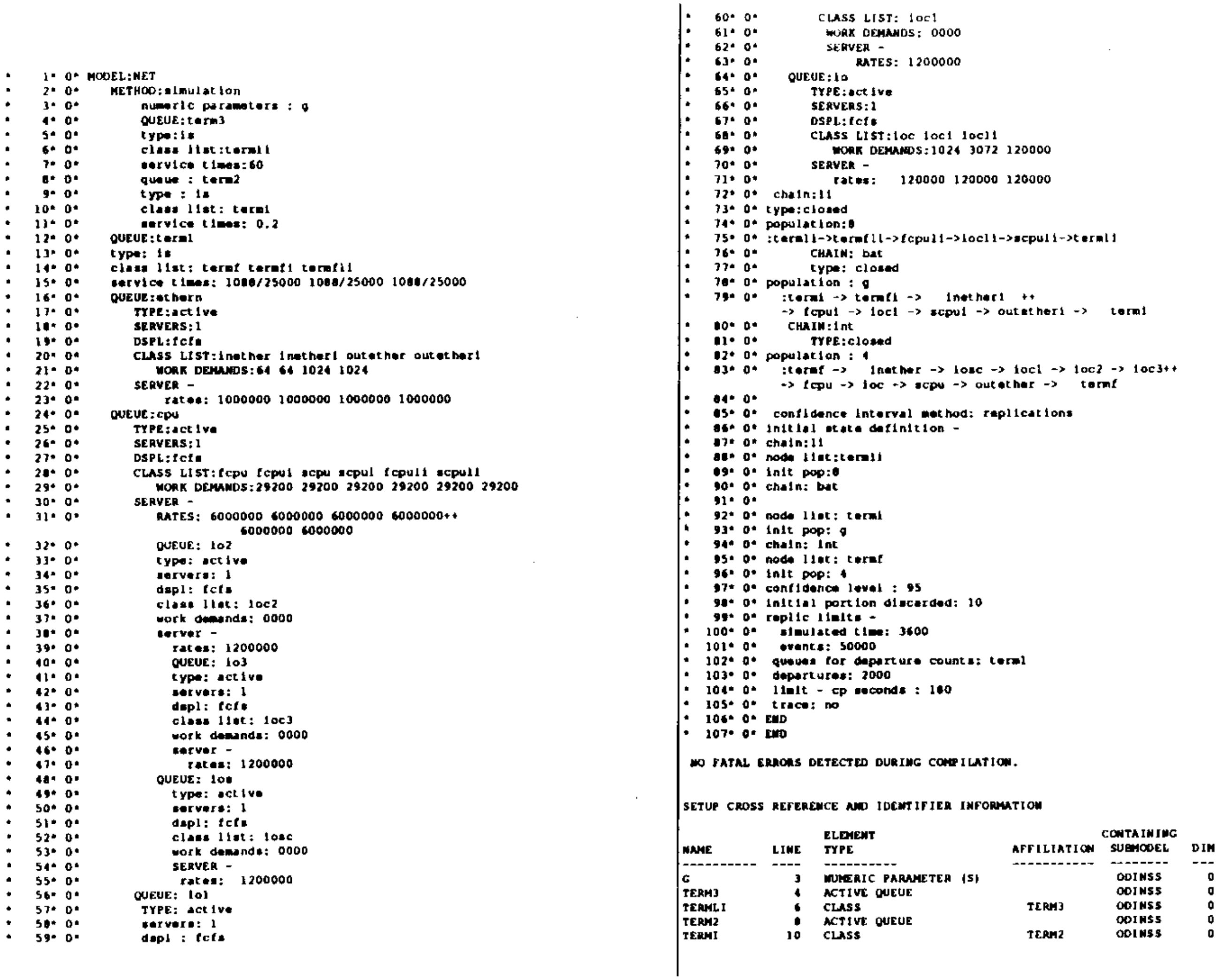
6.8 Further Results

Once a model was derived, additional future configurations could be evaluated with it. As an example to evaluations we present the three dimensional delay table generation that will enable us to use synthesis algorithms such as in chapter 7. We present in Figure 6.10 a sample of such three dimensional delay data. These results were obtained using a 3-class model such as in Figure 6.11. We note that this queueing network has 3 chains, corresponding User Classes CI, HI, and Int. We can see that these results, obtained by simulation of our model, are readily applicable for obtaining discrete delay information, thus relating expected performance to session assignment for each backbone server. We shall further elaborate on this idea in chapter 7.


4381 Comm. Int. Delay with 32 int x High Int. ses.

4361 Comm. Int. Delay with 8 int x High Int. ses.

Figure 6.10: Computed 3-Dimensional Delay Data Examples for Classes CI, HI, and Int



```
•    1* 0* MODEL:NET
•    2* 0*    METHOD:simulation
•    3* 0*       numeric parameters : g
•    4* 0*       QUEUE:term3
•    5* 0*       type:is
•    6* 0*       class list:termli
•    7* 0*       service times:60
•    8* 0*       queue : term2
•    9* 0*       type : is
•   10* 0*       class list: termi
•   11* 0*       service times: 0.2
•   12* 0*    QUEUE:terml
•   13* 0*    type: is
•   14* 0*    class list: termf termfi termfil
•   15* 0*    service times: 1088/25000 1088/25000 1088/25000
•   16* 0*    QUEUE:ethern
•   17* 0*      TYPE:active
•   18* 0*      SERVERS:1
•   19* 0*      DSPL:fcfs
•   20* 0*      CLASS LIST:inether inetheri outether outetheri
•   21* 0*         WORK DEMANDS:64 64 1024 1024
•   22* 0*      SERVER -
•   23* 0*          rates: 1000000 1000000 1000000 1000000
•   24* 0*    QUEUE:cpu
•   25* 0*      TYPE:active
•   26* 0*      SERVERS:1
•   27* 0*      DSPL:fcfs
•   28* 0*      CLASS LIST:fcpu fcpui scpu scpui fcpuli scpuli
•   29* 0*         WORK DEMANDS:29200 29200 29200 29200 29200 29200
•   30* 0*      SERVER -
•   31* 0*         RATES: 6000000 6000000 6000000 6000000++
                                  6000000 6000000
•   32* 0*         QUEUE: io2
•   33* 0*         type: active
•   34* 0*         servers: 1
•   35* 0*         dspl: fcfs
•   36* 0*         class list: ioc2
•   37* 0*         work demands: 0000
•   38* 0*         server -
•   39* 0*           rates: 1200000
•   40* 0*           QUEUE: io3
•   41* 0*           type: active
•   42* 0*           servers: 1
•   43* 0*           dspl: fcfs
•   44* 0*           class list: ioc3
•   45* 0*           work demands: 0000
•   46* 0*           server -
•   47* 0*             rates: 1200000
•   48* 0*         QUEUE: ios
•   49* 0*           type: active
•   50* 0*           servers: 1
•   51* 0*           dspl: fcfs
•   52* 0*           class list: iosc
•   53* 0*           work demands: 0000
•   54* 0*           SERVER -
•   55* 0*            rates:  1200000
•   56* 0*      QUEUE: io1
•   57* 0*       TYPE: active
•   58* 0*       servers: 1
•   59* 0*       dspl : fcfs
```

```
•   60* 0*         CLASS LIST: ioc1
•   61* 0*          WORK DEMANDS: 0000
•   62* 0*          SERVER -
•   63* 0*              RATES: 1200000
•   64* 0*    QUEUE:io
•   65* 0*       TYPE:active
•   66* 0*       SERVERS:1
•   67* 0*       DSPL:fcfs
•   68* 0*       CLASS LIST:ioc ioci iocli
•   69* 0*          WORK DEMANDS:1024 3072 120000
•   70* 0*       SERVER -
•   71* 0*          rates:   120000 120000 120000
•   72* 0*  chain:li
•   73* 0* type:closed
•   74* 0* population:8
•   75* 0* :termli->termfll->fcpuli->iocli->scpuli->termli
•   76* 0*       CHAIN: bat
•   77* 0*       type: closed
•   78* 0* population : g
•   79* 0*   :termi -> termfi ->   inetheri  ++
           -> fcpui -> ioci -> scpui -> outetheri ->   termi
•   80* 0*    CHAIN:int
•   81* 0*       TYPE:closed
•   82* 0* population : 4
•   83* 0*   :termf ->   inether -> iosc -> iocl -> ioc2 -> ioc3++
           -> fcpu -> ioc -> scpu -> outether ->   termf
•   84* 0*
•   85* 0*  confidence interval method: replications
•   86* 0* initial state definition -
•   87* 0* chain:li
•   88* 0* node list:termli
•   89* 0* init pop:8
•   90* 0* chain: bat
•   91* 0*
•   92* 0* node list: termi
•   93* 0* init pop: g
•   94* 0* chain: int
•   95* 0* node list: termf
•   96* 0* init pop: 4
•   97* 0* confidence level : 95
•   98* 0* initial portion discarded: 10
•   99* 0* replic limits -
•  100* 0*   simulated time: 3600
•  101* 0*   events: 50000
•  102* 0*  queues for departure counts: terml
•  103* 0*  departures: 2000
•  104* 0*  limit - cp seconds : 180
•  105* 0*  trace: no
•  106* 0* END
•  107* 0* END

 NO FATAL ERRORS DETECTED DURING COMPILATION.


SETUP CROSS REFERENCE AND IDENTIFIER INFORMATION

                      ELEMENT                                   CONTAINING
NAME          LINE    TYPE                         AFFILIATION  SUBMODEL    DIM
----------    ----    ----------                   -----------  --------    ---
G                3    NUMERIC PARAMETER (S)                      ODINSS       0
TERM3            4    ACTIVE QUEUE                               ODINSS       0
TERMLI           6    CLASS                          TERM3       ODINSS       0
TERM2            8    ACTIVE QUEUE                               ODINSS       0
TERMI           10    CLASS                          TERM2       ODINSS       0
```

Figure 6.11: Additional RESQ2 Example for the Performance Model - 3 User Classes

6.9 Accuracy and Credibility of Results

The accuracy of the model has been shown to be within 25% of measured behavior for unsaturated situations. We are confident that for the three user classes studied in this work and various combinations theirof, the model would yield comparable results. The methodology we developed is general, and can be readily applied to different situations, or further enhanced to accommodate for changes in system parameters.

One limitation of our work was the unavailability of detailed internal information. In particular, additional data regarding the system architecture of AIX/370 would be helpful in generalizing and refining our model. For this reason, we argue that additional modeling refinements would be necessary, should user classes of drastically different nature be defined. Environments such as X-windows, Graphics environments, or CPU intensive applications served directly by the backbone are a few examples that come to mind.

A number of these issues are currently under investigation at UCLA.

6.10 Computational Considerations

The simulation model we developed was run on the RESQ2 modeling package. The model components and corresponding chains that define the queuing network model were simulated via RESQ2 to obtain the performance measures of interest.

We ran RESQ2 on a high-end IBM 4341[4]. Individual runs consume about 3-5 minutes of cpu time for confidence intervals of 95%. In Figure 6.12 we show some numerical results obtained through simulation. The performance metrics are available for various subparts of the network. Each curve obtained requires 6-12 simulation points to obtain meaningful results. Hence, each change in the model required substantial evaluation effort for each set of curves that was generated. In the case of generating a three dimensional delay tables, a major several day effort was required. This effort was necessary in order to characterize the the various components that were later used for the synthesis effort.

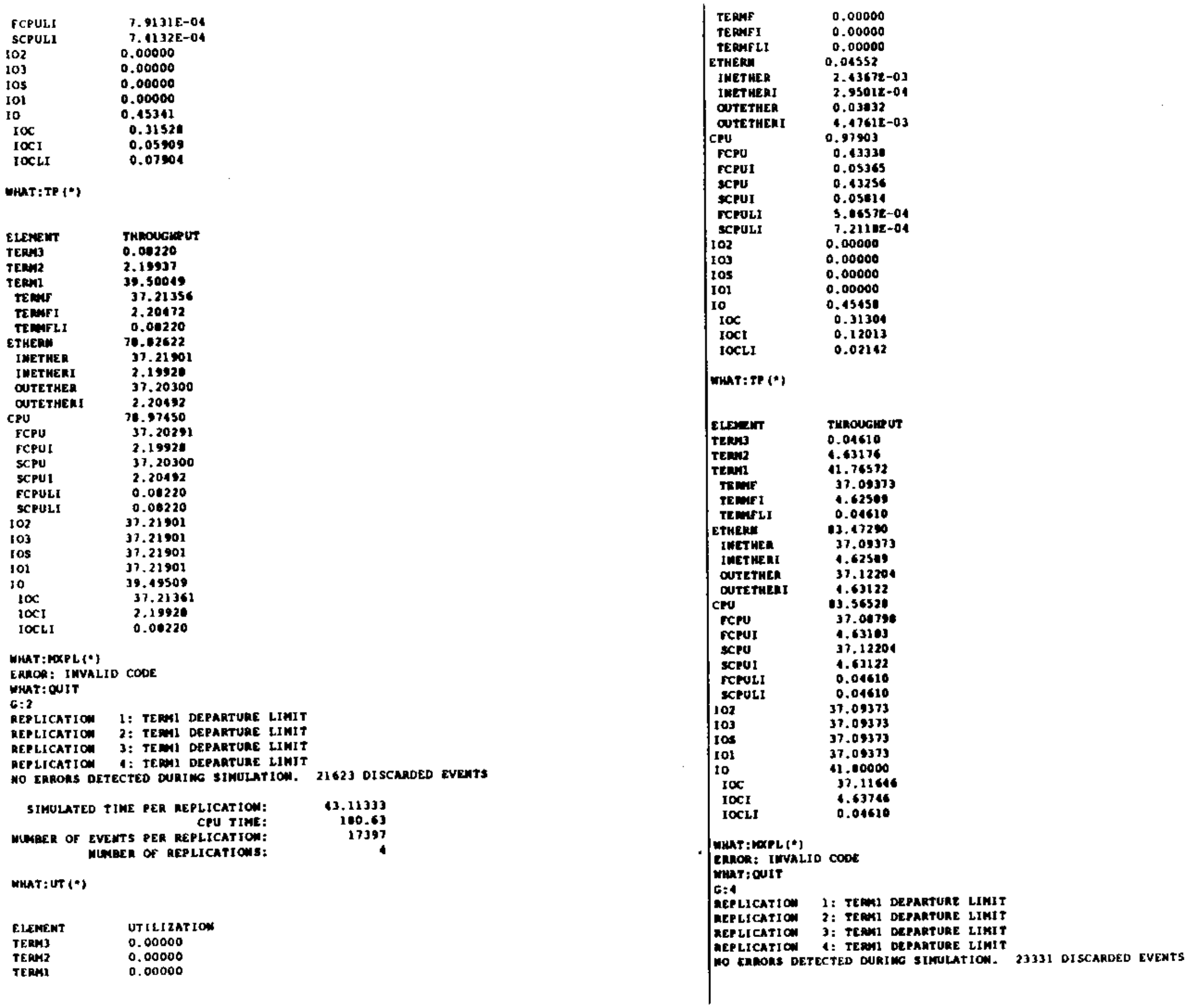
It should be noted that simulation costs are especially sensitive to the complexity of the problem, and increases in the detail of the model is likely to substantially increase the numerical computation cost.

6.11 Summary

The main impetus of our modeling work is to create a model for a complex system, in the presence of partial information about the system. This is a complex task, as one needs to deduce and assume values for details and parameters which are not available. As an example, the exact inter-relationships among the operating system layers as arranged by the developers are inaccessible to us, and we therefore lumped these effects as *total cpu service.*

Nevertheless, we were able to overcome the challenge of partial-parameter

[4] The UCLA CAD/CAM mainframe was used to perform the RESQ2 simulations for this work



```
 FCPULI          7.9131E-04
 SCPULI          7.4132E-04
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.45341
 IOC             0.31528
 IOCI            0.05909
 IOCLI           0.07904

WHAT:TP(*)


ELEMENT         THROUGHPUT
TERM3           0.08220
TERM2           2.19937
TERM1           39.50049
 TERMF           37.21356
 TERMFI          2.20472
 TERMFLI         0.08220
ETHERN          78.82622
 INETHER         37.21901
 INETHERI        2.19928
 OUTETHER        37.20300
 OUTETHERI       2.20492
CPU             78.97450
 FCPU            37.20291
 FCPUI           2.19928
 SCPU            37.20300
 SCPUI           2.20492
 FCPULI          0.08220
 SCPULI          0.08220
IO2             37.21901
IO3             37.21901
IOS             37.21901
IO1             37.21901
IO              39.49509
 IOC             37.21361
 IOCI            2.19928
 IOCLI           0.08220

WHAT:MXPL(*)
ERROR: INVALID CODE
WHAT:QUIT
G:2
REPLICATION   1: TERM1 DEPARTURE LIMIT
REPLICATION   2: TERM1 DEPARTURE LIMIT
REPLICATION   3: TERM1 DEPARTURE LIMIT
REPLICATION   4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  21623 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:         43.11333
                        CPU TIME:           180.63
NUMBER OF EVENTS PER REPLICATION:            17397
          NUMBER OF REPLICATIONS:                4

WHAT:UT(*)


ELEMENT         UTILIZATION
TERM3           0.00000
TERM2           0.00000
TERM1           0.00000
```

```
 TERMF           0.00000
 TERMFI          0.00000
 TERMFLI         0.00000
ETHERN          0.04552
 INETHER         2.4367E-03
 INETHERI        2.9501E-04
 OUTETHER        0.03832
 OUTETHERI       4.4761E-03
CPU             0.97903
 FCPU            0.43338
 FCPUI           0.05365
 SCPU            0.43256
 SCPUI           0.05814
 FCPULI          5.8657E-04
 SCPULI          7.2118E-04
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.45458
 IOC             0.31304
 IOCI            0.12013
 IOCLI           0.02142

WHAT:TP(*)


ELEMENT         THROUGHPUT
TERM3           0.04610
TERM2           4.63176
TERM1           41.76572
 TERMF           37.09373
 TERMFI          4.62509
 TERMFLI         0.04610
ETHERN          83.47290
 INETHER         37.09373
 INETHERI        4.62589
 OUTETHER        37.12204
 OUTETHERI       4.63122
CPU             83.56528
 FCPU            37.08798
 FCPUI           4.63103
 SCPU            37.12204
 SCPUI           4.63122
 FCPULI          0.04610
 SCPULI          0.04610
IO2             37.09373
IO3             37.09373
IOS             37.09373
IO1             37.09373
IO              41.80000
 IOC             37.11646
 IOCI            4.63746
 IOCLI           0.04610

WHAT:MXPL(*)
ERROR: INVALID CODE
WHAT:QUIT
G:4
REPLICATION   1: TERM1 DEPARTURE LIMIT
REPLICATION   2: TERM1 DEPARTURE LIMIT
REPLICATION   3: TERM1 DEPARTURE LIMIT
REPLICATION   4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  23331 DISCARDED EVENTS
```

Figure 6.12: RESQ2 Evaluation Example for the Performance Model - 3 User Classes

modeling, and obtain a representative description of our system, that demonstrated good predictive capacity. Qualities, such as delay and throughput, which were important to us were well modeled, and performance parameters were directly obtained from the model. This was demonstrated for a variety of system configurations and load patterns to which the system was subjected.

We shall show in the next chapter an important and interesting application for our model.

CHAPTER 7

Configuration Synthesis for a Heterogeneous Backbone Cluster and a PC-Interface Network

This chapter combines the results of the user model and the system model contributions into an important and interesting system synthesis contribution. We make note that the methodology generally applies to systems other than Locus, AIX/370 or PCI. We used our system as a typical example of a system whose model and parameterization were known.

We address in this chapter the design of Locus[1] [PopeWalk85] family networks rendering PCI service. Given are the expected user workload, the hardware costs and the performance constraints. The workload consists of three classes of users: *Interactive, High-Interactive, and Communication-Intensive.* The number of PC's is given and is equal to the number of users. We show that the problem can be reduced to a discrete Capacity and Flow Assignment (CFA) problem. The backbone capacity assignment is inspired by the Lagrangian Decomposition Approach [Fox66]. It starts from a backbone capacity assignment which matches the initial flow and chooses a backbone server upgrade that gives the greatest response time reduction per Dollar. At this point we apply the flow

[1] Locus is the predecessor to the forthcoming IBM AIX/370

deviation method and iterate until the constraints are met. We present several computational examples of network configuration synthesis, that emphasize the significance and generality of the results obtained.

7.1 Introduction

Locus is the predecessor of the newly announced IBM AIX/370. It is a Unix compatible distributed operating system that supports a large variety of architectures. Some of the mainframe architectures supported are 4300, 9370 and the 3090. A set of such backbone servers is called a "Transparent Computing Facility" cluster (TCF cluster). Personal workstations such as high-end PS/2 and PC/RT are supported as well, and they can be used to access a TCF backbone cluster.

The UCLA School of Engineering and Applied Sciences Network (SEASnet) is the primary computing facility of the school. SEASnet also serves as a pilot/demonstration center, as it is the most experienced production environment for pre-AIX/370. SEASnet demonstrates a high degree of heterogeneity, as it consists of two 4361s and a 4381 as backbone servers (as well as a number of DEC VAX/750s SUNs FPS and others as in Figure 7.1). On the client side, SEASnet users use primarily IBM PC/ATs running DOS to access the system (ascii terminals and PC/RTs are also available).

In this work we shall concentrate on the PC to backbone-cluster connection. This connection is accomplished by an operating system bridge called

PC-Interface (PCI). PCI allows a PC running DOS to transparently access an AIX or Unix file system through the so-called network drive (Drive E:).

The problem of optimal synthesis of such heterogeneous systems is both compound and difficult. There are several subparts to this general problem, namely - *component measurement and characterization, load and system model construction, and lastly, the sub-optimal configuration synthesis algorithms.* The major contribution of this Chapter is the last among the three - the configuration synthesis. This contribution is very general, as the presented algorithms can be applied to many resource systems which need to be optimized.

We begin this Chapter by providing the necessary context that inspired this work. We shall describe the system that was used to measure and characterize the components of the synthesized configurations in the computational example we present. Hence, further system description and workload characterization will be provided in section 7.2.

We then proceed with an exposition of our top-level approach in section 7.3. We describe how the backbone cluster synthesis problem can be mapped into an extension to the Capacity and Flow Assignment (CFA) problem.

In section 7.4 we present the extended sub-optimizing CFA algorithms we developed for the synthesis problem. Both Capacity Assignment and Sub-Optimal Session Assignment algorithms are outlined.

In section 7.5 we apply the algorithms to three different configuration sets. We derive sub-optimal session assignments, and ultimately present the cost /

performance curves that readily yield the configurations of choice.

In section 7.6 we outline analysis insights and provide general conclusions for this work, as well as suggest additional directions and further extensions to our work.

We proceed with the description of the system that inspired this work.

7.2 System Description and Workload Characterization

SEASnet is an Ethernet based network, serving the UCLA School of Engineering. Figure 7.1 illustrates the general layout of SEASnet through the school. Our focus for this work is in the PCI networking option. PC-Interface allows any of the 100+ PCs in the school to access any of the backbone machines. Figure 7.2 shows the part of SEASnet on which we focus, and from which we derive the computational example of section 7.5. The backbone servers provide extensive file service for class instruction, as well as research work. We present here algorithms that optimize resource utilization, as well as synthesize systems with more (or fewer) available resources.

Extensive measurements were carried out on SEASnet in order to parameterize its resources and construct a queueing network model. Most of these measurements are reported in Chapter 4 and in Appendix B [Betser87, BeAvCaKa88] An important benchmark was the delay incurred in a 1 Mega-Byte file transfer. This is a crucial yardstick for a file service oriented system.

Based on our study of the SEASnet user behavior, we introduced three classes

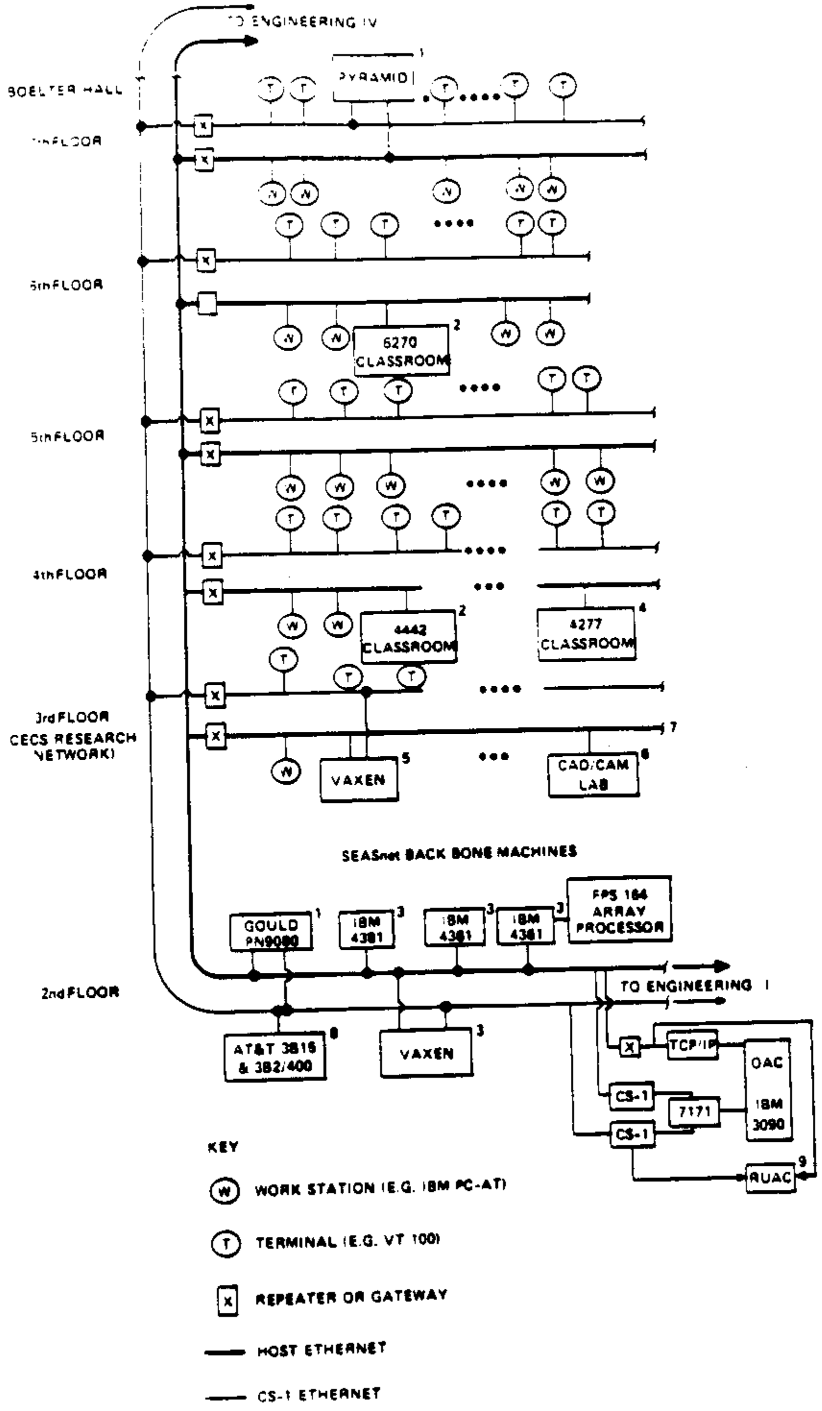


Figure 7.1: SEASnet - General Layout within the School of Engineering

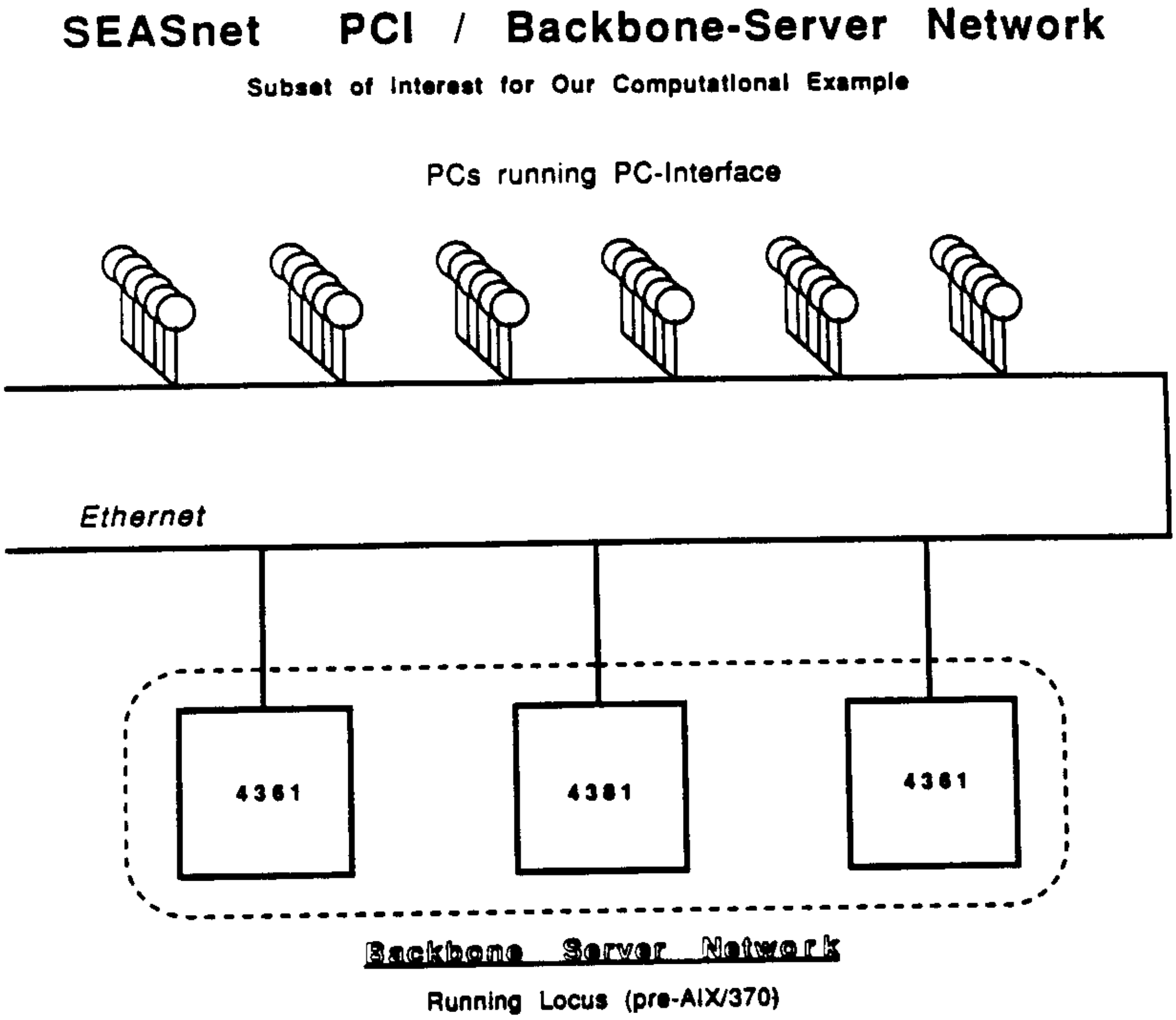

SEASnet PCI / Backbone-Server Network
Subset of Interest for Our Computational Example
PCs running PC-Interface
Ethernet
4361
4381
4361
Backbone Server Network
Running Locus (pre-AIX/370)

Figure 7.2: SEASnet - PCI / Backbone-Server Network

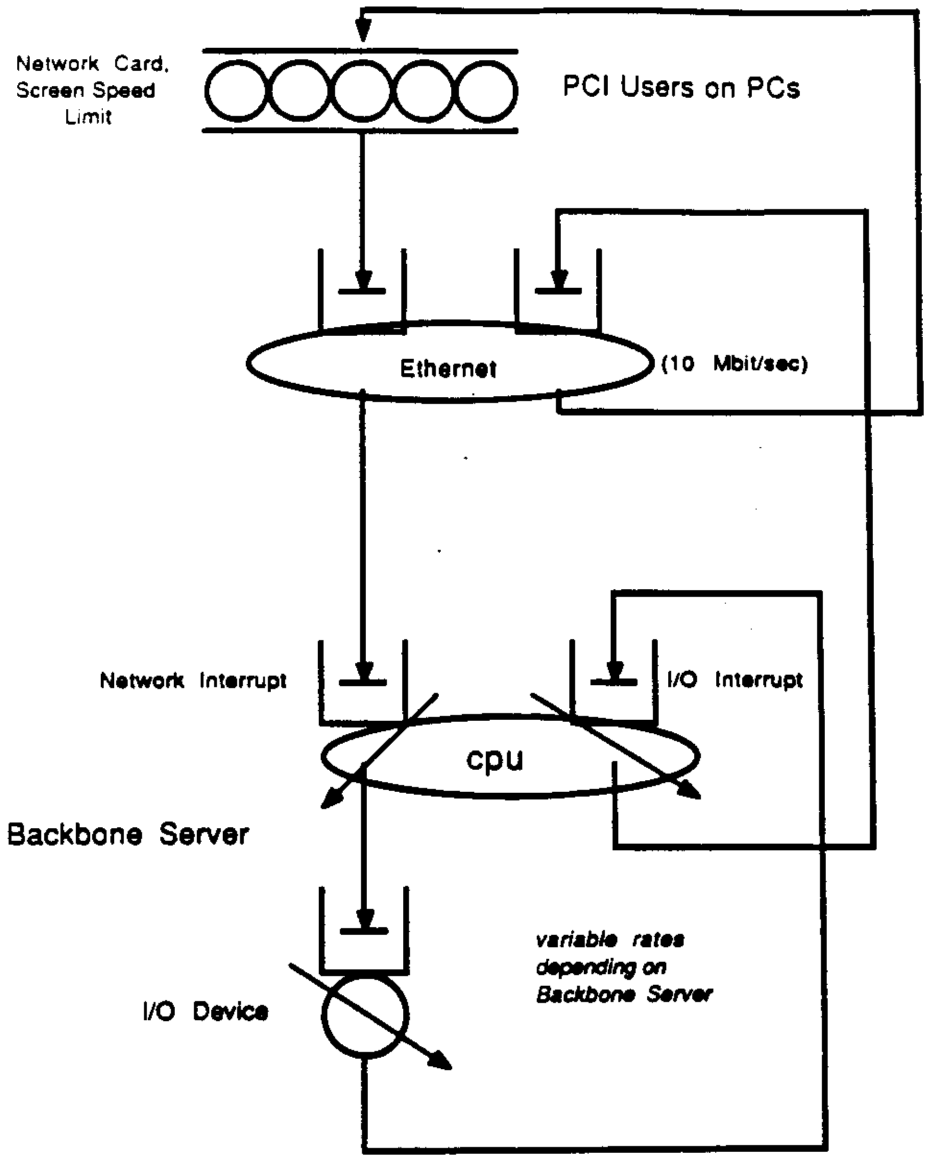
of users in our characterization:

1. Interactive (Int) : Short interactive commands interleaved with 1 min think time.
2. High Interactive (HI) : Short interactive commands in succession with 1-2 sec think time.
3. Communication Intensive (CI) : Massive continuous file transfer from backbone servers to PCs.

The performance figures for workloads with customers of different classes were derived through measurements and simulation. With the aid of many such experiments, under various configurations and loads combinations, we have identified the contention points for SEASnet. We then constructed the queueing network model shown in Figure 7.3. This model describes a Locus backbone server supporting a variety of PCI sessions.

We model the primary governing parameters of the Locus-PCI operation paradigm. It is important to note that we present the contention points as the CPU speed and I/O capacity of the backbone servers. We also present a bottleneck at the PC end, in the way of the Ethernet interface speed. We have observed that these are the primary resource contention modules within this architecture[2].

[2] The Ethernet is an example of a non-bottlenecking resource. We have never gotten even close to full Ethernet throughput during our extensive experimentation



Queueing Network Model for a Server and PCI Sessions

Figure 7.3: Queueing Network Model for a Backbone and PCI Sessions

The tuning and validation of of this model was a long iterative process, and is reported in detail in Chapter 6 and [BeAvCaKa88]. We present in Figure 7.4 some recent comparisons between measured results and simulated results generated by the queueing network model.

Our model was used to simulate many of the configurations that we reported in the computational example of section 5. These extensive simulations were used to construct the delay space tables. These tables were used as input to the optimization and synthesis algorithms we subsequently conducted. In Appendix E we present some of the 3-class simulation results that were used to construct the delay tables for the configuration synthesis algorithms.

Having provided this motivation, we proceed to the description of the optimizing algorithms.

7.3 Reduction to the Capacity and Flow Assignment Problem

For a given a user load, the *synthesis of heterogeneous backbone clusters for PCI networks* consists of two major decisions :

1. Number and type of backbone servers to allocate (cost constraint bounded)

2. Specific session allocation onto the backbone servers

The problem can be transformed into a discrete Capacity and Flow Assignment problem (CFA) by mapping each backbone server onto a link, and mapping

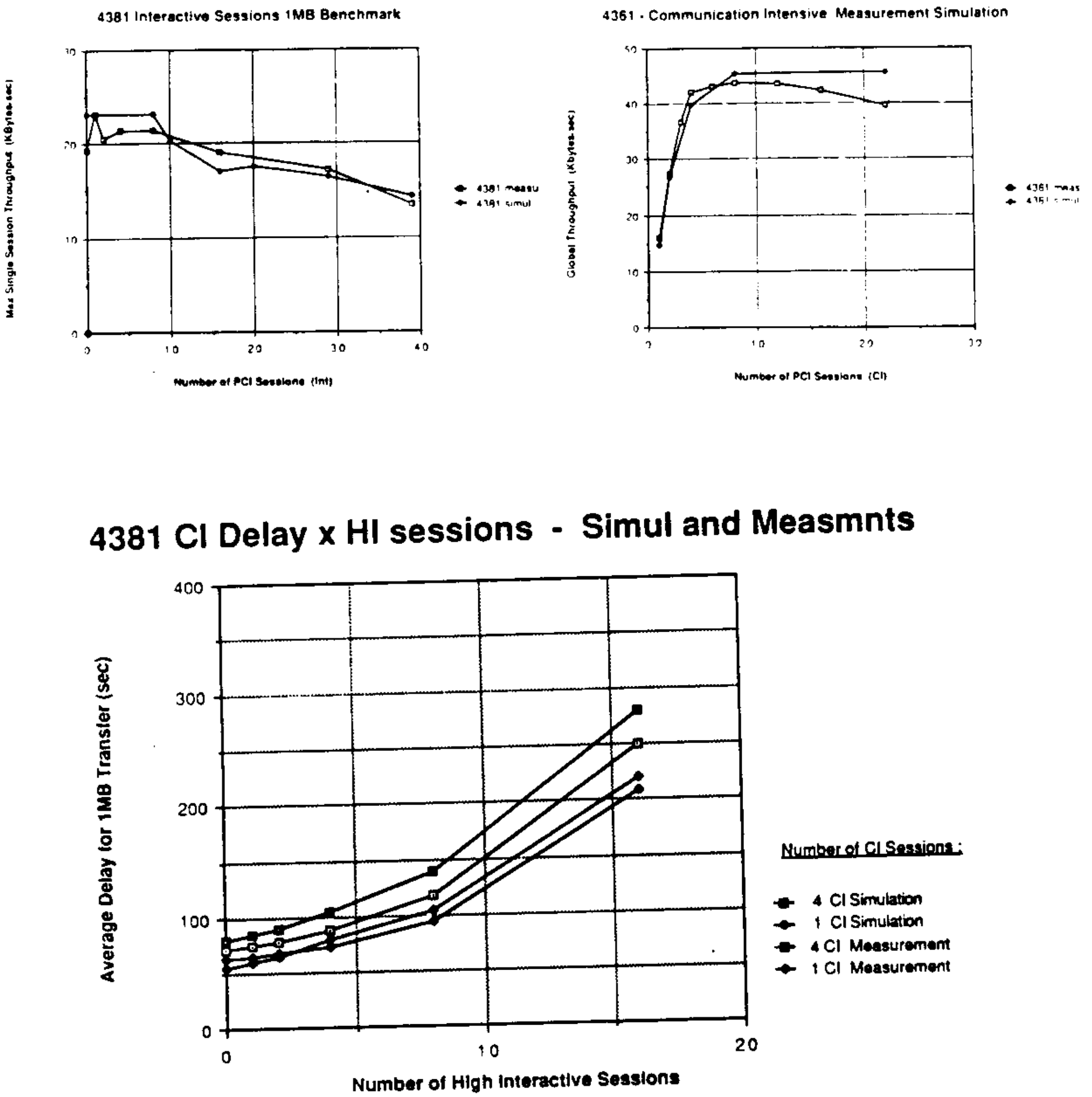


Figure 7.4: Comparison of Measurement and Simulation Results

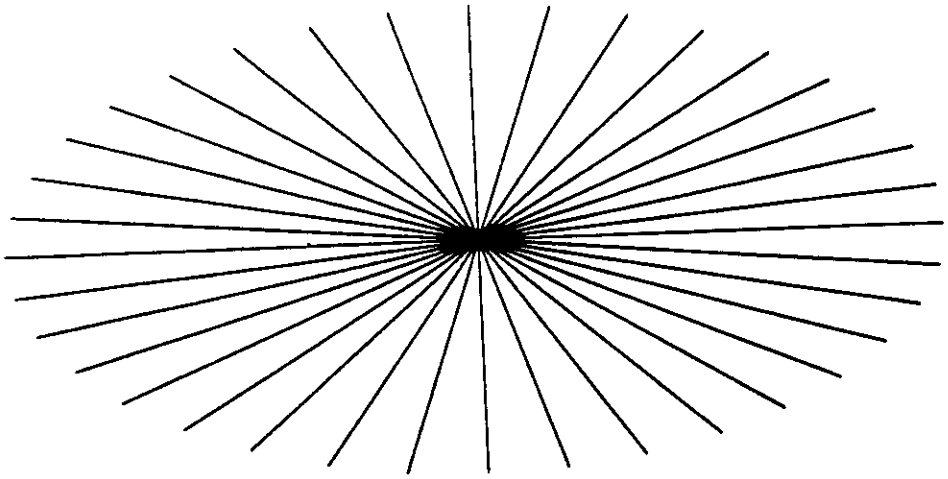


Figure 7.5: Corresponding Star Topology for Backbone-Server Cluster

the number of sessions allocated to a backbone onto the flow of the corresponding link. This entails a star topology, as illustrated in Figure 7.5.

We will solve the original problem by solving the CFA between a source and a destination connected by a number of links equal to the maximum number of sites that any feasible solution may include and having as the total flow the entire user workload. When the CFA algorithm converges, the number of backbone servers in the system will be equal to the number of links with nonzero capacity, and the flow on each link will represent the session set assigned to the backbone server.

We must search through many local minima until we find a cost effective solution that satisfies the problem constraints. That will be done by choosing different starting feasible flows for each topology considered.

We will execute the capacity assignment algorithm to find the initial session allocation to each backbone server that satisfies the cost constraints. We will then iterate between the Flow Deviation (FD) and the Capacity Assignment (CA) until we find a local minimum. The workload that will result from this process will be suboptimal due to the fact that the objective function is concave.

In the following sections we will consider the algorithms in detail, and apply them to concrete examples to evaluate their effectiveness.

7.4 The Capacity and Flow Assignment Algorithm

7.4.1 Capacity Assignment Algorithm

Let's first introduce some notation:

$backbone(i)$ - backbone server (site) allocated to link i.

$cost(i)$ - cost of backbone(i).

$upgrade(i)$ - backbone server allocated to link i after upgrade.

T - average system response time.

T_{max} - constraint on average system response time.

$\Delta T(i)$ - difference in the delay of link i upon upgrade of backbone server.

r - Incremental delay per upgrade cost.

The Capacity Assignment algorithm is as follows:

1. Choose the minimum fit capacity assignment that matches the flows in each link. For every backbone server that has at least one session assigned to it,

the minimum fit assignment will be the least expensive backbone server. If no session is assigned to a backbone server, that site will be deleted.

2. If the total cost assigned to the links is greater than the maximum cost, then STOP. The problem is not feasible.

3. Calculate the total average response time of the system by table look up.

4. If T $< T_{max}$ STOP. The problem is feasible, and the allocated capacity is suboptimal.

5. Compute the ratio $r = -\frac{\Delta T(i)}{cost(upgrade(i)) - cost(backbone(i))}$ for all links. r is the incremental delay per upgrade cost. If the backbone server assigned to a site cannot be improved (it is already the most expensive backbone server available), then set r=0.

6. Find the link that has the largest r (r is always positive). If $r = 0$, then STOP (all sites are already using the most expensive backbone server). Otherwise upgrade the backbone server in that link and GOTO step 2.

The algorithm will find a cost effective solution, since at every step it upgrades the capacity that gives the greatest response time reduction per Dollar. The method will typically generate suboptimal solutions. An optimal solution could be obtained (at higher cost) using a dynamic programming approach [Frank.etal69].

7.4.2 The Sub Optimal Session Assignment Algorithm

The sub-optimal session assignment algorithm is as follows:

1. Compute T_a the average response time at the initial flow assignment.

2. For each link compute the incremental delay as a function of a unit increment in the flow (transfer of δ_c sessions).

3. Find the link that has shortest incremental delay.

4. Find the link (having nonzero flow) that has the maximum incremental delay (the zero flow links are not taken into consideration).

5. Deviate a unit flow(δ_c) from the maximum incremental delay class of the maximum incremental delay link to the minimum incremental delay link.

6. Compute T_c the average response time at the current flow assignment.

7. If $T_a - T_c < \epsilon$ or $T_c > T_a$, then STOP. Otherwise do $T_a = T_c$, and GOTO step 2.

The algorithm computes the incremental delay for each link by computing the numerical partial derivative with respect to each class and deviates δ_c sessions of the maximum incremental delay class from the link with maximum incremental delay, to the one with minimum incremental delay. Each class has a constant δ_c that is calculated according to the load that a session from that class brings to the system. In our computational examples the classes High Interactive and Communication Intensive have $\delta_c = 1$ and the class Interactive has $\delta_c = 4$.

In the next section we apply the algorithms to three different topologies under various costs and compute the cost performance curves for uniform, skewed, and suboptimal workloads. The suboptimal workload is the workload that results upon convergence of the Capacity and Flow Assignment algorithm.

7.5 Configuration Synthesis Examples

In this section we present three configuration synthesis examples. The workload consists of three classes, namely Interactive (Int), High Interactive (HI), and Communication Intensive (CI). There is a total of 90 sessions divided among the classes (48Int, 24HI, 18CI). We consider costs in the range of 1.2 to 7.0 million Dollars and topologies with 4,6, and 10 backbone servers. The backbone servers can be a 4361 or a 4381 with typical costs of 300k and 700k Dollars[3]. The assignment of sessions to the backbone servers can be *skewed, uniform, or suboptimal.* A skewed assignment is one where some backbone servers are overloaded and others are underloaded. With this workload we represent the situation that occurs when users are permitted to connect to any backbone server and the system is not balancing the load. A uniform assignment results when each backbone server receives an equal number of sessions from each class. The suboptimal assignment

[3] These costs are illustrative. They can vary substantially depending on system peripherals, customer discount, etc. Appendix D contains some detailed examples of the cost determination procedure for typical systems.

	Uniform			*Skewed*		
Site	*Int*	*HI*	*CI*	*Int*	*HI*	*CI*
1	12	6	4	16	0	4
2	12	6	4	16	0	4
3	12	6	5	8	12	5
4	12	6	5	8	12	5

Table 7.1: Uniform and Skewed Workloads with 4 Backbone Servers

is the result of the optimization described in the previous sections.

7.5.1 Synthesis with 4 Backbone Servers

In this example we divide the 90 sessions among the classes as follows: 48 interactive, 24 high interactive and 18 communication intensive. The uniform and skewed workloads for 4 sites are shown in Table 7.1.

We ran the optimization for costs ranging from 1.2 million Dollars to 2.8 million Dollars. This represents the entire cost allocation space. In Tables 7.2 and 7.3 we show the backbone servers allocated to each site, the corresponding cost, and the suboptimal workload computed.

In Figure 7.6 we plot the corresponding average delay for each configuration. We can see that the upgrade of the backbone servers is producing significant improvement in the performance. This occurs because the load applied to system is overwhelming the 4361s.

		Sessions in each class		
Site	*Backbone*	*Int*	*HI*	*CI*
cost of 1.2 M Dollars				
1	4361	16	0	5
2	4361	16	0	4
3	4361	8	12	7
4	4361	8	12	8
cost of 1.6 M Dollars				
1	4361	12	6	4
2	4381	16	5	7
3	4361	12	7	3
4	4361	8	6	4
cost of 2.0 M Dollars				
1	4361	12	6	3
2	4381	12	5	7
3	4381	16	7	5
4	4361	8	6	3

Table 7.2: Suboptimal Workload with 4 Backbone Servers (part 1)

		Sessions in each class		
Site	*Backbone*	*Int*	*HI*	*CI*
cost of 2.4 M Dollars				
1	4381	12	6	6
2	4361	12	6	2
3	4381	12	5	6
4	4381	12	7	4
cost of 2.8 M Dollars				
1	4381	12	4	6
2	4381	12	5	6
3	4381	12	9	2
4	4381	12	6	4

Table 7.3: Suboptimal Workload with 4 Backbone Servers (part 2)

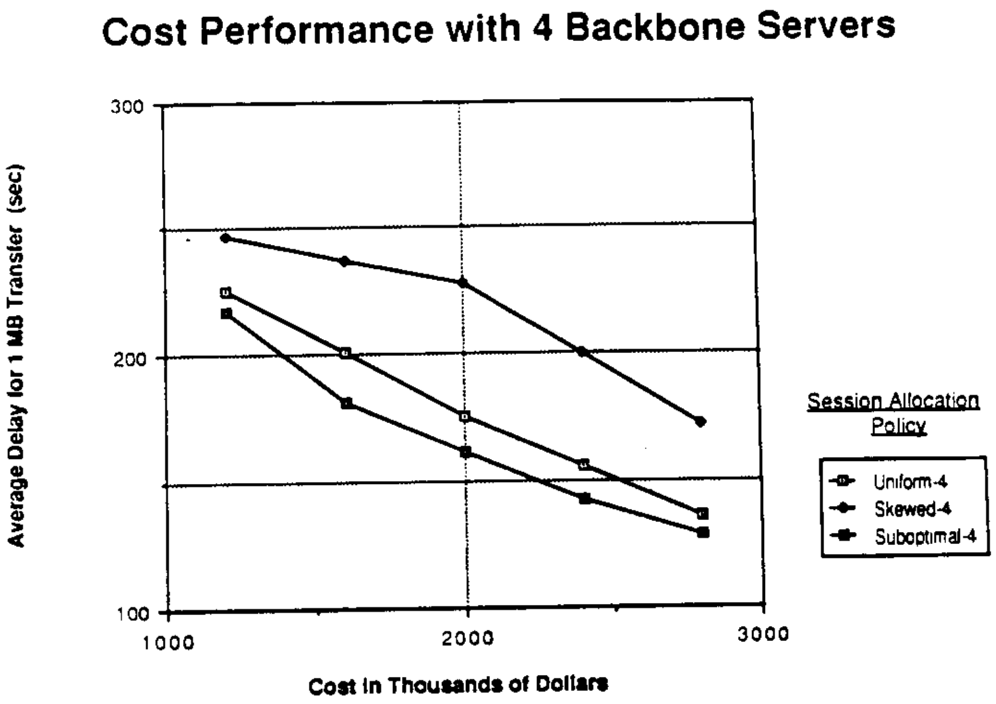


Figure 7.6: Cost Performance with 4 Backbone Servers

7.5.2 Cost Performance with 4,6, and 10 Backbone Servers

In Tables 7.4 and 7.5 we report the optimization results for 6 backbone servers in the range from 2.2 M Dollars to 3.0 M Dollars. Suboptimal session allocations are indicated for each backbone configuration set.

Figures 7.7 and 7.8 show the condensed cost performance curves for the optimization with 4, 6, and 10 sites. Solutions are defined for a spectrum of suboptimal cost and performance values. Once the constraints are given, a solution becomes readily available.

We can see that the best working range is located in the range of 2-3 M Dollars. This working range is defined by the knee of the curves. We note that the best workload configuration for 10 sites with 3.0 M Dollars is the uniform workload. This occurs because the 90 sessions workload distributed among 10 sites drives all backbone servers at very low utilizations. This makes performance less sensitive to load allocation in this capacity-overkill situation. Clearly, this is not a cost effective solution for the user load given in our example.

7.5.3 Computational Considerations

The synthesis of networks with discrete capacity is a time consuming task due to the concave shape of the objective function. In our case we have multiple user classes and backbone servers, as well as an unknown topology. An opti-

Site	*Backbone*	*Sessions in each class*		
		Int	*HI*	*CI*
cost of 2.2 M Dollars				
1	4381	16	5	3
2	4361	12	3	3
3	4361	12	2	3
4	4361	4	0	5
5	4361	4	7	2
6	4361	0	7	2
cost of 2.6 M Dollars				
1	4381	12	4	5
2	4381	12	4	5
3	4361	12	4	1
4	4361	4	0	4
5	4361	4	6	2
6	4361	4	6	1

Table 7.4: Suboptimal Workload with 6 Backbone Servers (part 1)

		Sessions in each class		
Site	*Backbone*	*Int*	*HI*	*CI*
cost of 3.0 M Dollars				
1	4381	12	3	6
2	4381	12	3	5
3	4361	12	4	1
4	4361	4	0	3
5	4381	4	8	2
6	4361	4	6	1

Table 7.5: Suboptimal Workload with 6 Backbone Servers (part 2)

mal solution using dynamic programming approach [Frank.etal69] would entail extremely expensive computations. Our approach iteratively finds a suboptimal solution that is as good as the computational budget available. In Appendix F we enclose the listing of the configuration synthesis program that we used for the computation.

The designer must allocate his budget to the subtasks of user workload characterization, the generation of the system delay tables, and the configuration synthesis optimization. Each of the phases is time consuming and computationally costly due to the very large state space. The optimization is quite efficient but it requires a number of iterations with different initial feasible flows until a good workload distribution is found. The uniform distribution produces good

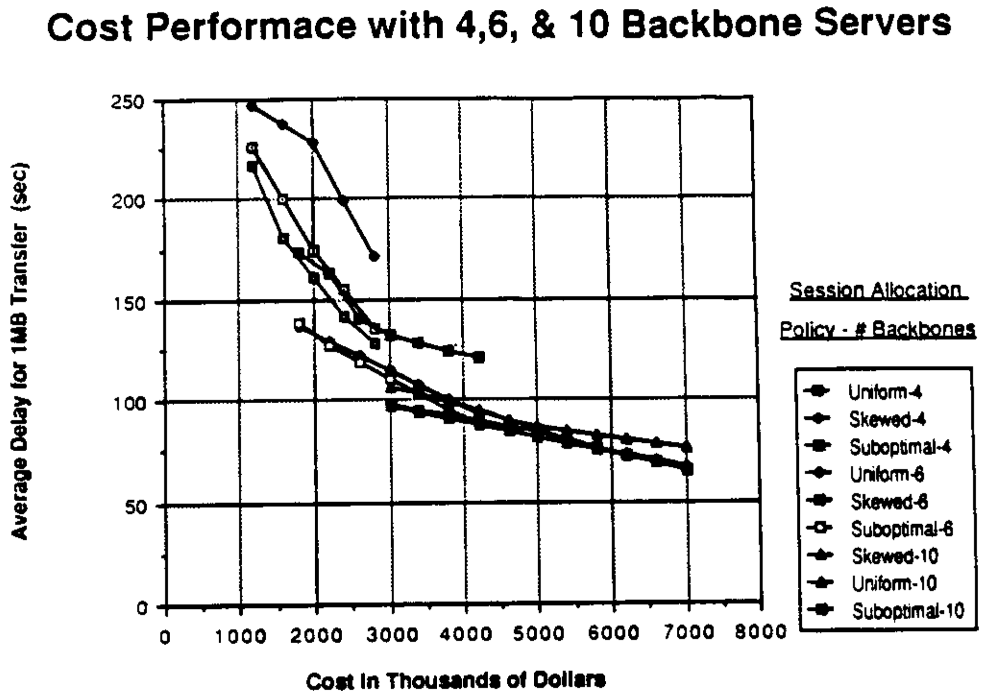
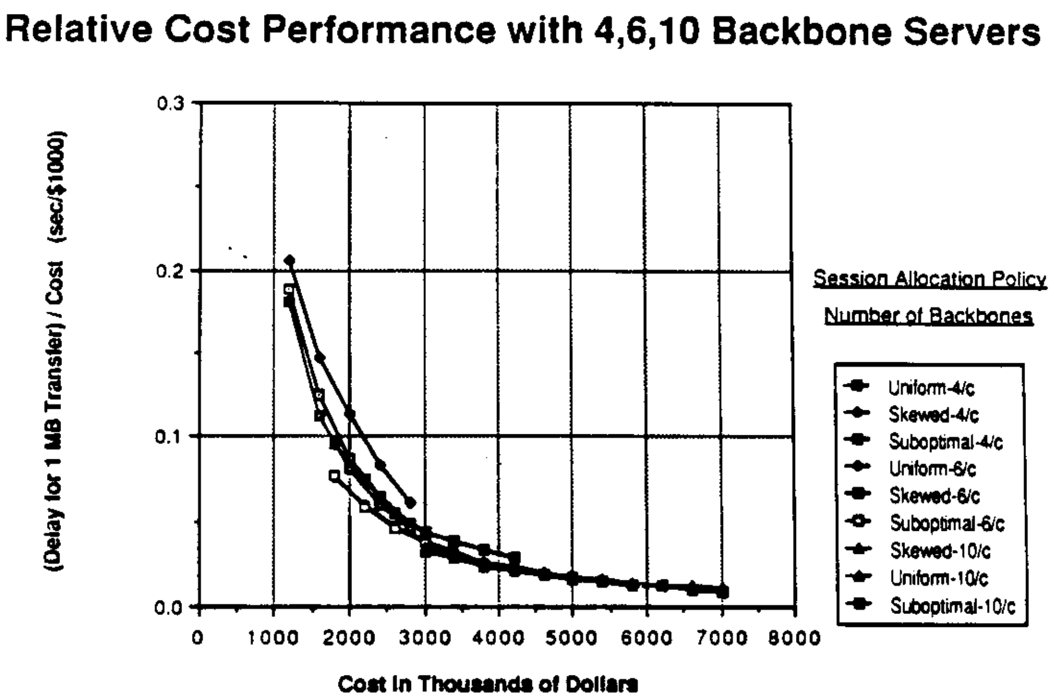


Figure 7.7: Cost Performance with 4,6 & 10 Backbone Servers



Relative Cost Performance with 4,6,10 Backbone Servers

Figure 7.8: Cost vs (Performance/Cost) with 4,6 & 10 Backbone Servers

initial results and should be the first to be compared with the given delay constraints.

7.6 Summary

In this Chapter we considered a rather complex configuration synthesis problem. Both the offered load and the computation/communication resources carry a high degree of richness and complexity. This makes an intuitive or straightforward engineering solution very difficult.

Through an extension of the CFA algorithm, combined with application to backbone computing resources, we derive a direct way to construct a performance-cost effective backbone network. This is accomplished through an optimization technique which deviates resource assignments in order to identify sub-optimal solutions to the problem.

We also note that heterogeneous resources are modeled by studying measured behavior, and by emphasizing the most dominant characteristics of the integrated system. The fact that we have measured and compared the physical system to predicted simulation figures gives us increased confidence in the presented results.

Unlike recent contributions to this area [TanTowWol88], we do not make restrictive assumptions with respect to service times for the user classes considered. We obtain stable numerical description of class behavior, and input these results into the optimization algorithm. This enhances the stability of the optimization algorithm, as possible instabilities in the simulation are handled in advance.

The convergence of the presented optimization algorithm is impressive, as solutions were obtained in less than 12 iterations for most cases, consuming about 1-2 minutes on a 68020 based microprocessor. Stability of the algorithm still needs to be studied, as in some cases there is divergence from local minima.

The presented configuration synthesis algorithms are very general, as they apply to resource allocation in the most general form. It is important to create an appropriate performance mapping from raw resources and load characteristics to the system under consideration. Once that is done, the optimization can be readily applied, resulting in efficient resource utilization.

It should be recognized that there is possibility to extend this work to obtain an automatic search on the resulting plots such as in Figures 7.7 and 7.8. Hence cost/performance criteria could be defined to suggest the best working area on the curve, depending on resource designation and financial capacity committed to the planned resources.

In Chapter 9 we shall elaborate on further research opportunities towards which this work might lead.

CHAPTER 8

Related Research

This work deals with performance evaluation, prediction, and system synthesis for large heterogeneous distributed systems. There exists related research in several sub-issues of the problem. One such issue is the design and development of such systems. other issues pertain to research done in performance evaluation of various systems. In particular, issues of system synthesis, resource assignment, and load distribution are of particular interest. In this chapter we shall briefly review work done in the various categories, and comment on some relationships to our work, which comprises of a marriage of the mentioned disciplines.

8.1 Large Heterogeneous Distributed Systems

The natural expansion of networks into "Meganets" has created new domains of interconnected machines. Most of these machines are loosely coupled, e.g. only electronic mail and remote login typically may take place. We are not directly concerned with such systems. Our interest is in distributed systems which are more tightly coupled, and present a high degree of transparency. Of the several efforts in existence, we shall mention the most significant ones which are related to our work.

8.1.1 Vice/Virtue/Andrew

This Carnegie Mellon effort [SaHo.etal85], [MoSa.etal86], [HoKa.etal87] is an interesting effort. It has begun in 1983, and in 1985 was becoming stable, and distribution began outside the development group. The main emphasis of the project was to develop a distributed file system, and a computing environment for academia. The effort is an immense investment, equivalent to 150 man-years. It is targeted to inter-connect the CMU campus by the end of the decade. Andrew uses the notion of clusters which contain individual servers and several workstations. There exists a limited degree of replication, and a noticeable degree of heterogeneity. This effort is sponsored by IBM.

The main shortcoming of this project considering it in ralation to our work, is the lack of analysis and synthesis tools that are based on queueing networks. Most of the emphasis in the project was directed towards high level design, and detailed implementation. Performance issues were dealt with by measurements and system enhancements resulting from experience and intuition. Quantitative simulations of queueing models were not used in any of the Vice/Virtue/Andrew studies we have seen from CMU.

8.1.2 The V Distributed System

The V distributed system [Cheriton88] was developed at Stanford University as part of a research project to explore issues of distributed systems. Aspects of the design suggest important directions for the design of future operating systems and communication systems.

Measurements conducted on V demonstrate file service in the range of 160 KByte/sec, with inter process communication faster yet. Most of the emphasis of the development is on the kernel V communication protocol. There is transparent address space with shared memory implementation. Quite a few of the workstations in system V where diskless workstations, and some performance modeling was done for that particular aspect [LaZaCeZw84]. Our emphasis has been on workstations with local disk storage utilization[1].

8.1.3 The Washington Heterogeneous Computer Systems (HCS)

This is a relatively fresh effort by the University of Washington [NoBlLa.etal88]. The main services provided by HCS are file access, mail, and remote computation. This effort presents interesting solutions for local network mail, naming service, and remote tasking. However, the system is undergoing early development, and is mostly a prototyped system. Its performance will probably improve with time, as performance analysis of this system was not reported in [NoBlLa.etal88].

8.1.4 SUN Network File System (NFS)

NFS [SUN86] is an industrial product that is gaining increasing popularity. This system allows several servers and clients to transparently access network file systems. This system has limited file replication, and uses stateless file servers

[1] Our system provides for *copy protect* mode of operation, in which software can be brought to the workstation's memory, but not copied onto media

for performance and easy recovery. NFS is compatible with Berkeley Unix 4.3, but will support a variety of architectures and operating systems. It defines an external data representation (XDR) which is used for all network communication. We have not seen performance analyses of NFS as a queueing network.

8.1.5 Other Efforts

We mention here additional efforts which are less related to our work, but are nevertheless important contributions to distributed computing.

The Cedar Distributed File System [GifNeeSch88] is a transparent environment built at Xerox PARC in order to allow programmer cooperation in program development. Atomic whole file transfers are predominant in this non-recent effort.

Argus [Liskov88] was developed at MIT to allow distributed programming. Most of the work here concentrates around nested transactions, and atomic commits. The implementation does not support high degree of heterogeneity.

Sprite [Ousterhout.etal88] is a recent effort at Berkeley. Diskless clients and servers use code sharing. Intensive caching employed throughout. Very efficient remote operations (500+ kbytes/sec) are reported. Measurements were done, but analysis is intuitive.

We conclude this subsection with reemphasizing the fact that very elaborate implementations of heterogeneous distributed systems are under development. Indeed, performance is a major parameter for all these systems, and most work done with respect to performance lacks the use of extensive simulation and other approaches that are beyond system engineering experience and intuition.

8.2 Related Research in Performance Modeling and Analysis

Following is a discussion of currently available modeling techniques, along with an assessment of their relationship to our study.

8.2.1 Analytical Methods

Deep analytic understanding of systems and processes is indispensable. Analytic solutions not only provide us with the best understanding of the systems under investigation, but often times give very efficient and fast numerical solutions to difficult problems. This is due to the fact that analytic solutions tend to be clean, closed-form, and easy to evaluate for varying system parameters. Recent examples of noteworthy contributions in this area are [HeidLaks87] and [NelTowTan87].

Granularity, however, is a key issue, especially when systems are complex. At times systems need to be reduced into smaller, more tractable sub-systems (more in Sec 8.2.4). One of the drawbacks of pure analysis, is that often times, the restrictions that analysis impose, create a system that is too far from any realistic system at hand. This is the general problem of theory versus application. In fact, often times detail is missing for a complete and elaborate system description. When this happens, approximations need to be made.

8.2.2 Approximations

While analytic solutions have their clear advantages, it is extremely challenging to model complex systems accurately and at the same time keep the solution tractable. It is for this reason that approximations have been introduced. The nature of approximations is to identify the *key* governing forces and trends within a system, and make conscious *intelligent* neglections of less important factors. This creates much simpler models with respect to analytic ones. They can be even faster to solve [Lave83], [LaZaGrSe84]. However, one must realize that accuracy is being sacrificed and the tradeoff should be assessed to determine the value of the approximation. It is *very* helpful to be able to *bound* the errors introduced by the approximate analysis. We have previously used approximations [BetGerPop84] successfully. It should be noted that in our current work we used approximations for simulations (Sec 8.2.5), as opposed to analytical approximations.

8.2.3 Queueing Networks

In general, computer networks can be modeled as queueing networks [Kleinroc76], with the computing, storage, and network being modeled resources, and the jobs being the customers in the queues. A comprehensive discussion on this approach is given in [Lave83]. Recent publications in this area can be found in conference proceedings of Performance, Sigmetrics, Computer Communications, and in the established journals in the discipline.

Substantial work has been done in the area, and there are several commercially available packages such as RESQ, PAWS, and SIMSCRIPT. In fact we have defined the queueing network which represents SEASnet. This queueing network was the key to our modeling effort, and was used to evaluate performance as well as synthesize new configurations of balanced load.

8.2.4 Decomposition

It is intuitively appealing to decompose complex architectures into smaller, easier to tackle subcomponents. In our case, this effort was influenced by seeking agreement with the experimentation results for a first order model. This goal of isolating parts and tasks of the compound system, which can be aggregated safely from the external point of view, is addressed also in [SaueMacN83], [LaZaGrSe84].

In fact we have found out that backbone server service is quite independent of other backbone servers. In our modeling efforts as well as in our measurements, we could successfully deal with one backbone server at a time, such that the complexity of the problem could be greatly reduced. It is also possible to address components within the hierarchy as sub part of the network, and obtain better representation for these elements. We have spent considerable effort in optimizing the representation of the backbone servers to accommodate for buffering, process space, and i/o service.

8.2.5 Simulation

System simulation has always been a methodology technique that enabled the modeler to make major economical savings in the design and implementation of such systems. In fact, even in the face of incomplete understanding of the analytical theory governing system behavior, simulation provides answers to important questions regarding system performance metrics and predicted behavior. This is true for several scientific and engineering disciplines.

In the case of computer system and network modeling, many such answers can be provided by means of discrete event simulation of queueing networks [SaueMaN83], [LaZaGrSe84], [MacNSaue85]. Recent publications in this area are conference proceedings of Performance, Sigmetrics and [MorSouSoa87].

We have opted to use available simulation packages, namely RESQ. Other packages are PAWS and SimScript, and our own special purpose software that is being developed at UCLA. An instance of the latter is provided by our work on distributed simulation of data communication networks [CheCarKar86], which utilized time-warp methods and exploited the distributed processing support provided by LOCUS. In fact, our current configuration synthesis work employed additional special purpose software that we constructed. This is reported in more detail in Chapter 7 and Appendix F.

8.3 Configuration Synthesis, Load Balancing

These are more specific areas in which there exists a diversified body of work. We will briefly mention some of the relevant efforts in both sub areas.

8.3.1 Configuration Synthesis

The problem of resource assignment is a general design problem and is addressed in several contexts. Traditionally this problem received treatment in network design problems, such as Capacity and Flow Assignment problems [Fox66], [FraGerKle73], [Gerla75], [Kleinroc76]. Most of the traditional applications dealt with traffic/flow assignment within various topologies. Some recent contributions in the area of load balancing deal with problems of assigning file service, although most of these deal with homogeneous backbone servers.

8.3.2 Load Balancing

Load balancing is traditionally an issue of task assignment by terminals to a distributed system consisting of several machines [DeSoGerl84, 87]. These problems typically deal with either static or dynamic strategies based on various optimization criteria. Most of the work is analytical, with some hypothetical examples for illustrative purposes. Usually there is no direct correlation or measurement relationship with an actual installed system in place.

Some recent contributions [TanTowWol88], deal with actual file service and session assignment problems in computer networks. However, they make restrictive assumptions with respect to service times for the user classes considered. We, on the other hand, obtain stable numerical description of class behavior, and input these results into the optimization algorithm. This enhances the stability of the optimization algorithm, as possible instabilities in the simulation are handled in advance.

8.4 Summary

We have reviewed other efforts in various subparts of our general work reported in this dissertation. While there are several on going efforts within the various sub disciplines reviewed, it is the unique contribution of our work to incorporate the thrust of these several disciplines into a research effort utilizing as a testbed system a novel academic computing system (SEASnet) that we have in operation.

As a result, the bodies of measurement, evaluation, and synthesis were integrated into what we have presented as a novel research program in the modeling of large heterogeneous distributed systems.

CHAPTER 9

Conclusions and Further Research

In this chapter we summarize the contributions of our work and their significance. We then conclude with an elaboration on future research directions that spawn off this work.

9.1 Contributions

Our work is an interesting combination of a system study and a performance modeling effort for a large heterogeneous distributed system. We used as a testbed a novel system - SEASnet which was installed at the UCLA School of Engineering. While SEASnet is an example of an academic operation, our methodology generalizes to other heterogeneous distributed systems as well. In spite of the fact that SEASnet exhibits a significant degree of system complexity, we found an approach that yielded a good first order approximation and proved useful for performance prediction and further system synthesis.

There are four distinct contributions in this work, namely the *User Model Concepts, Measurement Approach, Modeling Enhancements, and Configuration Synthesis*. We outline the contributions in the following sections.

9.1.1 User Model Concepts

Recognizing the immense richness of disciplines, applications, and user characteristics for such systems, we arrived at a tractable model at a level of granularity that expresses the qualities of interest for our modeling purposes. It was our effort to concentrate on the key behavior concepts of the user population, and identify the most significant patterns for the goal of user model construction. In a heterogeneous workstation-server environment (with a testbed of AIX/370 backbone servers and PC/DOS PCs connected via PCI Bridge), we have defined 3 user classes, whose combinations can represent a large number of user loads that are of interest. These user classes are Communication Intensive (CI), High Interactive (HI) and Interactive (Int).

9.1.2 Measurement Approach

We have generated a large measurement base that describes several aspects of the testbed systems we were evaluating. The main goal was to characterize and parameterize such a Large Heterogeneous Distributed system, as a required pre-modeling step. During the process, we crystalized the modeling concepts that were evolving, such that the iterative research process was progressing. Important bottlenecks were determined, and key system behavior patterns were observed.

One of the main challenges of the work was the fact that access to internal system parameters was limited, and we had to rely on global performance parameter measurements. This required a substantial amount of measurement and verification, especially at the first stages of the work, when we has less knowledge of the system.

Measurement were conducted for all classes of work load. In addition, some measurements for internal system parameters were carried out. We have used many of the measured results later in the dissertation. However, there are additional results which were not directly applicable for us. These might be used in future research efforts; i.e., our measurement effort can seed further work.

9.1.3 Modeling Enhancements

Modeling heterogeneous distributed systems is a difficult task. This problem can be made significantly more tractable by good determination of first order effect versus second order effects. We have successfully identified the key elements that constitute the backbone network for PCI file service, and constructed a system model that captures its behavior. We have tested this model for a variety of system configurations, user classes, and load patterns. Good predictive capacity was demonstrated throughout.

In spite of the system complexity, our model is relatively simple, and its utility proved to be very promising. The model was simulated with a general purpose queueing network simulator (RESQ). This is the model which was subsequently used in our synthesis efforts.

9.1.4 Configuration Synthesis

We utilized our queueing network model (of our characteristic heterogeneous system), to arrive at a new synthesis methodology for a system of backbone servers and PC users (PCI sessions), under various given sets of user loads and cost constraints. We proposed and utilized a new capacity and flow assignment (CFA) algorithm that assigns user sessions onto heterogeneous server networks. We also obtained a balanced load description for an optimal configuration. This simplified algorithm proved to be very efficient computationally, as well as being effective in generating configurations and cost/performance trade-offs. This algorithm also presents the seeds of future load balancing algorithms, as load balancing via session assignment was demonstrated.

9.2 Significance of the Work

The four contributions mentioned above constitute a major thrust in the understanding, evaluation, modeling and synthesis of large heterogeneous distributed systems. In contrast to other efforts (chapter 8) which deal with some sub part of the general performance problem of such heterogeneous systems, we took a global approach. We studied the user behavior in detail, as well as the general context of such systems. We determined general behavior patterns for such users, and obtained a significant body of measurements. These measurements could be used in other research efforts as well. We then constructed a very general model for the backbone to customer relationship. This model could be applied to other heterogeneous systems, once parameterized. In conclusion, we have combined all the results to demonstrate not only predictive capacity, but also synthesis capacity, for which new algorithms were derived.

9.3 Directions for Future Research

The results presented in this dissertation open several avenues for future research in a number of important directions.

9.3.1 Model Refinement

Additional effort could be invested to further refine this model, and include second order effects, to enhance the accuracy of the PCI-Backbone model. Additional system parameters such as task scheduling, cache optimizations etc, could be incorporated into the model. This would require additional information about system internals which may not be readily available. Potentially, additional measurements and design data would provide this information. Components within the model could be decomposed and the additional parameters incorporated therein.

9.3.2 Novel and Enhanced User Classes

There are growing user populations of new classes on SEASnet as well as on computing systems throughout the world. The most intersting class of interactive computing is the window based user community, such as those using X-windows. Typically, users require multi session capacity on window based workstations. This user class needs to be studied in detail, measured, and characterized. The resource requirements for these classes could then be incorporated into an extended system model

9.3.3 Load Balancing

In the synthesis work we are configuring a system such that cost is optimized. Part of the algorithm deals with good utilization of the resources. It would be even more challenging to maintain a balanced load situation for a variety of user loads in the future.

1. *Static allocation.* In this case we address the pre-allocation of jobs to a given configuration. A new job is assigned according to mean time statistics.

2. *Dynamic real-time.* This is a fine grain assignment scheme based on real-time monitoring of all jobs in operation. This scheme would not monitor a job once assigned, as only new jobs are assigned [EagLazZah88].

9.3.4 Distributed File Access and Assignment

The availability of data has two vastly varied flavors. Local data is accessible directly within a machine. Remote data requires all the communication/network-protocol overhead to become available. The trade-offs of replicating vs remote access or process migration constitute a rich array of possible extensions.

9.3.5 Reliability and Availability

Another aspect of design optimization relates to performance costs due to higher reliability. This price could be traded off with reliability design constraints to achieve the final resource allocation for the system.

9.4 New Trends in System Research

The quality and availability of computing systems is evolving at an astonishing rate. Desktop systems today are doing the work of last decade's mainframes. These machines are networked to yet more powerful servers which provide file service as well as cycle service. These computationally intensive services combined with the graphic capability of this new generation of desktop machines present an ever increasing modeling challenge. A rich array of performance issues is becoming available for tomorrow's researchers.

APPENDIX A

SEASnet Documentation

This Appendix contains additional information about SEASnet. The primary document is a progress report from Dr. Michael Stenstrom, SEASnet's Academic Director, to the UCLA Academic Computing Council.

Also included are some hardware date sheets, pertaining to performance of SEASnet processors and I/O devices.

SEASnet - A Progress Report

for the First Two Years

April 1987

Submitted to the
UCLA Academic Computing Council

by

Michael K. Stenstrom
Academic Director, SEASnet

INTRODUCTION

During the past three years computer usage in the School of Engineering and Applied Science has changed dramatically. We have evolved from dependency on a single large mainframe facility (OAC) to workstation-based computing supported by a school-wide Local Support Center and two departmental facilities. Undergraduate access to computer resources has increased at least ten fold over previous usage at OAC. This dramatic evolution has occurred within two years and has been made possible primarily through several large grants: two from IBM, one from AT&T, and one from Hewlett Packard.

This report was written in response to a request to evaluate the impact of the largest of these grants, IBM's AEP grant, Project Advance. The other grants which have impacted our efforts to develop a workstation based computing network are also described, but in less detail. The report describes the distribution of machines and their impact on instruction.

HISTORICAL COMPUTING USE IN SEAS

FACILITIES

Prior to Project Advance, computer usage in SEAS was restricted to three large facilities and a very few minor facilities. OAC was by far the major supplier of computing cycles to Engineering, with all departments having access. There were approximately thirty 3270 style terminals in Engineering with access to OAC.

The Computer Science Department operated a Unix-based network of VAXes and PDP-11's which were generally available only to their graduate students and faculty. At their peak they operated 21 VAXes (one 780 and twenty 750's) with more than half of them dedicated to research groups in Computer Science. Professor Popek's research group is one example.

The Manufacturing Engineering Program, housed primarily in the MANE department, obtained an IBM 4341 in November, 1983 from IBM under a Manufacturing Engineering grant, a number of graphics terminals, several microcomputers attached to robots, and a large number of specialized software packages. This facility runs VM/CMS, and its operation has not been dramatically affected by Project Advance. Project Advance has provided network access, and competition for the Manufacturing Engineering computer was less pronounced than it would have been had there been no Project Advance.

Other smaller efforts included SEAS Administrative Computing which started with a single PDP-11 and has evolved to two VAX 11-750's. Although these machines were intended primarily for SEAS administrative computing, they were for a long time the only source of Unix for many faculty; consequently, a number of faculty outside the Computer Science Department used these machines to prepare educational materials. However, no "hands-on" student access was permitted.

A small number of micros and mini's existed in the school, and a few examples are summarized as follows:

1. A Sol-8 eight-bit (S-100 bus) based hardware laboratory in the Computer Science Department.
2. Several PDP-11's in Computer Science (e.g., Prof Bussell's hardware laboratory, Prof. DiStefano's Biocybernetics facility)
3. A PDP-11 in the MANE Department used for real-time data acquisition.
4. Professor Martin's PDP-11/44 instructional laboratory in Electrical Engineering
5. A Vector Graphics 8 bit CP/M classroom in Computer Science.

These facilities had varying success. The machines which were closely associated with researchers generally fared the best, since there was considerable motivation and resources to use the machines. The Sol-8 hardware laboratory was used extensively in Computer Science. The Vector Graphics laboratory was generally a failure due to the limited power of CP/M and 8 bit processors, and the room was mostly used for instruction not requiring computer access.

At about the time Project Advance started for other parts of the Campus, which was about a year earlier than it began in Engineering, several new computer projects were started in Engineering. These included:

1. The Artificial Intelligence Laboratory in Computer Science, based upon 25 Apollo's in a token ring network. (acquired with Keck Foundation grant funds).
2. Electrical Engineering's purchase of a Pyramid 90X supermini.
3. A donation from Gould of a PN-9080 supermini.
4. Purchases of small, multiuser micro's, such as MANE's purchase of a Fortune Systems 68000-based Unix machine. These machines were small by current standards (+ 3 years), but were significant departmental purchases.
5. A Hewlett Packard grant of thirty 68020 based workstations to support Artificial Intelligence (AI) and Very Large Scale Integration (VLSI).

Planning for these projects was for the most part not affected by Project Advance. Undoubtedly had we known the extent of Project Advance we would have done many things differently. The eventual use of these early machines was dramatically affected by Project Advance; for example, the Sol-8 based machines have been replaced by PC's, and the Fortune Systems machine became a "white elephant" very quickly with users preferring DOS machines.

Finally there were some "high-end" users who had access to networked Cray's (e.g. MFEnet). These users were strictly research users having little impact on the classroom-based educational programs.

NETWORKS

Prior to Project Advance there were no networks in SEAS, other than Computer Science's VAX network. The first ethernets in CS were introduced to provide terminal access to VAXes (Bridge Communications CS-1's) and to network the experimental Locus VAXes. The Apollo token ring in Computer Science was started about the time we were planning SEASnet. Other machines were connected in star configurations (terminals to timesharing machines, which is not considered a network for the purposes of this document).

QUANTITATIVE ASSESSMENT OF PROJECT ADVANCE

IBM GRANT EQUIPMENT

During the planning of Engineering's share of Project Advance we envisioned 175 workstations and a single 4381 server. At the time of the planning the PC-AT was only a rumor and the RT was unknown, except that we expected IBM to compete with other companies making scholar's workstations. We made plans around 115 workstations based on the 80286 CPU and 60 scholar's workstations. Three classrooms of 30 workstations each were planned.

The allocation of machines to departments within the school was deliberately not included in our initial planning. We envisioned that the allocation of workstations from the School's pool would be made to the departments by the Dean based on his assessment of each department's educational plan. Departmental educational plans were initially described in our first proposal[1] and were discussed in greater detail in a proposal to the Office of Instructional Development (OID).[2-4] The Dean purchased 10 PC-AT's for use by faculty at home and were allocated on the basis of each department's educational plan. Table 1 shows the distribution of IBM equipment.

AT&T GRANT EQUIPMENT

After a lengthy negotiation period AT&T donated eighty-five 6300 Plus PC's (an
80286-based workstation, very compatible with an IBM PC-AT), one 3B15 and ten 3B2/400
servers. These machines were configured in such a way that they would complement the exist-
ing workstations and network, and produce an environment 100% compatible with our existing
environment using PC-AT workstations. The workstations were allocated to departments in a
similar fashion as the IBM equipment, except that departments were required to create "clusters"
of machines in small rooms that would be available to graduate students and other "good
citizens" of the department on a 24-hour basis. Table 2 shows the distribution of the AT&T
grant equipment, and Table 3 shows the combined distribution.

Table 1. Distribution of IBM Workstations*

Department	Convertibles	AT's	RT's	Total
CS	1	30	13/9**	44
ChE	1	7	1/1	9
CE	1	9	3/1	13
EE	1	12	3/0	16
MANE	1	19	5/2	25
MSE	1	5	1/0	7
Classrooms, Staff***	4	74	42	89
Total	10	156	68/24	192

* includes 10 purchased by the Engineering Dean

** the RT column shows allocated machines/delivered machines, at the time of this writing.

*** includes a pool of machines used as spares to replace classroom machines being repaired, and machines to be delivered to departments.

Table 2. Distribution of AT&T Grant Workstations

Department	Clusters	Labs/Offices	Total
CS	7	5	12
ChE	4	2	6
CE	5	2	7
EE	8	3	11
MANE	8	3	11
MSE	3	2	5
Classrooms, staff	33	0	33
Total	68	17	85

Table 3. Distribution of Grant Workstations

Department	AT&T	IBM		Total
		PC-DOS	UNIX*	
CS	12	31	13	56
ChE	6	8	1	15
CE	7	10	3	20
EE	11	13	3	27
MANE	11	20	5	36
MSE	5	6	1	12
Classrooms, Staff	33	78	42	122
Total	85	166	68	277

* allocated as of this writing. Additional Project Advance funds are required to complete this allocation.

EXPENDITURES AND NON-GRANT EQUIPMENT

Virtually all of the networking equipment and all the VAXes were obtained through University resources. SEASnet purchased one used VAX-11-750 expressly for the purpose of allowing non-CS users terminal access to the servers. Computer Science's VAXes existed prior to the grant, and Extension purchased a VAX-11-750 to allow their students to access the network. It was necessary to equip each PC-AT with an ethernet card at a cost of $500. Locus Computing Corporation (LCC) donated 150 PC-Interface licenses. For the early RT's we purchased ethernet cards; later the IBM grant was able to supply ethernet cards. Appendix A shows the equipment expenditures for non-Grant equipment and personnel. UCLA has spent $2.5 million on SEASnet over the last three years. Additional FTE within the School are used for computing that are not associated with SEASnet. Approximately 3.5 FTE in Computer Science, 1.0 in Electrical Engineering, and 1.5 in MANE are associated with computing and are paid from University resources.

CLASSROOM AND NETWORK UTILIZATION

Appendix B shows the classroom utilization over the periods from the Fall Quarter, 1985 to the Winter Quarter, 1987. The courses offered by departments for each quarter since the SEASnet classrooms were opened are summarized in Table 4. The majority of the impacted courses are upper division; only a few graduate and lower division courses have been regularly scheduled in the SEASnet classroom.

Table 4. Sections Taught in SEASnet Classrooms or Using SEASnet Classrooms, by Department

Dept	F-85	W-86	S-86	F-86	W-87
CE	4	4	1	2	8
ChE	0	1	1	0	2
CS	2	5	5	3	8
EE	0	2	2	2	4
MANE	7	6	3	6	3
MS & B	0	0	0	0	2
Totals	13	18	12	13	27

It is difficult to recognize trends from Table 4; however, some explanations are offered. Initially the classrooms were made available to instructors in addition to their regular classrooms. This policy continued until the end of the Spring Quarter, 1986. At that time it was felt that space was being wasted and we changed our policy.

Beginning in the Fall of 1986, one classroom was kept open during the weekdays from 10 AM to 10 PM and was open to all students with accounts, on a space available basis. The other classroom was used for classes which were regularly scheduled *only* in the computer classroom. This policy was very successful, allowing many more students to utilize the rooms. The new AT&T classroom is also being used as an open access classroom. A number of professors now make assignments and expect students to complete them using the open-access SEASnet classrooms. They provide no guidance except for helping the students acquire accounts.

A second factor affecting classroom use was network and server performance. The server performance was generally poor in the 1985-1986 academic year, with frequent crashes and very poor response time. This problem resulted primarily from the difficulty of attaching ethernets to IBM 370 Channels, and from our inability to adjust the priority in which programs execute under Locus on the IBM 4300's. A major improvement to network and server performance was made in August, 1986, which resulted in improved stability but only marginal improvement in network response time. A second major change was made in December, 1986, which markedly improved network performance and server response time.

Figures 1 and 2 show the number of users and system loads on the Locus machines. The "Total" line refers to the sum of the PCI and time sharing users on the two IBM 4300's and 6 VAXes. Most of the time sharing users are faculty and staff. The dotted line shows the PCI users who are generally all the students using the classrooms. The dashed line shows the machine CPU load for all servers. The Unix CPU load figure is a representation of the number of CPU-bound jobs averaged over a one minute interval. A highly CPU-intensive job, such as simulation performing many floating point calculations, would create a load of slightly less than 1.0. An editing session would produce a load of approximately 0.1, except when the file is being rewritten to disk. Loads of 3 or 4 generally produce noticeable lags in response time on a VAX-11/750. Loads of 5 or 6 generally cause the 4381 to respond sluggishly to the network.

Figure 1 shows a high load week proceeding exam week in the Fall of 1986. At this time the network was stable but we had not yet implemented the new changes which increased response time. The maximum allowable users on the 4381 was 32 in this time period. Figure 1 shows that we were nearly constrained by this upper bound. In practice it was not possible to connect more than about 25 PCI users to the 4381 because of sluggish response time. Figure 2 shows the average (10AM to 10PM) network loads over the Fall Quarter, 1986. From a practical viewpoint, we were constrained in the Fall Quarter by server performance.

Figure 3 shows the utilization of our open access classroom during the Fall Quarter, 1986 and Winter Quarter, 1987. The numbers represent head counts by the proctors. The number of students using the machine exceeded the number connected to the servers, indicating that many users were operating without logging into the net. This is possible and quite workable since all machines have local hard disks. On a number of occasions the number of students using the classroom exceeds the number of machines that are available, indicating that students were working together in groups of two or more.

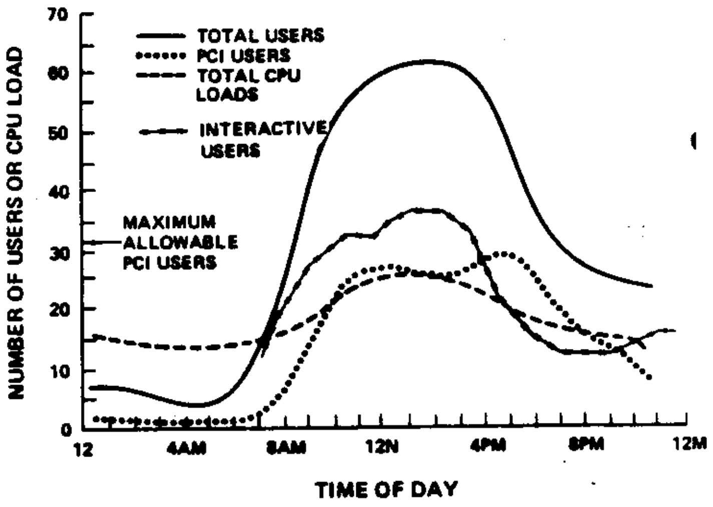


Fig. 1 Average Daily Network Load versus Time of Day (five day average during a week near the end of the 1986 Fall Quarter)

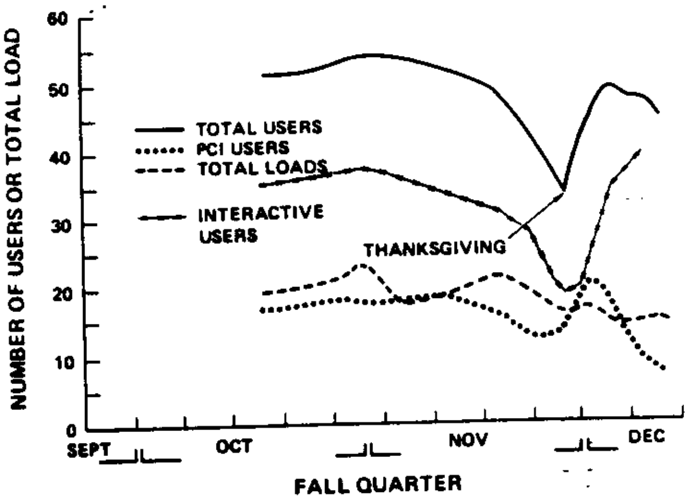
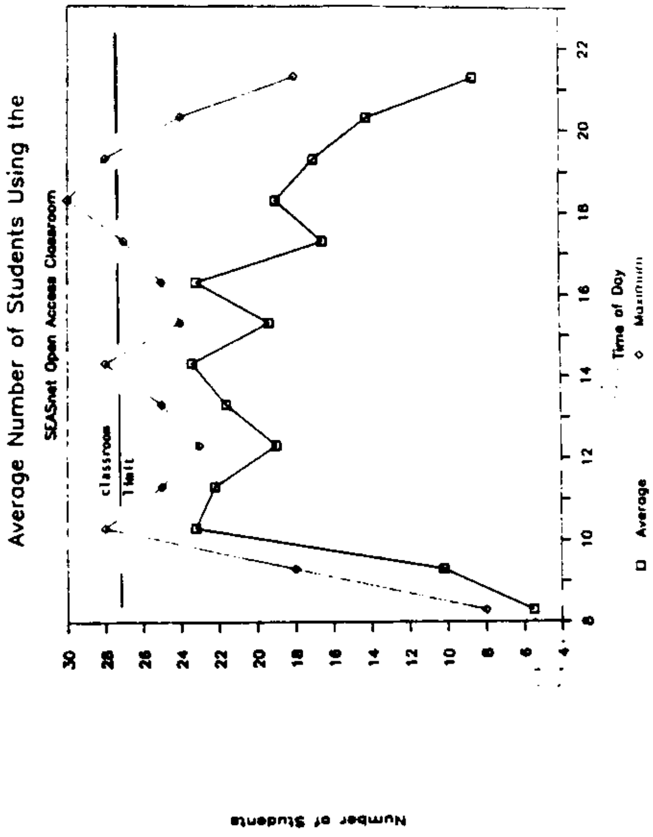


Fig. 2 Average Daily Network Loads versus Day of the Quarter (1986 Fall Quarter)



' ure 3. Average I' ·ly Classroom Ut—tion (Oct 1986-March 1987)

It is not possible to exactly summarize the utilization of classrooms. A fair statement of utilization for the open access classroom is shown in Figure 3. The tables in Appendix A show the utilization of our scheduled classroom. The reserved hours for the 1985-1986 Academic year are only approximate, since the instructors may have had more than one room available to them; for the 1986-1987 year instructors had only the SEASnet classroom which means that the room was utilized as shown in the tables. The total number of enrollments in SEASnet classrooms for the 1985/86 year was 1108 and increased to 1876 in 1986/87, or an increase of 69%. This compares to the total school enrollment of 15,125 for all of SEAS in 1986/87. Approximately 12.5% of the students enrolled in SEAS use the SEASnet classrooms as a part of their formal instructional program. The largest inroad was made into upper division courses, where 13.7% of the students used SEASnet classrooms.

The activities being performed in the classroom are also impossible to exactly quantify, but are primarily recitation, with TA's and instructors working with students; some instructors also lecture in the classrooms. The open access rooms are almost always being used by students completing homework assignments or working on projects.

SOFTWARE ACQUIRED THROUGH SEASnet

Forty eight different software packages were purchased or acquired by SEASnet during Project Advance. A great deal of IBM software was acquired from the Grant. OID funds were used extensively for acquiring special purpose software not available on the Grant. The most general piece of software acquired with OID funds was Turbo-Pascal. Appendix C lists the software and shows a number of site licenses. This list has been restricted to those products which were primarily sought by SEASnet (software licensed by the MIC, which also happens to be used at SEASnet, is not shown). Many SEASnet users take advantage of the products made available through the Microcomputer Information Center (MIC), such as SAS, T_EX, and BMD; we do not function as a liaison between SEASnet users and the MIC, although we will occasionally make a product available over our network for which the MIC acquired the site license.

A great deal of software has been acquired by faculty in various ways and in varying quantities. Many faculty routinely exchange software through their graduate students. It is not possible to quantify this informal exchange.

No software standards have been adopted by our academic units or SEASnet, other than some broad guidelines and selection of particular products for particular courses. The availability of IBM grant software has created some defacto standards, e.g., Professional Fortran is used by most PC-DOS Fortran programs at SEASnet.

We are supporting PC-DOS and several flavors of Unix. We are not supporting VM/CMS, MVS, VMS or other operating systems. OAC is still considered the supplier of VM/CMS and MVS cycles and software to Engineering (other than the Manufacturing Engineering Program), and we have provided networking to OAC in anticipation of their continued role in computing for Engineering and Computer Science. OAC will continue to serve SEAS in several areas, including CPU-intensive programming, SAS, and other mainframe applications.

We also hope that it will be possible for them to run a suitable version of Unix at OAC under VM once it becomes technically feasible to do so.

We are attempting to provide similar tools for PC-DOS and Unix environments. For example, we are attempting to make PC-DOS editors function like Unix editors, with the objective of making the transition of PC-DOS users to Unix as easy as possible.

We still believe that Unix can satisfy most of the needs of engineering/computer science users. The largest shortfall of Unix and particularly Locus is the poor compiler quality. FORTRAN is particularly poor. Our initial interest in PC-DOS was created primarily by our inability to obtain an adequate number of Unix workstations from Project Advance. As a result of this experience we have discovered that PC-DOS and an AT's computing power are adequate for many engineering requirements. Additionally, our transparent network bridge PC-Interface has provided an excellent way to integrate PC-DOS into our environment. At present, the 640K memory appears to be its biggest drawback for non-Computer Science users. Computer Science users have not found PC-DOS as useful, and most instructors would quickly convert to Unix if the hardware and support were available.

NETWORK CONFIGURATION

Figure 4 shows our network configured as of (3-1-87). The network is now available at all places in Engineering. Users with appropriate accounts can access all the machines using a single flavor of workstation and a single connection.

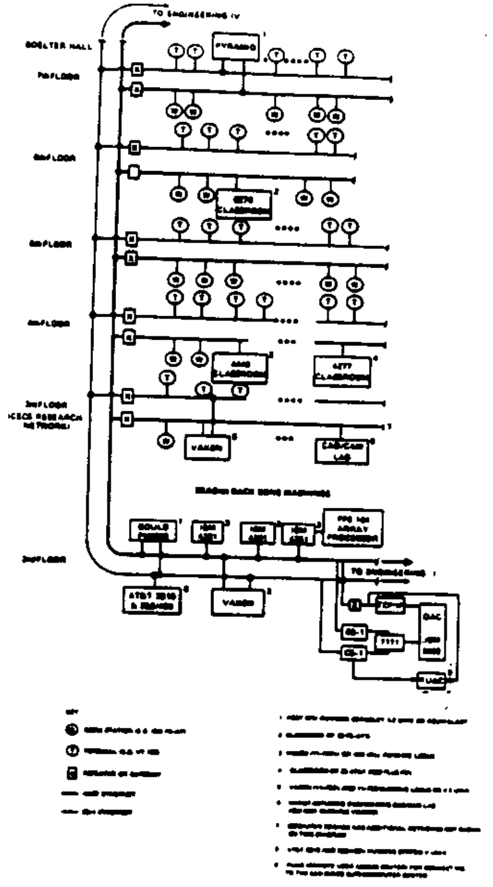


Figure 4. Network Configuration

QUALITATIVE ASSESSMENT OF PROJECT ADVANCE

Our original Project objective was to provide computer based instruction to engineering students. To accomplish this objective we required a network and open access computing for engineering students and faculty. For practical purposes we developed such a network and have accomplished our objective. We have networked over 275 workstations. Any undergraduate student enrolled in a course requiring computing has access to three PC-DOS classrooms (2 IBM and 1 AT & T). There are over 100 workstations in departments for graduate students and faculty. Several medium sized computers are available for problem solving, and the FPS-164 MAX is available to faculty and graduate student researchers, and for "high-end" educational projects. All of these users have access to electronic mail and OAC, and other locations, such as the San Diego Supercomputer Center and the ARPAnet, if they have permission.

Unfortunately we have made only marginal impact in Computer Science. Many of the new aspects of networking, which have now been extended to the five other SEAS departments, existed in Computer Science before SEASnet began. At the time of this writing only nine RT's have been made available to Computer Science from the Grant. The major educational impact for Computer Science will not come until after the RT classroom is constructed. We need additional Project Advance resources to complete this classroom.

The success of SEASnet is a two-edged sword. We have created a very large demand for computing resources. We are limited now in our classrooms (although the AT & T room and the RT classrooms will provide temporary relief). We have too little staff to accomplish our objectives. We need to double the number of workstations each year for the next two years in order to keep pace with the increased need for resources.

Our biggest shortfall, and perhaps the least anticipated need is for user education and documentation. We have constructed a new environment linking PC-DOS and Unix; there is no existing documentation to tell users how to take advantage of the synergism. Many users have not begun to take advantage of the networking resources simply because they do not know how to use them. Unfortunately our environment is developmental and has evolved quite rapidly; the specifics of the user documentation change monthly, which makes it difficult to write documentation.

EDUCATIONAL IMPACT

Survey Results

A student survey was conducted over the three quarters of SEASnet operation. Appendix C shows the questionnaire. Table 5 shows the combined summaries of the basic questions. The surveys were not solicited in any statistically significant way. Some instructors gave students the opportunity to complete the survey in class, which biased the survey to those classes. For this reason the survey is biased toward the CS 12 and CS 13 students. We mailed surveys to other students.

Table 5. Survey Results

SEASnet COURSE EVALUATION

Class Information:

Instructor:

Class (date and time):

TA of your class using SEASnet:

Personal Information:

What year are you completing?

Freshman	*Sophomore*	*Junior*	*Senior*	*Grad*
10	20	18	41	7

What is your major?

Civil	Elec.	Chem.	Comp. Sci.	Mane	MS&E	Other
5	1	5	47	35		3

Why are you taking this course?

Required	*Breadth (Because I want to...)*
90	6

Course Evaluation:

How much did you know about PC/DOS before?

Expert	*Intermediate*	*Beginner*	*None*
8	27	31	32

Have you used an MS/DOS microcomputer before?

Yes, alot	*Yes, a few times*	*Never*
11	50	36

Did you know much about SEASnet before?

Yes, a lot	*Yes, a little*	*No*
5	34	57

How tentative were you in using the PC/AT's?

Very	*A little*	*Not at all*
14	50	32

What was your previous experience with computers?

CS Classes (Please list))		
99		
Engineering Courses	*Science Courses*	*Non-science Courses*
51	8	10
Work Related	*Home Computing*	*Other (Please specify)*
26	50	3 - highschool

Several trends in the second page comments are noted:

1. The most frequent comment related to open access times of the classrooms. Courses such as CS 12 - CS 13 and other courses where programming is taught require large blocks of open time, especially toward the end of the quarter. Students want 24 hour access to the classrooms and do not understand why they cannot have it. Upper division courses, where concepts are taught using computers, usually require less open access time, and tax the classrooms far less. Students taking courses where programming is taught and other courses which require large blocks of computer time, need 24 hour access to workstations. To remedy the situation a large number of the AT&T computers were placed into "clusters" under departmental control. These machines are easier to manage and graduate students can have 24 hour access. Departments are free to create their own access policies. Although the clusters have only been operational for one or two months, their impact is already noticeable. All departments except Electrical Engineering have completed their clusters.

   To provide more hours for undergraduates, and to improve the quality of open access time, TA's are being sought from school resources. Currently we use undergraduate students, especially work study students, to proctor the classrooms. Graduate students functioning as TA's could be asked and trusted to proctor the classrooms for the after midnight hours near exam time.

2. A number of students felt disadvantaged with respect to students with more experience in computing, and students with better access to PC's. Several students commented that grading was not fair because they did not have a PC at home and therefore could not prepare as well as other students.

3. Several students commented that they were being expected to learn quite a bit more material in the same length of time - the normal course material in addition to computing technology. They felt that insufficient time was being allotted to cover computing principles. Many faculty have just the opposite view, feeling that there is insufficient time to cover the course material since the students are preoccupied with understanding the computer applications. Undoubtedly many of the problems are transient problems, and will disappear as we teach more classes using computers. The computer skills learned in beginning courses, perhaps the basic programming courses at the freshman level, will provide the foundation for the computing skills in more advanced courses.

   Undoubtedly the most important ingredient to success of this policy is a common operating system and software tools for beginning, intermediate, and advanced courses. Unfortunately we already have two operating systems. Unix and PC-DOS with which the students must be familiar. Further proliferation of operating systems is deleterious to success, and efforts are needed to make other environments resemble Unix or PC-DOS as much as possible.

   To accomplish these goals we have purchased a set of Unix utilities for PC-DOS. These include the most commonly used Unix utilities, including editors and text manipulation tools. They work in nearly the same way as the server Unix utilities function. They can make the PC-DOS user interface very similar to Unix, and will allow students to more easily move to the Unix environment from PC-DOS.

4. The students concluded almost without exception that the PC-DOS/network environment was preferable to OAC and PICnet. The students preferred SEASnet to OAC because of greater resource availability and easier operating systems. They felt less inhibited since computer resources were not metered in dollar amounts. Those with home PC's also expressed satisfaction with the similarity of home computing and school computing.

   PICnet users preferred SEASnet because of faster response time and the less crowded environment. They preferred PICnet because of its Unix operating system and its 24-hour per day availability.

5. Only one response categorily rejected the notion of using computers in instruction.

6. Almost all students with previous Unix experience indicated a preference for Unix.

7. Another major problem relates to the types of user support provided by SEASnet to users. It was envisioned that many faculty would take a leading roll in teaching the students to use the PC-DOS based computers. This has happened only with a few instructors. Many faculty and department TA's have shown the students how to execute "canned" programs and obtain answers to the example problems. Many students have avoided learning operating system principles and have concentrated on course concepts. This approach minimizes time spent on non-course material, but does not assist the students in learning sufficient computer skills to perform independent work and complete programs of self-study.

   To alleviate this problem one of the SEASnet programming positions will be downgraded in order to hire a programmer or graduate student(s) to assist students in the classroom and provide consulting to faculty. The lower level position will also provide savings for other functions.

   An error in our original planning related to "high-end" user applications and consulting. It was thought that hiring a programmer who understood the high-end applications would be welcomed by faculty. Faculty understand the applications and often resist the programmer's efforts; instead they prefer to obtain assistance on the parts of the problem that directly relate to the computer, e.g. operating system questions, and solve the applications oriented computer problems themselves, or have their own RA's assist them.

Overview

When we were planning SEASnet it was intended that many courses would be taught as "computer aided" classes, where lectures would be interspersed with demonstrations on computers. This concept has been tried by a number of instructors and has succeeded in only a few cases. Where it has succeeded the success is related to the instructor's interest in using computers, OID funds, and amenable subject matter. An instructor's interest in research applications related to the course material is also a large motivating factor.

The most popular and successful courses are those that use computers in recitation periods and as self-study tools. Subject matter and instructor's interest are important but secondary for these types of courses. MANE 171A, *Introduction to Feedback and Control Systems*, is a good example. This course was modified in order to use the PC-DOS classrooms in the first year of SEASnet's operation. Funds were made available to the instructor for summer salary (one month) and a special TA (50% time, one year). Additionally, special software was purchased from a third party using OID funds which is customized for this application. The instructor in charge of this course has explained the impact of computing by saying that he can now expect the students to work greater numbers of realistic problems, where in the past they had only been able to work one or two smaller and less significant problems. This course normally has enrollments greater than 50, which makes lecturing in the computer classrooms impossible; however, the instructor is not convinced that he would want to lecture in the classroom even if this were possible.

CE 184D is another course where significant progress has been made. This is a smaller course and lectures and computing are combined. OID funds were used to develop courseware. TA and summer salary were also provided. Computer programs were written to demonstrate major course topics. Turbo Pascal was used and the instructor wrote most of the programs. Students now work many more example problems than previously and obtain a much better understanding. The easy-to-use graphics afforded by Turbo Pascal were very helpful.

In Computer Science there has been much less success. The AT's are useful for only a few of their courses. Most faculty are unwilling to convert their existing Unix based programs to PC-DOS. The Locus servers have been helpful for many courses, but their poor stability in the first year, and the continuing poor compiler quality have inhibited its use in Computer Science. Fortran users in other departments have had difficulty with the Locus Fortran compiler as well.

The major impact in Computer Science will not be seen until after the RT classroom becomes available. Four of the nine RT's that were allocated to Computer Science have found their way into the VLSI lab. Their impact there is limited so far, due to poor CPU speed and lack of high resolution color displays (1024 x 1024 x 8). The new RT is promising for this and other CS applications. A number of problems still exist, such as support for all the applications and hardware under a single operating system (e.g. 4.2 or AIX). Currently most of the software works under 4.2 while the best monitors only work with AIX.

The School's shops and other services were heavily utilized in building the network and getting it extended to all parts of the School very quickly. Most of the cabling was pulled within a few months and at low cost. This was very effective in getting us started. Unfortunately there were no similar resources for computer maintenance. Resources must be redirected to assist with maintenance. We are hoping to hire an additional person with existing resources. More will be needed. We hope to be able to fix 80% of the problems internally and send the remaining 20% to outside shops. Over a two year period one PC-AT in two has required maintenance. The largest problem has been the CMI hard disk and the second largest problem has been the EGA display. A large fraction of the hard disk failures are unique to the CMI disks, but we still anticipate a significant number of failures in the future.

Software maintenance is a growing problem. Most of the IBM supplied PC software is not current, and will need replacing. It appears that IBM does not intend to keep its third party software current. On the surface this does not appear to be a major drawback, since the most current version of a particular software item is not always required; however, it is a serious problem exists because much of the current software has at least one serious bug which hampers users' efforts. Professional Fortran is a good example. Many of our users have obtained Microsoft Fortran 3.3 in lieu of Professional Fortran, even though Professional Fortran executes faster and produces smaller executable files. Another example of a poor product is IBM's version of Multiplan. It is so inferior to Microsoft's Multiplan, and Lotus 123 that instructors will not invest the time required to develop courseware using Multiplan.

An additional software maintenance problem is keeping DOS file systems current. In spite of our efforts to protect the local hard disks on the AT's by hiding files and educating users, we continue to have a problem with lost files. These files can be restored over the network, but most users do not know how to restore them. Consequently we spend a great deal of time restoring DOS files.

The OID funds have been instrumental in converting new classes. The faculty who have obtained resources have responded very well. The junior faculty have been responsive to these resources, even though they know they must start up their research.

Currently about 30% of the SEAS student population uses the SEASnet facilities in any given quarter. We believe that at the junior and senior levels nearly 100% of the students have been impacted. Our rate of growth indicates that we will need additional classrooms by next year. The IBM RT classroom will be a very important additional resource. Its completion will unload the PC-DOS classrooms of most of the Computer Science courses, since they will greatly prefer the Unix workstations. Most of the Computer Science courses which use the PC-DOS classrooms do so only because Unix workstations are not available.

SUMMARY

Over the next year SEASnet resources must be allocated in different ways and additonal resources must be obtained. We need to achieve the following goals:

1. Personnel must be changed and added to provide more support for low-end applications, including classroom proctors who can consult as well as protect the classroom equipment. Maintenance personnel must also be provided.

2. Additional resources for user consulting, especially to faculty, are required.

3. Additional resources for maintenance are required. IBM maintenance for the 4381 and peripherals is approximately $40,000 per year. PC maintenance was previously provided by IBM. As of July 1, 1987 most of the PC's and RT's will no longer have IBM supplied maintenance. AT & T is providing maintenance until December 31, 1987.

4. We need to improve the IBM 4381 ethernet interface. Although it is very much improved over the original Auscom version, it is still the greatest source of problems related to network performance and stability.

5. We need to acquire a number of new software packages, such as a high quality spreadsheet.

6. We need additional resources from Project Advance, as follows, in order of priority:

   a. Funds to cover the existing RT order (30 machines for the RT classroom, and 13 machines to cover existing allocations to courseware development projects, $1,075,000 list price)[+]

   b. Upgrades to existing software when available.

   c. PC/2's to replace the PC-AT's in existing classrooms, with the existing PC-AT's to become student/faculty workstations in offices and clusters (40 machines, $500,000 list price).

   d. A 4381-13 to add to current server CPU capacity, but using existing tapes and disk drives ($600,000 list price).

7. The newly acquired AT&T classroom, and the soon to be constructed RT classroom will provide growth for the 1987/88 year. Growth in student enrollments in sections using SEASnet classrooms can climb from the current peak of 1,800 to approximately 3,000 without overtaxing the four classrooms.

To accommodate the additional 12,000 students enrolled in SEAS classes, more classrooms and additional server and financial resources will be required. Undoubtedly not all classes and students will require computers for instruction; however, many of the current classroom uses are embryonic and more time will be required as instructors become more proficient in incorporating computing into their instruction.

As an approximation for projecting classroom growth, the following model is offered. Given that there are 15,000 enrollments per year in SEAS classes, or 5,000 per quarter, and assuming that 80% of the students will require computing in some way for three hours per week per student, a total of 12,000 hours per week of student access to computers are required. Assuming that a classroom can be operated for six days per week, for ten hours per day with an average of twenty students using the room, a single classroom can provide 1,200 student hours per week. Under these assumptions ten computer classrooms, or six new computer classrooms need to be constructed.

Additional computing resources which are not included in the above classroom figures, will be required for graduate students working on their theses and dissertation related projects. OAC will continue to meet a portion of these needs, but additional workstations will be required for graduate students.

Space will become a severely limiting factor in creating new computer classrooms. One way of addressing the space requirements are to place as many workstations as possible in graduate student and faculty offices. Clusters in dormitories are another option.

For new classrooms we will need an additional 200 workstations (six rooms at 33 workstations each). For graduate students we will need an additional 200 workstations (one workstation per four graduate students). We will need an additional 125 workstations for faculty (one workstation per faculty). Therefore, 525 new workstations will be required.

We cannot rely entirely on Project Advance to address these needs, since its funds are limited; the RT classroom will exhaust SEAS's share of remaining funds. Therefore we will need the School or Campus to cover shortfalls and growth. We will also need to pursue IBM and other manufacturers for additional grants to obtain new resources for growth.

A second issue which needs to be resolved is where growth should occur. Departments can acquire more of their own microcomputer workstation rooms or we can create additional school-wide classrooms. A balance of both is required; departmental based workstation rooms for graduate students and classrooms for undergraduates.

Staffing changes in SEASnet will alleviate some of the problem in that an additional FTE can be created by downgrading an existing position. Our best estimate of additional staffing needs are two full-time professional FTE beyond the newly created position and two to four 50% time graduate students to proctor classrooms and provide user consulting.

[+] At the time of this writing it appears that 22 of the 43 machines and upgrades for our existing RT's, will become available to SEASnet through a special study.

The impact on instruction has generally taken the form of improved teaching through better and more abundant computer resources. Faculty are now able to enhance their teaching with more realistic, and therefore more complex, instructional problems. The availability of the new computer resources has stimulated a few new courses, such as courses to teach computing skills. There is a controversy between teaching programming versus teaching computing skills, such as using spreadsheets.

We have not seen very many radically different courses stimulated by the new computing resources. This is probably due in large part because only PC-AT's are widely available. Higher power machines will enable Engineering faculty to explore and develop more innovative courses.

This section has concentrated primarily on the current deficiencies in SEASnet and the need for change and growth. This should not be taken as a negative appraisal of SEASnet, but as recognition of our need for growth and evolution to higher powered workstations and servers. The positive appraisal is that SEASnet has created an environment which is stimulating a *renaissance* for instructional computing in SEAS.

REFERENCES

1. SEASnet: A Distributed Academic Computing Environment, A proposal to develop SEASnet, School of Engineering and Applied Science, October 15, 1984.

2. SEASnet - Courseware Development Projects, April 1985.

3. SEASnet - Courseware Development Projects, April 1986.

4. SEASnet - Courseware Development Projects, April 1987.

The 4381 Processor offers significant changes in function and price/performance for the intermediate system user, including:

- Improvements in intermediate system performance
- Improved I/O channel capability
- 68 ns (nanoseconds) processor cycle time.
- Support for both System/370 and extended architecture (XA) operating systems.
- Full dynamic path selection and dynamic reconnections support in XA mode for Model Group 3 on 3-megabyte channels and devices with the appropriate function.

Modes of Operation

The two available modes are System/370 and extended architecture. You select the mode of operation at initial microcode load (IML) time.

Extended architecture mode is used with an extended architecture (XA) operating system. Programs written for a 4381 Processor in extended architecture (XA) mode are supported by:

- MVS/XA
- VM/XA Migration Aid.

System/370 mode runs programs used on System/370 and 4300 Processors that do not violate the exceptions noted under "Compatibility." System/370 mode support for the 4381 Model Groups 1, 2, and 3 is shown in Figure 2.

Model Groups 1 and 2	**Model Group 3**
DOS/VSE	VM/SP
VS1/BPE	MVS/SP
MVS/SP	
VM/SP	
VM/SP (with or without High Performance Option)	

Figure 2. System/370 Support for 4381 Processors

For more information on programming support, including descriptions and release levels, see your IBM representative.

maximum	28.0	ms
Seek Time: Models AE4 and BE4		
minimum (single cylinder)	3.0	ms
average	17.0	ms
maximum	31.0	ms
Full track rotation time (all models)	16.56	ms
Average rotational delay (all models)	8.3	ms
Data transfer rate (all models)	3.0	Mb/sec

Figure 5 (Part 2 of 2). Performance Summary for IBM 3380 Access Mechanisms

Some of the terms used in Figure 5 are:

Seek time, or access motion time: The time required to move the access arm from one cylinder to another. More precisely defined, the seek time is the time interval beginning when the channel issues a Seek command (requiring access motion) and ending when the 3380 responds with a Seek Complete indication to the IBM 3880. If the access arm is already at the correct cylinder, there is no access motion. Seek time is negligible, because little time is needed to electronically switch from one read/write head to another.

Average seek time: The average time taken when moving the access arm across 1/3 of the cylinders.

Average rotational delay: The average time required for the disk to rotate, to position the desired data record under the read/write head so data transfer can begin. This is sometimes called average rotational latency. Average rotational delay is 1/2 full track rotation time.

Data transfer rate: The rate at which data is transferred between the 3380 and the storage control, the 3880.

Performance Comparison: IBM 3350 vs. 3380 Models

Figure 6 compares performance between 3350 and 3380 models.

Performance Characteristics	3350	3380 Models A04, AA4, B04	3380 Models AD4 and BD4	3380 Models AE4 and BE4
Average seek time (ms)	25.0	16.0	15.0	17.0
Full track rotation time (ms)	16.7	16.56	16.56	16.56
Data transfer rate (Mb/sec)	1.2	3.0	3.0	3.0

Figure 6. Performance Comparison: IBM 3350 vs. 3380

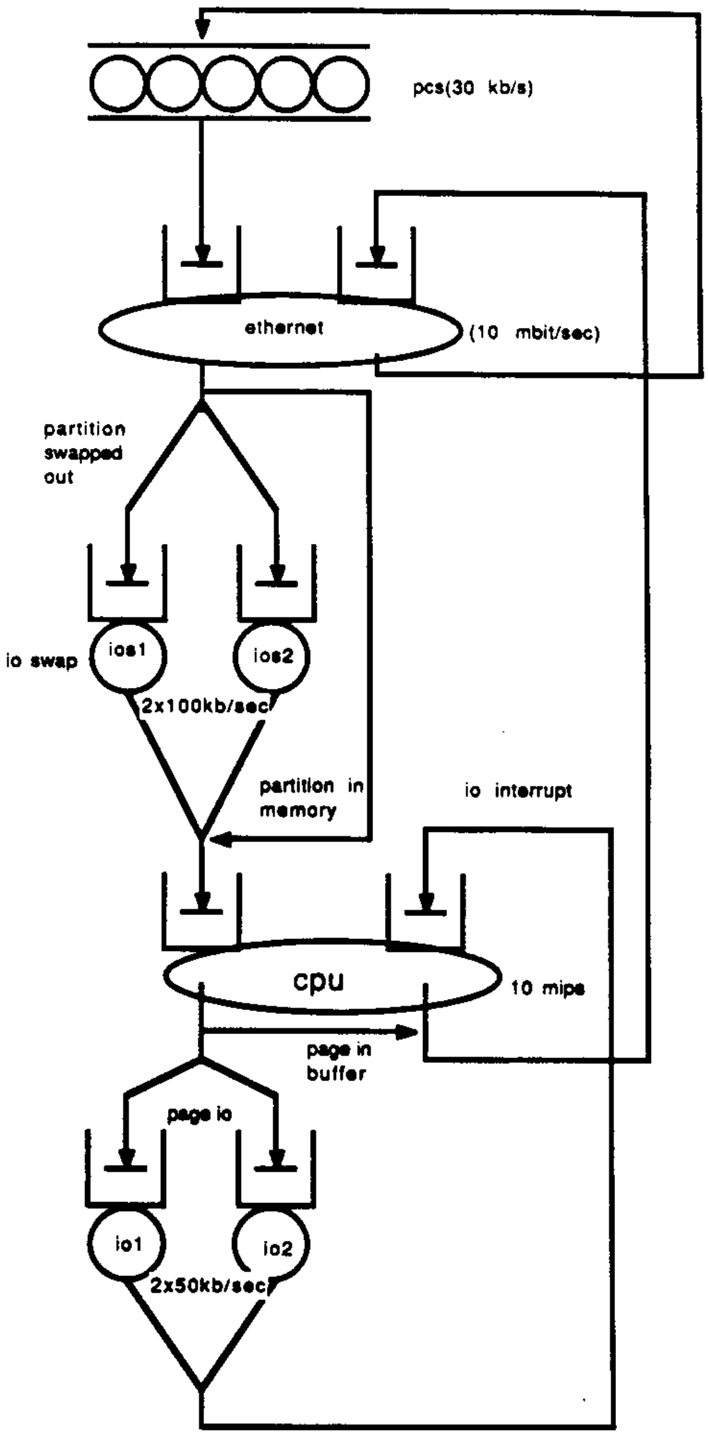
Data Capacity Summary: IBM 3350 and 3380 Models

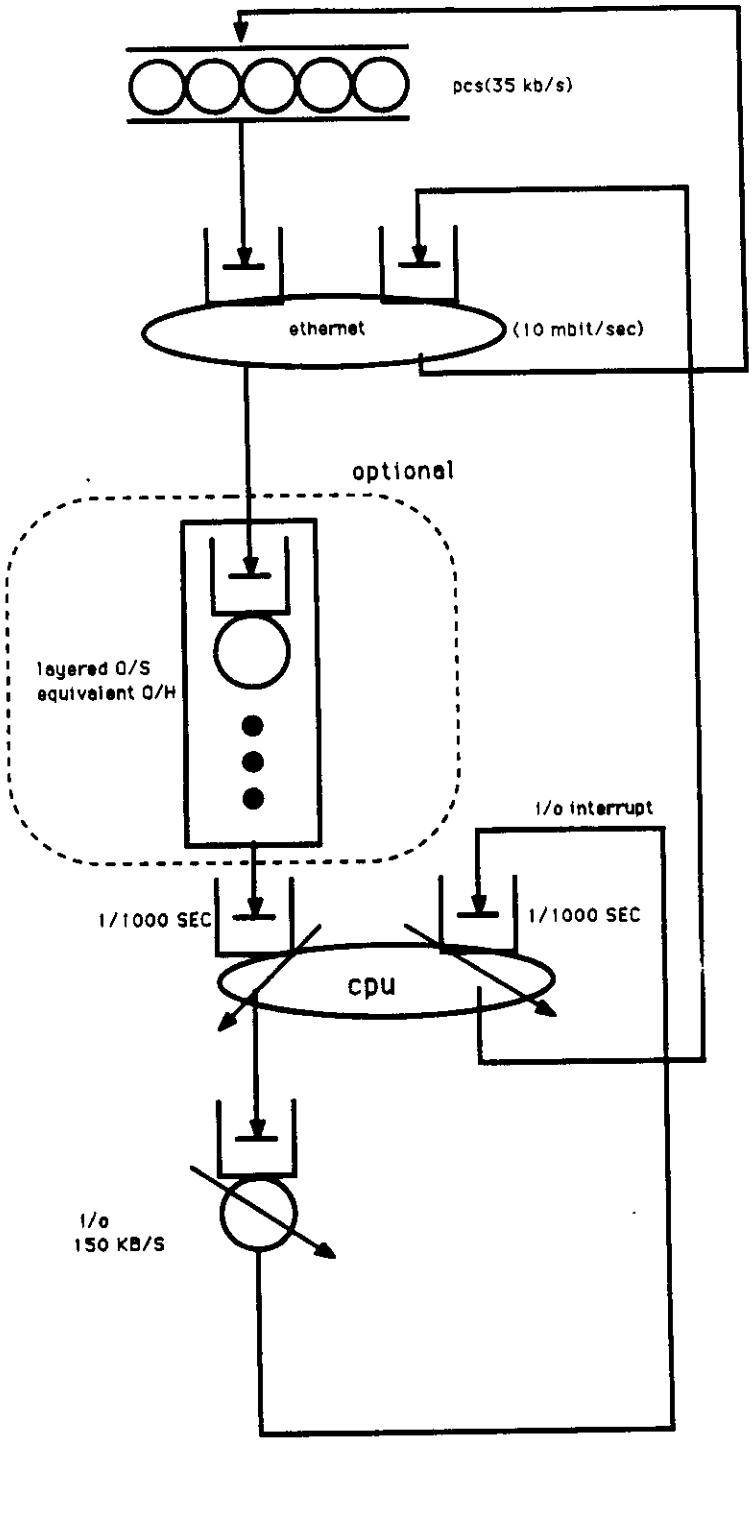
An IBM 3380 Model A04, AA4, B04, AD4, or BD4 unit can store as much data as four IBM 3350 units. An IBM 3380 Model AE4 or BE4 unit can store as much as data as eight 3350 units. (See Figure 7.)

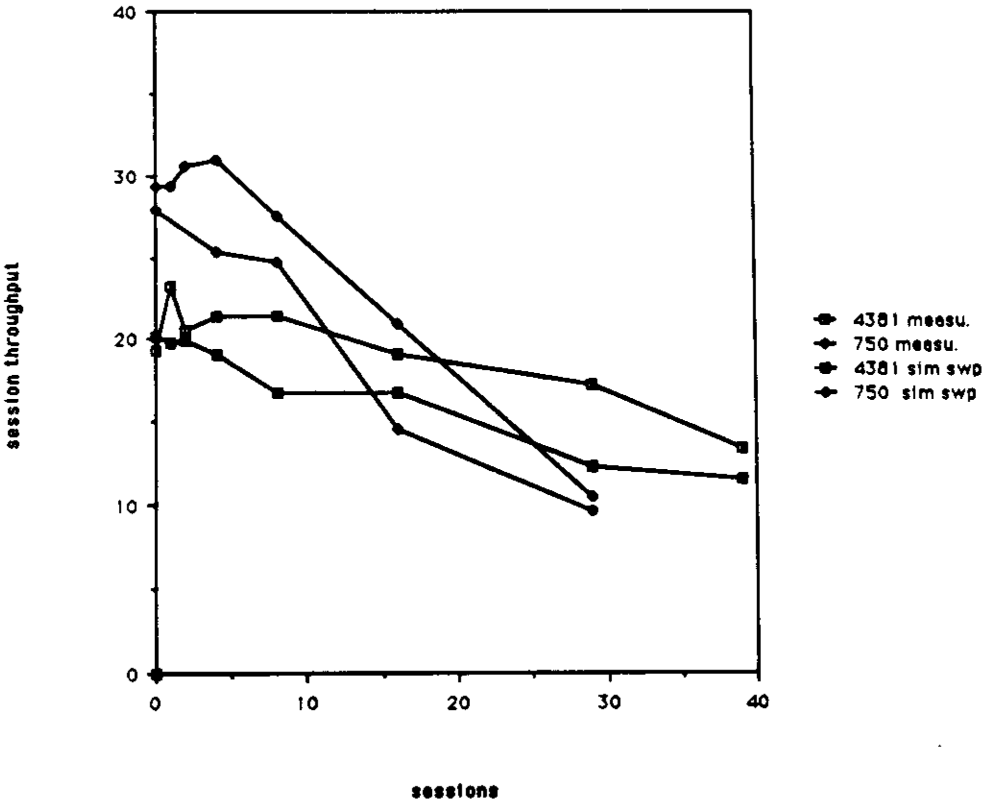
APPENDIX B

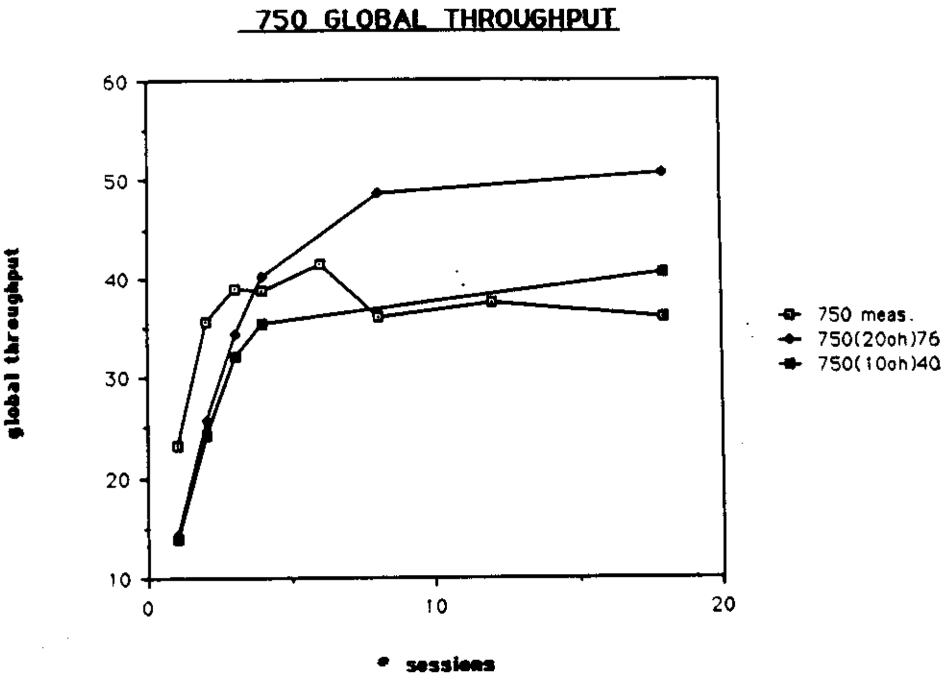
Additional Models, Measurements, and Simulations

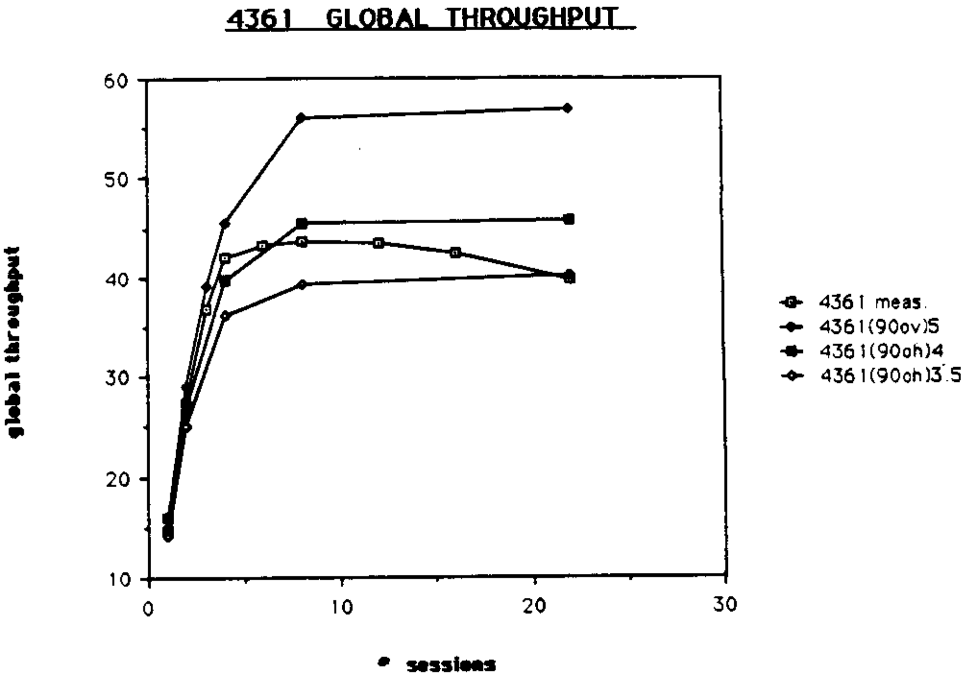
The material presented in the body of the dissertation is only a part of the incremental effort we undertook. In this appendix we present other, earlier models that were considered, as well as simulation results, and relevant measurement results. This additional data gives the flavor of some of the progress of our work.


pcs(30 kb/s)
ethernet
(10 mbit/sec)
partition swapped out
io swap
ios1
ios2
2x100kb/sec
partition in memory
io interrupt
cpu
10 mips
page in buffer
page io
io1
io2
2x50kb/sec

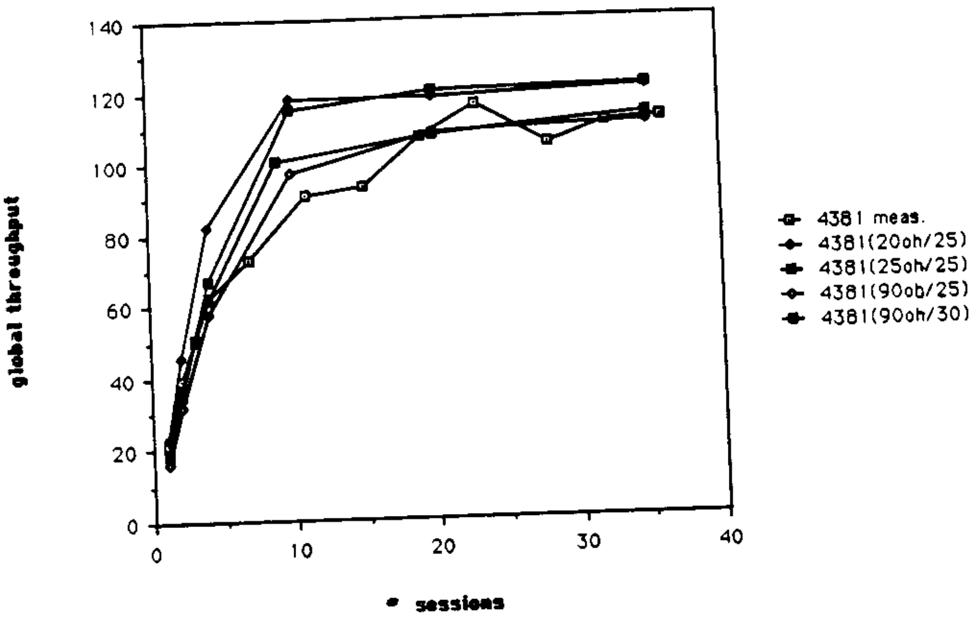

pcs(35 kb/s)
ethernet
(10 mbit/sec)
optional
layered O/S
equivalent O/H
i/o interrupt
1/1000 SEC
1/1000 SEC
cpu
i/o
150 KB/S

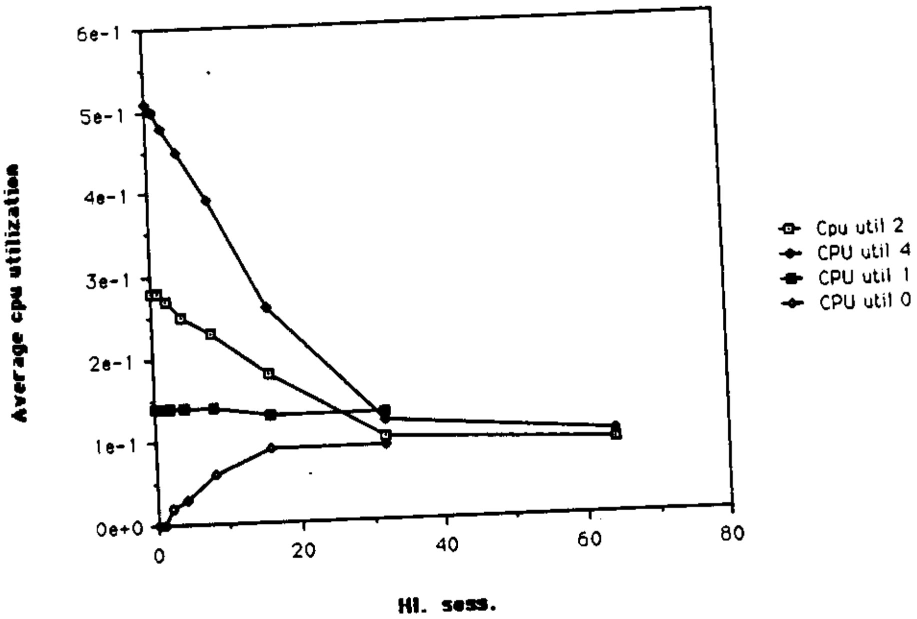

40
30
20
10
0
0
10
20
30
40
session throughput
sessions
4381 measu.
750 measu.
4381 sim swp
750 sim swp

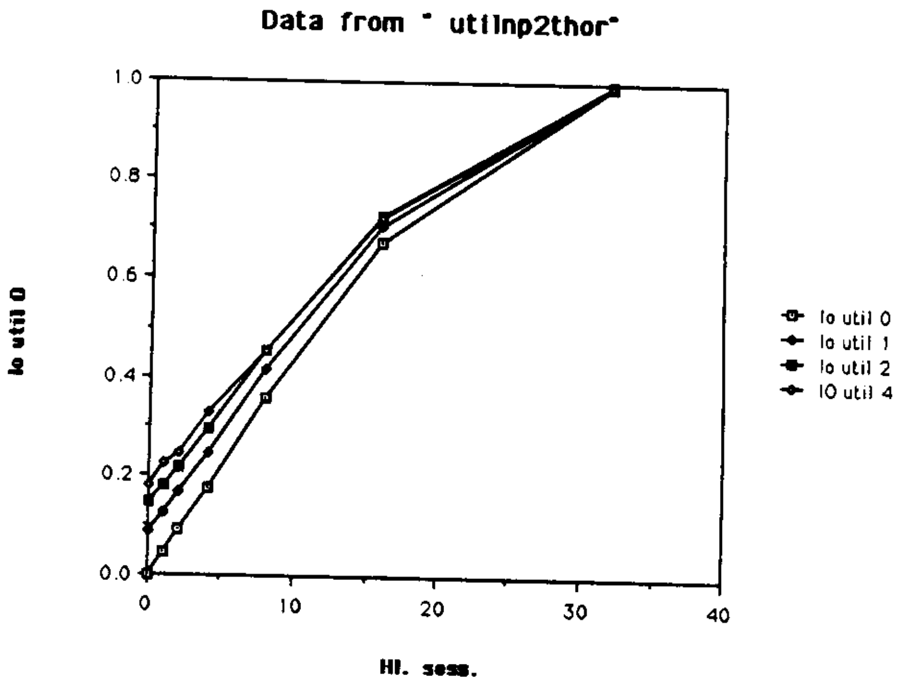

750 GLOBAL THROUGHPUT
60
50
40
30
20
10
0
10
20
global throughput
# sessions
750 meas.
750(20oh)76
750(10oh)40

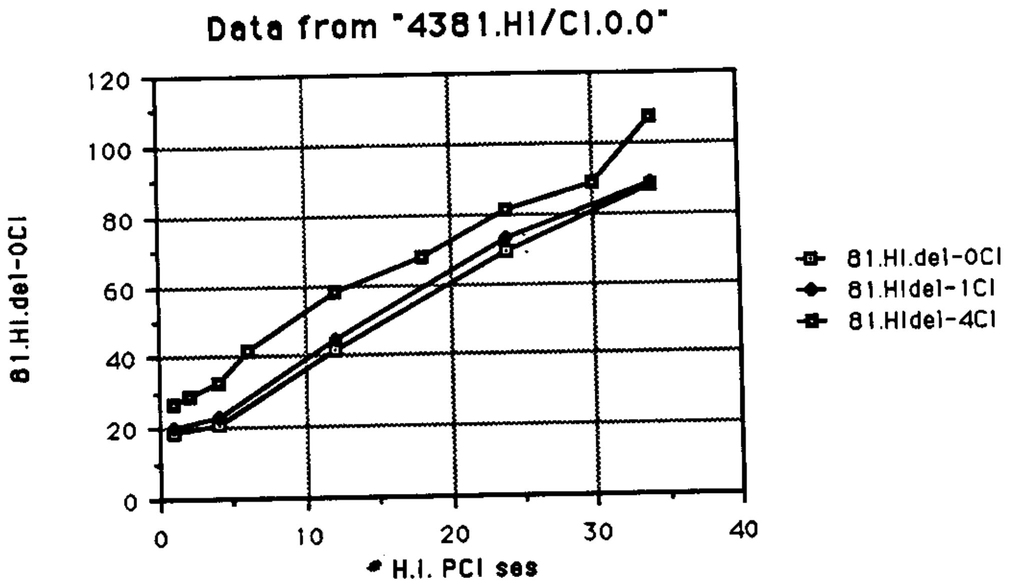

4361 GLOBAL THROUGHPUT
global throughput
60
50
40
30
20
10
0
10
20
30
# sessions
4361 meas.
4361(90ov)5
4361(90oh)4
4361(90oh)3.5

4381 GLOBAL THROUGHPUT




6e-1
5e-1
4e-1
3e-1
2e-1
1e-1
0e+0
0
20
40
60
80
Average cpu utilization
Hl. sess.
Cpu util 2
CPU util 4
CPU util 1
CPU util 0


Data from " utilnp2thor"
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
lo util 0
HI. sess.
lo util 0
lo util 1
lo util 2
IO util 4


Data from "4381.HI/CI.0.0"
120
100
80
60
40
20
0
0
10
20
30
40
81.HI.del-0CI
# H.I. PCI ses
81.HI.del-0CI
81.HIdel-1CI
81.HIdel-4CI

APPENDIX C

Additional Resq Modeling Reports

We present here additional RESQ2 reports that we have constructed during our modeling effort. Some of the reports show results of simulation runs, while others show earlier model representations that evolved with the modeling process.

```
RESQ2 VERSION DATE: SEPTEMBER 23, 1986-  TIME: 21:25:50  DATE: 07/05/88
MODEL:THORS
G:0
REPLICATION   1: TERM1 DEPARTURE LIMIT
REPLICATION   2: TERM1 DEPARTURE LIMIT
REPLICATION   3: TERM1 DEPARTURE LIMIT
REPLICATION   4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  22901 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:        42.52490
                        CPU TIME:          181.58
NUMBER OF EVENTS PER REPLICATION:           17999
        NUMBER OF REPLICATIONS:                4

WHAT:ALL


ELEMENT         UTILIZATION
TERM2           0.00000
TERM1           0.00000
 TERMF           0.00000
 TERMFI          0.00000
ETHERN          0.04627
 INETHER         2.6904E-03
 OUTETHER        0.04357
CPU             0.99516
 FCPU            0.49329
 SCPU            0.50187
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.36278
 IOC             0.36278


ELEMENT         THROUGHPUT
TERM1           42.33177
 TERMF           42.33177
ETHERN          84.64000
 INETHER         42.33177
 OUTETHER        42.30821
CPU             84.65759
 FCPU            42.34938
 SCPU            42.30821
IO2             42.33177
IO3             42.33177
IOS             42.33177
IO1             42.33177
IO              42.33762
 IOC             42.33762


ELEMENT         MEAN QUEUE LENGTH
TERM1           1.80543
 TERMF           1.80543
ETHERN          0.05024
 INETHER         4.6166E-03
 OUTETHER        0.04562
CPU             5.58474
 FCPU            2.77684
 SCPU            2.80790
IO2             0.00000
IO3             0.00000
```

```
IOS             0.00000
IO1             0.00000
IO              0.55960
 IOC             0.55960


ELEMENT         STANDARD DEVIATION OF QUEUE LENGTH
TERM1           1.33416
 TERMF           1.33416
ETHERN          0.23682
 INETHER         0.06840
 OUTETHER        0.21837
CPU             1.61669
    RESQ ATTENTION CONDITION
REPLICATION   1: TERM1 DEPARTURE LIMIT
REPLICATION   2: TERM1 DEPARTURE LIMIT
REPLICATION   3: TERM1 DEPARTURE LIMIT
REPLICATION   4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  35933 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:        42.52490
                        CPU TIME:          182.01
NUMBER OF EVENTS PER REPLICATION:           17999
        NUMBER OF REPLICATIONS:                4

WHAT:QUIT
G:1
REPLICATION   1: TERM1 DEPARTURE LIMIT
REPLICATION   2: TERM1 DEPARTURE LIMIT
REPLICATION   3: TERM1 DEPARTURE LIMIT
REPLICATION   4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  23793 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:        42.33745
                        CPU TIME:          181.69
NUMBER OF EVENTS PER REPLICATION:           17693
        NUMBER OF REPLICATIONS:                4

WHAT:ALL


ELEMENT         UTILIZATION
TERM2           0.00000
TERM1           0.00000
 TERMF           0.00000
 TERMFI          0.00000
ETHERN          0.04696
 INETHER         2.5309E-03
 INETHERI        1.3424E-04
 OUTETHER        0.04190
 OUTETHERI       2.3061E-03
CPU             0.99214
 FCPU            0.46977
 FCPUI           0.02926
 SCPU            0.46552
 SCPUI           0.02759
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.41685
 IOC             0.35201
 IOCI            0.06484
```

```
ELEMENT         THROUGHPUT
TERM2           2.33205
TERM1           42.52916
 TERMF           40.18527
 TERMFI          2.34389
ETHERN          84.99335
 INETHER         40.18527
 INETHERI        2.34389
 OUTETHER        40.13805
 OUTETHERI       2.32614
CPU             84.96432
 FCPU            40.15625
 FCPUI           2.34389
 SCPU            40.13805
 SCPUI           2.32614
IO2             40.18527
IO3             40.18527
IOS             40.18527
IO1             40.18527
IO              42.48790
 IOC             40.14401
 IOCI            2.34389


ELEMENT         MEAN QUEUE LENGTH
TERM2           0.48405
TERM1           1.86711
 TERMF           1.76993
 TERMFI          0.09718
ETHERN          0.05117
 INETHER         4.5807E-03
 INETHERI        2.6224E-04
 OUTETHER        0.04392
 OUTETHERI       2.4094E-03
CPU             5.83146
 FCPU            2.73764
 FCPUI           0.17961
 SCPU            2.75488
 SCPUI           0.15934
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.76621
 IOC             0.68906
 IOCI            0.07715


ELEMENT         STANDARD DEVIATION OF QUEUE LENGTH
TERM2           0.49951
TERM1           1.34779
 TERMF           1.31234
 TERMFI          0.29599
ETHERN          0.23992
 INETHER         0.06873
 INETHERI        0.01602
 OUTETHER        0.21404
 OUTETHERI       0.04900
CPU             1.82287
 FCPU            1.43765
 FCPUI           0.38380
 SCPU            1.46492
 SCPUI           0.36595
```

```
IO2             0.00000
IO3             0.00000
    RESQ ATTENTION CONDITION
REPLICATION    1: TERM1 DEPARTURE LIMIT
REPLICATION    2: TERM1 DEPARTURE LIMIT
REPLICATION    3: TERM1 DEPARTURE LIMIT
REPLICATION    4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  37053 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:        42.33745
                        CPU TIME:          182.23
NUMBER OF EVENTS PER REPLICATION:           17693
          NUMBER OF REPLICATIONS:               4

WHAT:QUIT
G:2
REPLICATION    1: TERM1 DEPARTURE LIMIT
REPLICATION    2: TERM1 DEPARTURE LIMIT
REPLICATION    3: TERM1 DEPARTURE LIMIT
REPLICATION    4: TERM1 DEPARTURE LIMIT
NO ERRORS DETECTED DURING SIMULATION.  23603 DISCARDED EVENTS

  SIMULATED TIME PER REPLICATION:        42.12358
                        CPU TIME:          180.60
NUMBER OF EVENTS PER REPLICATION:           17434
          NUMBER OF REPLICATIONS:               4

WHAT:ALL


ELEMENT         UTILIZATION
TERM2           0.00000
TERM1           0.00000
 TERMF           0.00000
 TERMFI          0.00000
ETHERN          0.04676
 INETHER         2.5006E-03
 INETHERI        3.0324E-04
 OUTETHER        0.03938
 OUTETHERI       4.5765E-03
CPU             0.99126
 FCPU            0.44268
 FCPUI           0.05281
 SCPU            0.43907
 SCPUI           0.05669
IO2             0.00000
IO3             0.00000
IOS             0.00000
IO1             0.00000
IO              0.44069
 IOC             0.32668
 IOCI            0.11401


ELEMENT         THROUGHPUT
TERM2           4.51678
TERM1           42.73463
 TERMF           38.21785
 TERMFI          4.51678
ETHERN          85.49290
 INETHER         38.21785
 INETHERI        4.51678
 OUTETHER        38.24153
 OUTETHERI       4.51673
 IO            21.84369
  IOCI          21.84369

 ELEMENT       STANDARD DEVIATION OF QUEUE LENGTH
 TERM2         2.75549
 TERM1         1.31162
  TERMFI        1.31162
 ETHERN        0.23060
  INETHERI      0.06789
  OUTETHERI     0.21134
 CPU           0.94862
  FCPUI         0.60498
  SCPUI         0.60621
 IO            3.18894
  IOCI          3.18894

 ELEMENT       MEAN QUEUEING TIME
 TERM2         0.19914
 TERM1         0.04396
  TERMFI        0.04396
 ETHERN        5.9279E-04
  INETHERI      1.1237E-04
  OUTETHERI     1.0736E-03
 CPU           7.1249E-03
  FCPUI         7.1139E-03
  SCPUI         7.1357E-03
 IO            0.55599
  IOCI          0.55599

 ELEMENT       STANDARD DEVIATION OF QUEUEING TIME
 TERM2         0.19960
 TERM1         0.04369
  TERMFI        0.04369
 ETHERN        9.4420E-04
  INETHERI      3.4448E-04
  OUTETHERI     1.0960E-03
 CPU           7.1667E-03
  FCPUI         7.1980E-03
  SCPUI         7.1315E-03
 IO            0.14431
  IOCI          0.14431

 ELEMENT       MEAN TOKENS IN USE

 ELEMENT       MEAN TOTAL TOKENS IN POOL

 ELEMENT       QUEUE LENGTH DISTRIBUTION

 ELEMENT       QUEUEING TIME DISTRIBUTION

 ELEMENT       DISTRIBUTION OF TOKENS IN USE

 ELEMENT       DISTRIBUTION OF TOTAL TOKENS IN POOL
```

```
ELEMENT       MAXIMUM QUEUE LENGTH
TERM2         20
TERM1         9
 TERMF         0
 TERMFI        9
ETHERN        5
 INETHER       0
 INETHERI      3
 OUTETHER      0
 OUTETHERI     4
CPU           9
 FCPU          0
 FCPUI         7
 SCPU          0
 SCPUI         7
IO2           0
IO3           0
IOS           0
IO1           0
IO            31
 IOC           0
 IOCI          31

ELEMENT       MAXIMUM QUEUEING TIME
TERM2         1.74389
TERM1         0.45413
 TERMF         0.00000
 TERMFI        0.45413
ETHERN        0.01313
 INETHER       0.00000
 INETHERI      8.7369E-03
 OUTETHER      0.00000
 OUTETHERI     0.01313
CPU           0.06267
 FCPU          0.00000
 FCPUI         0.06022
 SCPU          0.00000
 SCPUI         0.06267
IO2           0.00000
IO3           0.00000
IOS           0.00000
IO1           0.00000
IO            1.11514
 IOC           0.00000
 IOCI          1.11514

WHAT:QUIT
G:
```

```
FILE: NETVAX   RQ2LIST  A    MANUFACTURING ENGINEERING PROGRAM


RESQ2 Translator V02.1986.10.22  Time: 20:21:26  Date: 03/21/88


 *     1* 0* MODEL:ret
 *     2* 0*    METHOD:simulation
 *     3* 0*        numeric parameters : q
 *     4* 0*        queue : term2
 *     5* 0*        type : is
 *     6* 0*        class list: termi
 *     7* 0*        service times: 60
 *     8* 0*    QUEUE:term1
 *     9* 0*    type: is
 *    10* 0*    class list: termf termfi
 *    11* 0*    service times: 576/25000 576/25000
 *    12* 0*    QUEUE:ethern
 *    13* 0*       TYPE:active
 *    14* 0*       SERVERS:1
 *    15* 0*       DSPL:fcfs
 *    16* 0*       CLASS LIST:inether inetheri outether outetheri
 *    17* 0*          WORK DEMANDS:64 64 576 576
 *    18* 0*       SERVER -
 *    19* 0*           rates: 750000 750000 750000 750000
 *    20* 0*    QUEUE:cpu
 *    21* 0*       TYPE:active
 *    22* 0*       SERVERS:1
 *    23* 0*       DSPL:fcfs
 *    24* 0*       CLASS LIST:fcpu fcpui scpu scpui
 *    25* 0*          WORK DEMANDS:10000 10000 10000 10000
 *    26* 0*       SERVER -
 *    27* 0*          RATES:4000000 4000000 4000000 4000000
 *    28* 0*          QUEUE: io2
 *    29* 0*          type: active
 *    30* 0*          servers: 1
 *    31* 0*          dspl: fcfs
 *    32* 0*          class list: ioc2
 *    33* 0*          work demands: 0000
 *    34* 0*          server -
 *    35* 0*            rates: 1200000
 *    36* 0*            QUEUE: io1
 *    37* 0*            type: active
 *    38* 0*            servers: 1
 *    39* 0*            dspl: fcfs
 *    40* 0*            class list: ioc1
 *    41* 0*            work demands: 0000
 *    42* 0*             server -
 *    43* 0*              rates: 1200000
 *    44* 0*          QUEUE: io3
 *    45* 0*            type: active
 *    46* 0*            servers: 1
 *    47* 0*            dspl: fcfs
 *    48* 0*            class list: iosc
 *    49* 0*            work demands: 0000
 *    50* 0*            server -
```

```
FILE: NETVAX   RQ2LIST  A    MANUFACTURING ENGINEERING PROGRAM

 *    51* 0*                rates:  1200000
 *    52* 0*          QUEUE: io1
 *    53* 0*           TYPE: active
 *    54* 0*           servers: 1
 *    55* 0*           dspl : fcfs
 *    56* 0*           CLASS LIST: ioc1
 *    57* 0*            WORK DEMANDS: 0000
 *    58* 0*            SERVER -
 *    59* 0*                RATES: 1200000
 *    60* 0*     QUEUE:io
 *    61* 0*         TYPE:active
 *    62* 0*         SERVERS:1
 *    63* 0*         DSPL:fcfs
 *    64* 0*         CLASS LIST:ioc ioci
 *    65* 0*            WORK DEMANDS:0000 0000
 *    66* 0*         SERVER -
 *    67* 0*            rates:   80000 80000
 *    68* 0*         CHAIN: bat
 *    69* 0*         type: closed
 *    70* 0* population : q
 *    71* 0*   :termi -> termfi ->   inetheri  ++
              -> fcpui -> ioci -> scpui -> outetheri ->    termi
 *    72* 0*    CHAIN:int
 *    73* 0*       TYPE:closed
 *    74* 0* population : 1
 *    75* 0*   :termf ->   inether -> iosc -> ioc1 -> ioc2 -> ioc3++
              -> fcpu -> ioc -> scpu -> outether ->   termf
 *    76* 0*
 *    77* 0*  confidence interval method: replications
 *    78* 0* initial state definition -
 *    79* 0* chain: bat
 *    80* 0*
 *    81* 0* node list: termi
 *    82* 0* init pop: q
 *    83* 0* chain: int
 *    84* 0* node list: termf
 *    85* 0* init pop: 1
 *    86* 0* confidence level : 95
 *    87* 0* initial portion discarded: 10
 *    88* 0* replic limits -
 *    89* 0*   simulated time: 3600
 *    90* 0*   events: 50000
 *    91* 0*  queues for departure counts: term1
 *    92* 0*  departures: 400
 *    93* 0*  limit - cp seconds : 100
 *    94* 0*  trace: no
 *    95* 0* END
 *    96* 0* END

 NO FATAL ERRORS DETECTED DURING COMPILATION.


SETUP CROSS REFERENCE AND IDENTIFIER INFORMATION

                    ELEMENT                                     CONTAINING
```

```
FILE: ODINSIN  RQ2IPP   A   MANUFACTURING ENGINEERING PROGRAM

MODEL:NET
   METHOD:simulation
        numeric parameters : g
   QUEUE:term1
   type: is
   class list: termf
  service times: 64/25000
   QUEUE:ethers
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:inether outether
         WORK DEMANDS:64 576
      SERVER -
          rates: 750000 750000
   QUEUE:cpu
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:fcpu scpu
         WORK DEMANDS:10000 10000
      SERVER -
         RATES:10000000 10000000
         QUEUE: io2
         type: active
         servers: 1
         dspl: fcfs
         class list: ioc2
         work demands: 4000
         server -
           rates: 1200000
           QUEUE: io3
           type: active
           servers: 1
           dspl: fcfs
           class list: ioc3
           work demands: 4000
           server -
             rates: 1200000
         QUEUE: ios
           type: active
           servers: 1
           dspl: fcfs
           class list: iosc
           work demands: 4000
           SERVER -
            rates:  1200000
      QUEUE: io1
       TYPE: active
       servers: 1
       dspl : fcfs
       CLASS LIST: ioc1
        WORK DEMANDS: 4000
        SERVER -
            RATES: 1200000
```

```
FILE: ODINSIN  RQ2IPP   A   MANUFACTURING ENGINEERING PROGRAM

   QUEUE:io
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:ioc
         WORK DEMANDS:4000
      SERVER -
         rates:   1200000
   CHAIN:int
      TYPE:closed
population : g
  :termf ->   inether -> iosc -> ioc1 -> ioc2 -> ioc3++
  -> fcpu -> ioc -> scpu -> outether ->   termf
 confidence interval method: replications
initial state definition -
chain: int
node list: termf
init pop: g
confidence level : 95
initial portion discarded: 10
replic limits -
  simulated time: 1600
  events: 50000
 queues for departure counts: term1
 departures: 400
 limit - cp seconds : 100
 trace: no
END
END
```

```
FILE: NET      RQ2INP   A   MANUFACTURING ENGINEERING PROGRAM

MODEL:NET
   METHOD:simulation
       numeric parameters : g
     queue:rt
       type:passive
       tokens:100
       dspl:fcfs
       allocate node list:grt
       number of tokens to allocate: 1
       release node list: frt
  QUEUE:term1
  type: is
  class list: termf
 service times: 64/25000
  QUEUE:ethern
     TYPE:active
     SERVERS:1
     DSPL:fcfs
     CLASS LIST:inether outether
        WORK DEMANDS:64 576
     SERVER -
        rates: 750000 750000
  QUEUE:cpu
     TYPE:active
     SERVERS:1
     DSPL:fcfs
     CLASS LIST:fcpu scpu
        WORK DEMANDS:10000 10000
     SERVER -
        RATES:4000000 4000000
        QUEUE: ios
          type: active
          servers: 1
          dspl: fcfs
          class list: iosc
          work demands: 4000
          SERVER -
           rates:  800000
  QUEUE:io
     TYPE:active
     SERVERS:1
     DSPL:fcfs
     CLASS LIST:ioc
        WORK DEMANDS:4000
     SERVER -
        rates:   800000
  CHAIN:int
     TYPE:closed
population : g
:termf -> grt -> inether -> fcpu iosc ; 0/g 1-0/g
:iosc -> fcpu
:fcpu ->++
ioc free; 8/8 0/8
:ioc -> scpu -> free
:free -> outether -> frt -> termf
```

```
FILE: NET      RQ2INP   A   MANUFACTURING ENGINEERING PROGRAM

 confidence interval method: replications
initial state definition -
chain: int
node list: termf
init pop: g
confidence level : 95
initial portion discarded: 10
replic limits -
  simulated time: 3600
  events: 50000
 queues for departure counts: term1
 departures: 400
 limit - cp seconds : 100
 trace: no
END
END
```

```
FILE: NETN     RQ2INP   A   MANUFACTURING ENGINEERING PROGRAM


MODEL:NET
   METHOD:numerical
       numeric parameters : g
       numeric parameters : gg
     queue:rt
       type:passive
       tokens:100
       dspl:fcfs
       allocate node list:grt
       number of tokens to allocate: 1
       release node list: frt
   QUEUE:term1
   type: is
   class list: termf
  service times: 64/25000
   QUEUE:ether
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:inether outether
         WORK DEMANDS:64 576
      SERVER -
          rates: 750000 750000
   QUEUE:cpu
      TYPE:active
      SERVERS:1
      DSPL:fcfs
      CLASS LIST:fcpu scpu
         WORK DEMANDS:10000 10000
      SERVER -
         RATES:10000000 10000000
         QUEUE: ios
           type: active
           servers: 2
           dspl: fcfs
           class list: iosc
           work demands: 110000
           SERVER -
            rates:  500000
   QUEUE:io
      TYPE:active
      SERVERS:2
      DSPL:fcfs
      CLASS LIST:ioc
         WORK DEMANDS:4000
      SERVER -
         rates:   100000
   CHAIN:int
      TYPE:closed
population : g
:termf -> grt -> inether -> fcpu iosc ; gg/g 1-gg/g
:iosc -> fcpu
:fcpu ->••
ioc free; 8/8 0/8
:ioc -> scpu -> free
```

```
FILE: NETN     RQ2INP   A   MANUFACTURING ENGINEERING PROGRAM


:free -> outether -> frt -> termf
 confidence interval method: replications
initial state definition -
chain: int
node list: termf
init pop: g
confidence level : 95
initial portion discarded: 10
replic limits -
  simulated time: 3600
  events: 50000
 queues for departure counts: term1
 departures: 400
 limit - cp seconds : 100
 trace: no
END
END
```

```
FILE: NETSAV   EQ2INP   A   MANUFACTURING ENGINEERING PROGRAM

MODEL:NET
   METHOD:simulation
        numeric parameters : g
        numeric parameters : m
      queue:rt
        type:passive
        tokens:100
        dspl:fcfs
        allocate node list:grt
        number of tokens to allocate: 1
        release node list: frt
    QUEUE:term1
       TYPE:is
       CLASS LIST:termf
          service times: 1
    QUEUE:others
       TYPE:active
       SERVERS:1
       DSPL:fcfs
       CLASS LIST:inother outother
          WORK DEMANDS:512 512
       SERVER -
           RATES: 1250000 1250000
    QUEUE:memory
       TYPE:passive
       tokens:25
       DSPL:fcfs
       ALLOCATE NODE LIST:get
          NUMBERS OF TOKENS TO ALLOCATE:1
       RELEASE NODE LIST:free
    QUEUE:ios
       TYPE:active
       SERVERS:2
       DSPL:fcfs
       CLASS LIST:iomc
          work demands: m
       SERVER -
          RATES:30000
    QUEUE:cpu
       TYPE:active
       SERVERS:1
       DSPL:fcfs
       CLASS LIST:fcpu mcpu
          WORK DEMANDS:10000 10000
       SERVER -
          RATES:10000000 10000000
    QUEUE:io
       TYPE:active
       SERVERS:2
       DSPL:fcfs
       CLASS LIST:ioc
          WORK DEMANDS:4000
       SERVER -
          RATES:30000
    CHAIN:int
```

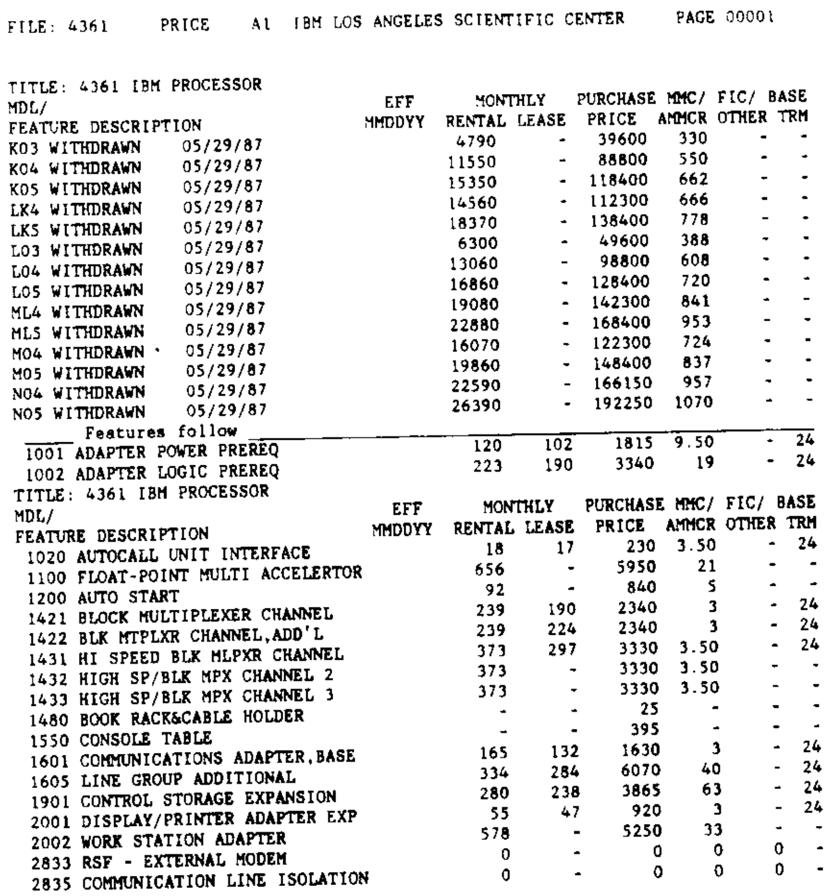
```
FILE: NETSAV   EQ2INP   A   MANUFACTURING ENGINEERING PROGRAM

      TYPE:closed
population : g
:termf -> grt -> inother -> get -> iomc -> fcpu -> ++
ioc -> mcpu -> free -> outother -> frt -> termf

 confidence interval method: replications
initial state definition -
chain: int
node list: termf
init pop: g
confidence level : 95
initial portion discarded: 10
replic limits -
  simulated time: 3600
  events: 50000
 queues for departure counts: term1
 departures: 400
 limit - cp seconds : 100
 trace: no
END
END
```

APPENDIX D

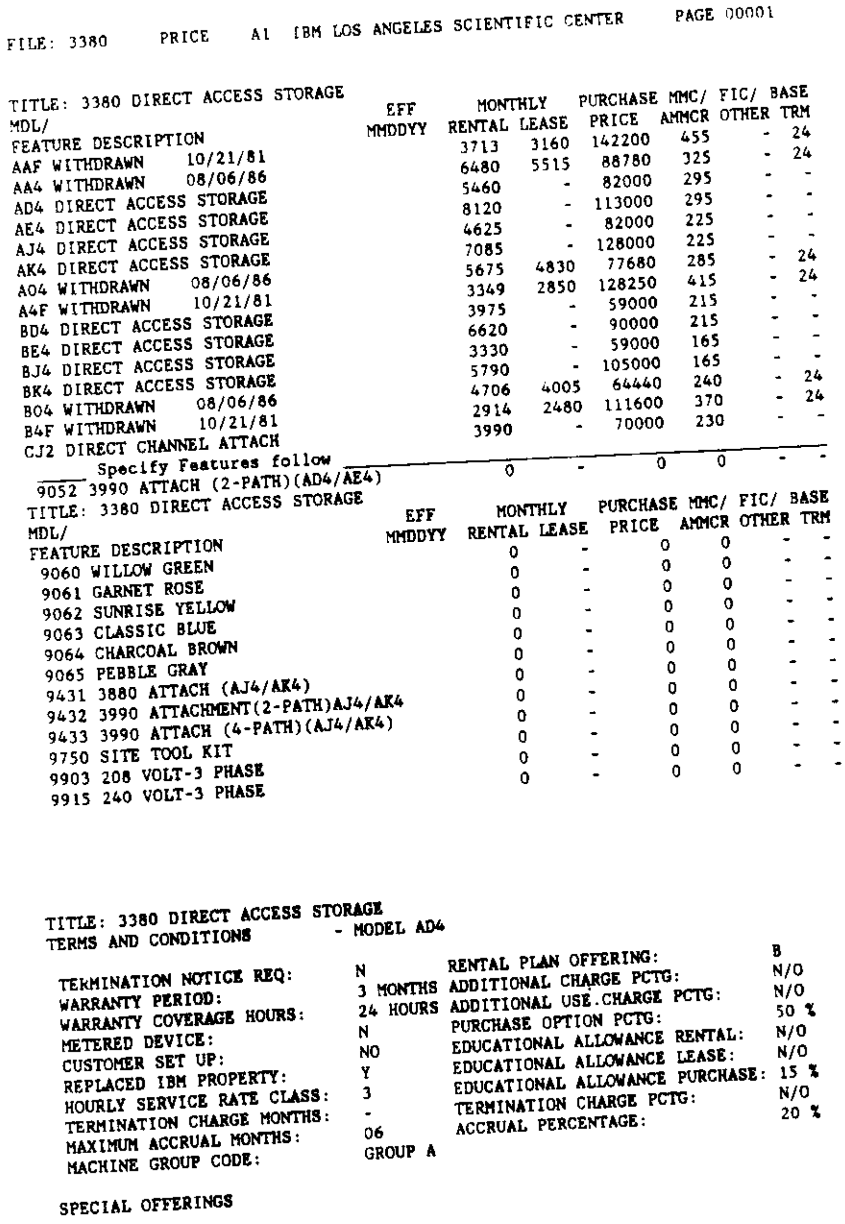
Pricing Estimates for IBM Hardware

This appendix provides sample data from which cost information was obtained for the purpose of cost/performance optimization in the *Configuration Synthesis* work.

We give price listings for several 4381 and 4361 processors, as well as 3380 and 3370 direct access devices. In addition, we provide an illustrative example of complete system pricing for a specific configuration.



```
FILE: 4361     PRICE    A1  IBM LOS ANGELES SCIENTIFIC CENTER      PAGE 00001


TITLE: 4361 IBM PROCESSOR
MDL/                                  EFF      MONTHLY    PURCHASE MMC/  FIC/  BASE
FEATURE DESCRIPTION                  MMDDYY  RENTAL LEASE   PRICE  AMMCR OTHER TRM
K03 WITHDRAWN      05/29/87                   4790     -    39600   330     -    -
K04 WITHDRAWN      05/29/87                  11550     -    88800   550     -    -
K05 WITHDRAWN      05/29/87                  15350     -   118400   662     -    -
LK4 WITHDRAWN      05/29/87                  14560     -   112300   666     -    -
LK5 WITHDRAWN      05/29/87                  18370     -   138400   778     -    -
L03 WITHDRAWN      05/29/87                   6300     -    49600   388     -    -
L04 WITHDRAWN      05/29/87                  13060     -    98800   608     -    -
L05 WITHDRAWN      05/29/87                  16860     -   128400   720     -    -
ML4 WITHDRAWN      05/29/87                  19080     -   142300   841     -    -
ML5 WITHDRAWN      05/29/87                  22880     -   168400   953     -    -
M04 WITHDRAWN ·    05/29/87                  16070     -   122300   724     -    -
M05 WITHDRAWN      05/29/87                  19860     -   148400   837     -    -
N04 WITHDRAWN      05/29/87                  22590     -   166150   957     -    -
N05 WITHDRAWN      05/29/87                  26390     -   192250  1070     -    -
_____ Features follow ____________________________________________________________
 1001 ADAPTER POWER PREREQ                     120   102     1815  9.50     -   24
 1002 ADAPTER LOGIC PREREQ                     223   190     3340    19     -   24
TITLE: 4361 IBM PROCESSOR
MDL/                                  EFF      MONTHLY    PURCHASE MMC/  FIC/  BASE
FEATURE DESCRIPTION                  MMDDYY  RENTAL LEASE   PRICE  AMMCR OTHER TRM
 1020 AUTOCALL UNIT INTERFACE                   18    17      230  3.50     -   24
 1100 FLOAT-POINT MULTI ACCELERTOR             656     -     5950    21     -    -
 1200 AUTO START                                92     -      840     5     -    -
 1421 BLOCK MULTIPLEXER CHANNEL                239   190     2340     3     -   24
 1422 BLK MTPLXR CHANNEL,ADD'L                 239   224     2340     3     -   24
 1431 HI SPEED BLK MLPXR CHANNEL               373   297     3330  3.50     -   24
 1432 HIGH SP/BLK MPX CHANNEL 2                373     -     3330  3.50     -    -
 1433 HIGH SP/BLK MPX CHANNEL 3                373     -     3330  3.50     -    -
 1480 BOOK RACK&CABLE HOLDER                     -     -       25     -     -    -
 1550 CONSOLE TABLE                              -     -      395     -     -    -
 1601 COMMUNICATIONS ADAPTER,BASE              165   132     1630     3     -   24
 1605 LINE GROUP ADDITIONAL                    334   284     6070    40     -   24
 1901 CONTROL STORAGE EXPANSION                280   238     3865    63     -   24
 2001 DISPLAY/PRINTER ADAPTER EXP               55    47      920     3     -   24
 2002 WORK STATION ADAPTER                     578     -     5250    33     -    -
 2833 RSF - EXTERNAL MODEM                       0     -        0     0     0    -
 2835 COMMUNICATION LINE ISOLATION               0     -        0     0     0    -
```



```
FILE: 3380    PRICE    A1  IBM LOS ANGELES SCIENTIFIC CENTER    PAGE 00001

TITLE: 3380 DIRECT ACCESS STORAGE                                 PURCHASE MMC/  FIC/  BASE
MDL/                                  EFF      MONTHLY            PRICE    AMMCR OTHER TRM
FEATURE DESCRIPTION                   MMDDYY   RENTAL  LEASE
AAF WITHDRAWN     10/21/81                     3713    3160    142200   455     -    24
AA4 WITHDRAWN     08/06/86                     6480    5515     88780   325     -    24
AD4 DIRECT ACCESS STORAGE                      5460       -     82000   295     -     -
AE4 DIRECT ACCESS STORAGE                      8120       -    113000   295     -     -
AJ4 DIRECT ACCESS STORAGE                      4625       -     82000   225     -     -
AK4 DIRECT ACCESS STORAGE                      7085       -    128000   225     -     -
A04 WITHDRAWN     08/06/86                     5675    4830     77680   285     -    24
A4F WITHDRAWN     10/21/81                     3349    2850    128250   415     -    24
BD4 DIRECT ACCESS STORAGE                      3975       -     59000   215     -     -
BE4 DIRECT ACCESS STORAGE                      6620       -     90000   215     -     -
BJ4 DIRECT ACCESS STORAGE                      3330       -     59000   165     -     -
BK4 DIRECT ACCESS STORAGE                      5790       -    105000   165     -     -
B04 WITHDRAWN     08/06/86                     4706    4005     64440   240     -    24
B4F WITHDRAWN     10/21/81                     2914    2480    111600   370     -    24
CJ2 DIRECT CHANNEL ATTACH                      3990       -     70000   230     -     -
_____ Specify Features follow ___________________________________________________________
 9052 3990 ATTACH (2-PATH)(AD4/AE4)               0       -         0     0     -     -
TITLE: 3380 DIRECT ACCESS STORAGE                                 PURCHASE MMC/  FIC/  BASE
MDL/                                  EFF      MONTHLY            PRICE    AMMCR OTHER TRM
FEATURE DESCRIPTION                   MMDDYY   RENTAL  LEASE
 9060 WILLOW GREEN                                0       -         0     0     -     -
 9061 GARNET ROSE                                 0       -         0     0     -     -
 9062 SUNRISE YELLOW                              0       -         0     0     -     -
 9063 CLASSIC BLUE                                0       -         0     0     -     -
 9064 CHARCOAL BROWN                              0       -         0     0     -     -
 9065 PEBBLE GRAY                                 0       -         0     0     -     -
 9431 3880 ATTACH (AJ4/AK4)                       0       -         0     0     -     -
 9432 3990 ATTACHMENT(2-PATH)AJ4/AK4              0       -         0     0     -     -
 9433 3990 ATTACH (4-PATH)(AJ4/AK4)               0       -         0     0     -     -
 9750 SITE TOOL KIT                               0       -         0     0     -     -
 9903 208 VOLT-3 PHASE                            0       -         0     0     -     -
 9915 240 VOLT-3 PHASE                            0       -         0     0     -     -


TITLE: 3380 DIRECT ACCESS STORAGE
TERMS AND CONDITIONS             - MODEL AD4

 TERMINATION NOTICE REQ:        N         RENTAL PLAN OFFERING:                 B
 WARRANTY PERIOD:               3 MONTHS  ADDITIONAL CHARGE PCTG:               N/O
 WARRANTY COVERAGE HOURS:       24 HOURS  ADDITIONAL USE.CHARGE PCTG:           N/O
 METERED DEVICE:                N         PURCHASE OPTION PCTG:                 50 %
 CUSTOMER SET UP:               NO        EDUCATIONAL ALLOWANCE RENTAL:         N/O
 REPLACED IBM PROPERTY:         Y         EDUCATIONAL ALLOWANCE LEASE:          N/O
 HOURLY SERVICE RATE CLASS:     3         EDUCATIONAL ALLOWANCE PURCHASE:       15 %
 TERMINATION CHARGE MONTHS:     -         TERMINATION CHARGE PCTG:              N/O
 MAXIMUM ACCRUAL MONTHS:        06        ACCRUAL PERCENTAGE:                   20 %
 MACHINE GROUP CODE:            GROUP A

 SPECIAL OFFERINGS
```

```
FILE: 3370     PRICE    A1  IBM LOS ANGELES SCIENTIFIC CENTER      PAGE 00001

TITLE: 3370 DIRECT ACCESS STORAGE
MDL/                                EFF     MONTHLY    PURCHASE MMC/  FIC/  BASE
FEATURE DESCRIPTION                MMDDYY RENTAL LEASE   PRICE  AMMCR OTHER TRM
A01 WITHDRAWN     11/05/86                 1980  1685   35480   173      -   24
A02 DIRECT ACCESS STORAGE                  2340     -   35480   134      -    -
A11 WITHDRAWN     11/05/86                 1980  1685   35480   173      -   24
A12 DIRECT ACCESS STORAGE                  2570     -   35480   139      -    -
B01 WITHDRAWN     11/05/86                 1481  1260   26600   129      -   24
B02 DIRECT ACCESS STORAGE                  1750     -   26600   101      -    -
B11 WITHDRAWN     11/05/86                 1481  1260   26600   129      -   24
B12 DIRECT ACCESS STORAGE                  1925     -   26600   105      -    -
 _____ Features follow ____________________________________________________
 113WITHDRAWN PRIOR TO 02/25/86             240     -         15.50      -    -
 2907 WITHDRAWN  11/05/86                     0     -       0     0      -    -
 8150 STRING SWITCH                         193   164    3830  1.50      -   24
 _____ Specify Features follow ____________________________________________
 2800 POWER 220V 60 HZ 3 PHASE                0     -       0     0      0    -
 9491 SYSTEM ATTACH FOR 3090                  0     -       0     0      -    -
 9986 CHICAGO POWER CORD                      0     -       0     0      0    -

TITLE: 3370 DIRECT ACCESS STORAGE
TERMS AND CONDITIONS          - MODEL A01

 TERMINATION NOTICE REQ:       -        RENTAL PLAN OFFERING:             -
 WARRANTY PERIOD:              3 MONTHS ADDITIONAL CHARGE PCTG:           N/O
 WARRANTY COVERAGE HOURS:      -        ADDITIONAL USE CHARGE PCTG:       N/O
 METERED DEVICE:               -        PURCHASE OPTION PCTG:             N/O
 CUSTOMER SET UP:              NO       EDUCATIONAL ALLOWANCE RENTAL:     15 %
 REPLACED IBM PROPERTY:        -        EDUCATIONAL ALLOWANCE LEASE:      N/O
 HOURLY SERVICE RATE CLASS:    3        EDUCATIONAL ALLOWANCE PURCHASE:   15 %
 TERMINATION CHARGE MONTHS:    00       TERMINATION CHARGE PCTG:          N/O
 MAXIMUM ACCRUAL MONTHS:       00       ACCRUAL PERCENTAGE:               40 %
 MACHINE GROUP CODE:           GROUP A
```

(LAST RESTORED FILE = UC4381)   (USERID = SG6FCC)

Proposal Usage

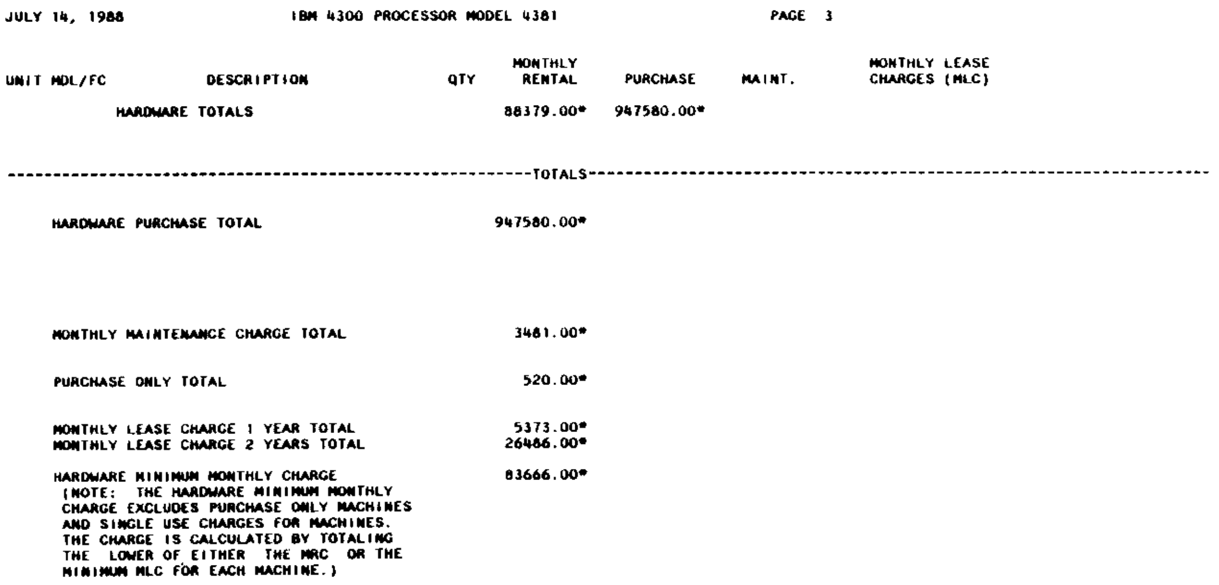
IBM Aids programs classified for Proposal Usage may be used by IBM personnel to support marketing recommendations that result in proposals for IBM equipment, programs, and lease or purchase plan offerings, or to revalidate such proposals. Reports generated by these Aids may be included directly in proposals or presentations.

JULY 14, 1988 IBM 4300 PROCESSOR MODEL 4381 PAGE 1

UNIT MDL/FC	DESCRIPTION	QTY	MONTHLY RENTAL	PURCHASE	MAINT.	MONTHLY LEASE CHARGES (MLC)
					MONTHLY	
4381-P13	PROCESSOR 16,777,216 BYTES	1	54470.00	445000.00	690.00	
1480	BOOKRACK AND CABLE HOLDER +	1	N/O	25.00P	N/O	
1550	CONSOLE TABLE	1	N/O	495.00P	N/O	
1870	BLOCK MULTIPLEX/CH,ADDL	1	2710.00	35580.00	12.50	
9063	COVER - COLOR CLASSIC BLUE	1	NC	NC	NC	
9514	INTEGRATED MODEM - AUTO ANS	1	NC	NC	NC	
9903	VOLTAGE FEATURE	1	NC	NC	NC	
			57180.00*	481100.00*	702.50*	
					MONTHLY	MLC2
3279-02C	COLOR DISP CONSOLE	1	264.00	2500.00	29.00	225.00
4632	KYBD,75KEY-OP CON W/O CH-CH	1	63.00	909.00	5.50	54.00
			327.00*	3409.00*	34.50*	279.00*
					MONTHLY	MLC2
3380-AA4	DIRECT ACCESS STORAGE	1	6480.00	88780.00	325.00	5515.00
9063	CLASSIC BLUE	1	NC	NC	NC	NC
9903	208 VOLT-3 PHASE	1	NC	NC	NC	NC
			6480.00*	88780.00*	325.00*	5515.00*
					MONTHLY	MLC2
3380-B04	DIRECT ACCESS STORAGE	1	4706.00	64440.00	240.00	4005.00
9063	CLASSIC BLUE	1	NC	NC	NC	NC
9903	208 VOLT-3 PHASE	1	NC	NC	NC	NC
			4706.00*	64440.00*	240.00*	4005.00*
					MONTHLY	MLC2
3380-AA4	DIRECT ACCESS STORAGE	1	6480.00	88780.00	325.00	5515.00
9063	CLASSIC BLUE	1	NC	NC	NC	NC
9903	208 VOLT-3 PHASE	1	NC	NC	NC	NC
			6480.00*	88780.00*	325.00*	5515.00*
					MONTHLY	MLC2
3380-B04	DIRECT ACCESS STORAGE	1	4706.00	64440.00	240.00	4005.00
9063	CLASSIC BLUE	1	NC	NC	NC	NC
9903	208 VOLT-3 PHASE	1	NC	NC	NC	NC
			4706.00*	64440.00*	240.00*	4005.00*

JULY 14, 1988 IBM 4300 PROCESSOR MODEL 4381 PAGE 2

UNIT MDL/FC	DESCRIPTION	QTY	MONTHLY RENTAL	PURCHASE	MAINT.	MONTHLY LEASE CHARGES (MLC)	
					MONTHLY	MLC2	
3880-003	STORAGE CONTROL-3380.2 ST/DR	1	1370.00	51000.00	176.00	1165.00	
8170	2 CHANNEL SWITCH-PAIR	1	91.00	4500.00	11.00	77.00	
9063	CLASSIC BLUE	1	NC	NC	NC	NC	
9193	3380 W/O SPEED MATCH BUFFER	2	NC	NC	NC	NC	
9903	POWER 208V 60HZ 3 PHASE	1	NC	NC	NC	NC	
			1461.00*	55500.00*	187.00*	1242.00*	
					MONTHLY	MLC1	MLC2
3420-008	TAPE UNIT	2	3130.00	43720.00	802.00	2880.00	2630.00
6425	FOR 6250/1600 DENSITY	2	322.00	4850.00	198.00	296.00	270.00
9043	BLUE COVERS	2	NC	NC	NC	NC	NC
9903	POWER 208V 60HZ 3 PHASE	2	NC	NC	NC	NC	NC
			3452.00*	48570.00*	1000.00*	3176.00*	2900.00*
					MONTHLY	MLC1	MLC2
3803-002	TAPE CONTROL	1	2080.00	30300.00	218.00	1914.00	1747.00
8100	TWO CHANNEL SWITCH	1	308.00	5060.00	6.50	283.00	258.00
9043	BLUE COVERS	1	NC	NC	NC	NC	NC
9903	POWER 208V 60HZ 3 PHASE	1	NC	NC	NC	NC	NC
			2388.00*	35360.00*	224.50*	2197.00*	2005.00*
					MONTHLY	MLC2	
3262-005	PRINTER CH SYSTEM ATTACHED	1	1193.00	17000.00	202.50	1015.00	
1090	AUDIBLE ALARM	1	6.00	201.00	N/O	5.00	
9536	96 CHAR. SET INTERNATIONAL	1	NC	NC	NC	NC	
9950	2.4MM(.095IN.) CHAR.SET HEIGHT	1	NC	NC	NC	NC	
			1199.00*	17201.00*	202.50*	1020.00*	



```
JULY 14, 1988             IBM 4300 PROCESSOR MODEL 4381                  PAGE  3

                                                 MONTHLY                                     MONTHLY LEASE
UNIT MDL/FC          DESCRIPTION          QTY     RENTAL      PURCHASE      MAINT.           CHARGES (MLC)
             HARDWARE TOTALS                    88379.00*   947580.00*


-------------------------------------------------------TOTALS------------------------------------------------------------

     HARDWARE PURCHASE TOTAL                   947580.00*



     MONTHLY MAINTENANCE CHARGE TOTAL            3481.00*

     PURCHASE ONLY TOTAL                          520.00*

     MONTHLY LEASE CHARGE 1 YEAR TOTAL           5373.00*
     MONTHLY LEASE CHARGE 2 YEARS TOTAL         26486.00*

     HARDWARE MINIMUM MONTHLY CHARGE            83666.00*
      (NOTE:  THE HARDWARE MINIMUM MONTHLY
      CHARGE EXCLUDES PURCHASE ONLY MACHINES
      AND SINGLE USE CHARGES FOR MACHINES.
      THE CHARGE IS CALCULATED BY TOTALING
      THE  LOWER OF EITHER  THE MRC  OR THE
      MINIMUM MLC FOR EACH MACHINE.)
```

CUSTOMER FINANCING ALTERNATIVES
In addition to the prices stated above, IBM Credit Corporation financing programs are offered for most IBM machines.
TERM LEASES
- 3, 4, or 5 year term for most machines
- Fixed monthly payments
- Flexible upgrade choices
- Renewal and purchase options
- Tailored financing for transactions over $250,000

INSTALLMENT PAYMENT PLANS
- 10 percent minimum down payment
- maximum financing term of 60 months for most machines
- $5,000 minimum financed amount
- Separate plan for state and local government.

The prices stated are for your information only and are subject to change. Applicable taxes are not shown. Purchase/Lease or Rental of IBM products will be by agreement signed by the customer and IBM either prior to or subsequent to this date, with the prices governed by the price protection therein. Licensed Programs are available only under the Agreement for IBM Licensed Programs signed by the customer and IBM, or the IBM Program License Agreement.

NC - NO CHARGE
N/O - NOT OFFERED
P - PURCHASE ONLY
+ - ACCESSORY FEATURES
MMMC - MINIMUM MONTHLY MAINTENANCE CHARGE.

SUMMARY OF WITHDRAWN HARDWARE PRODUCTS

The following machines/features have been withdrawn from marketing or are scheduled for future withdrawal:

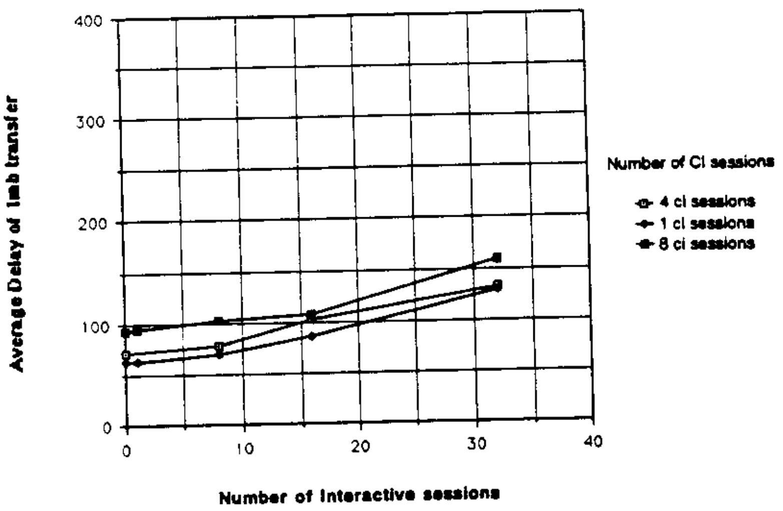
UNIT-MDL	FEAT	DESCRIPTION	WITHDRAWAL DATE
4381-P13		PROCESSOR 16,777,216 BYTES	06/15/88
3279-02C		COLOR DISP CONSOLE	04/26/88
3279-02C	4632	KYBD,75KEY-OP CON W/O CH-CH	04/26/88
3380-AA4		DIRECT ACCESS STORAGE	08/06/86
3380-B04		DIRECT ACCESS STORAGE	08/06/86
3420-008		TAPE UNIT	01/19/88
3803-002		TAPE CONTROL	01/19/88

APPENDIX E

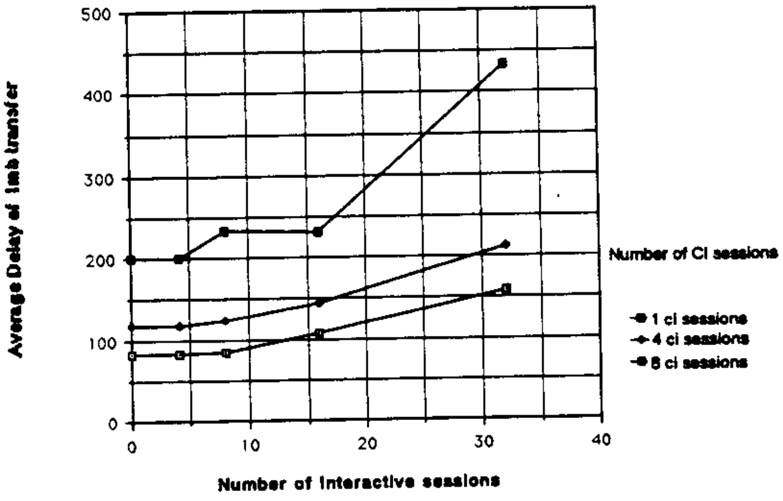
Simulation Results for 3 User Class Combinations

In this appendix we present the simulation results that rendered the 3-space tables representing the 3-class mix of user load. This data was later input to the optimization algorithm, whose program is presented in Appendix F. Such space tables give the necessary system parameterization for configuration synthesis

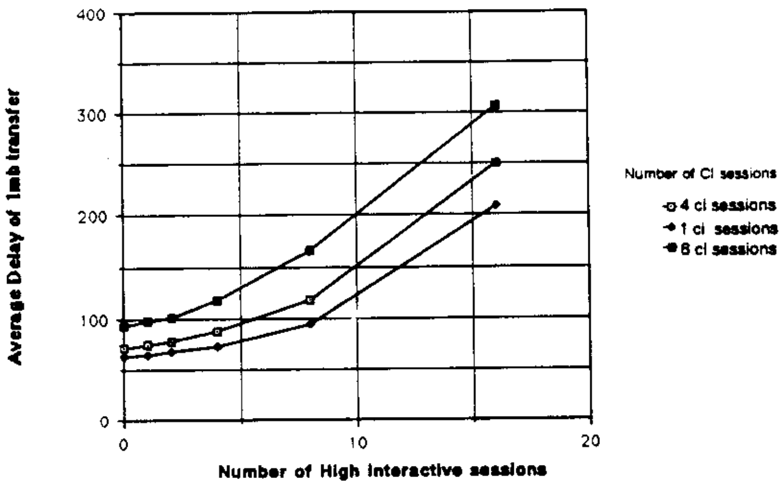
4381 Comm. Intensive Delay x Int. sessions



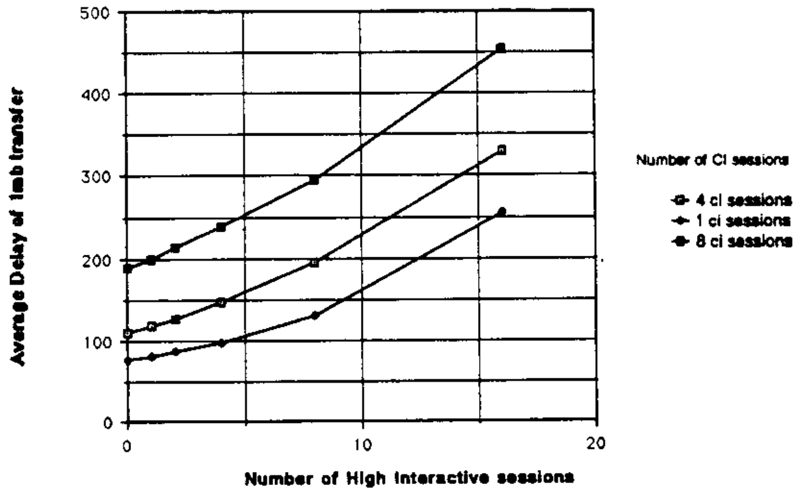
4361 Comm. Intensive Delay x Int. sessions



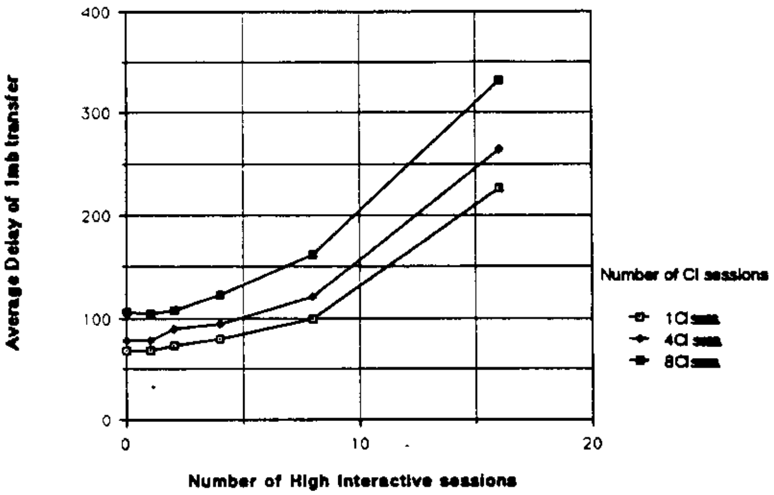
4381 Comm. Intensive Delay x HI sessions



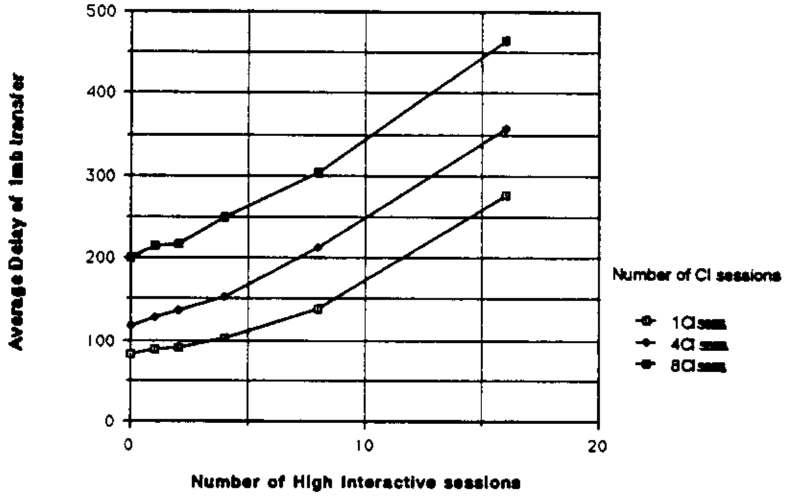
4361 Comm. Intensive Delay x HI sessions



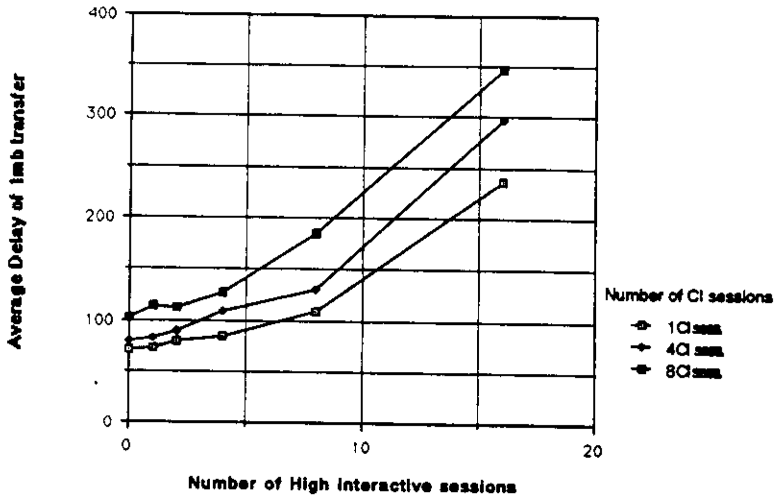
4381 Comm. Int. Delay with 4 int x High Int. ses.



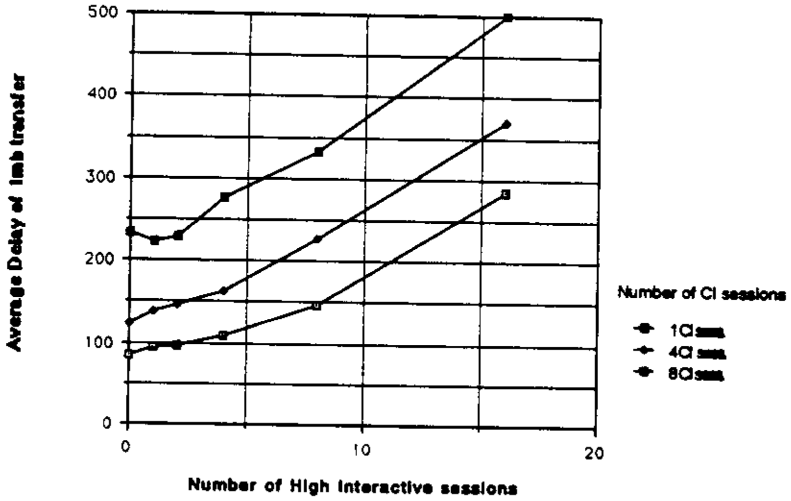
4361 Comm. Int. Delay with 4 int x High Int. ses.



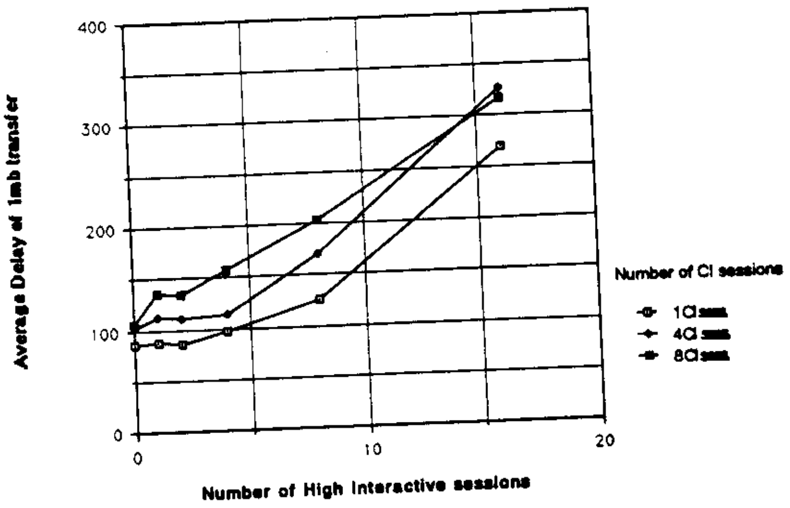
4381 Comm. Int. Delay with 8 int x High Int. ses.



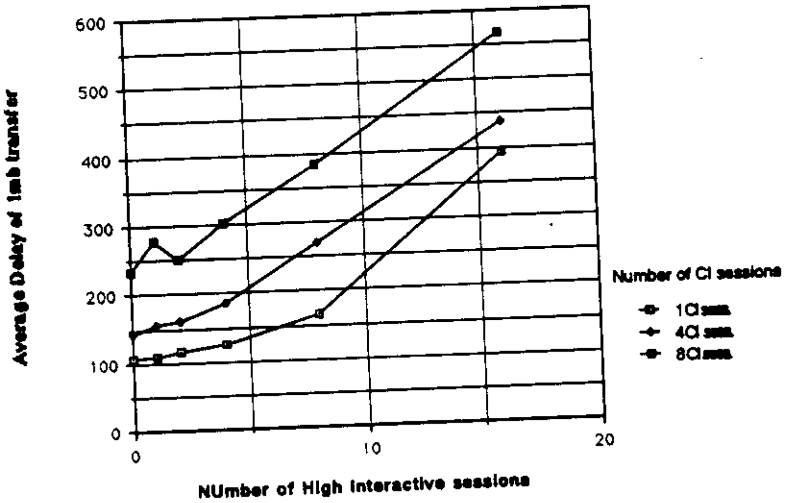
4361 Comm. Int. Delay with 8 int x High Int. ses.



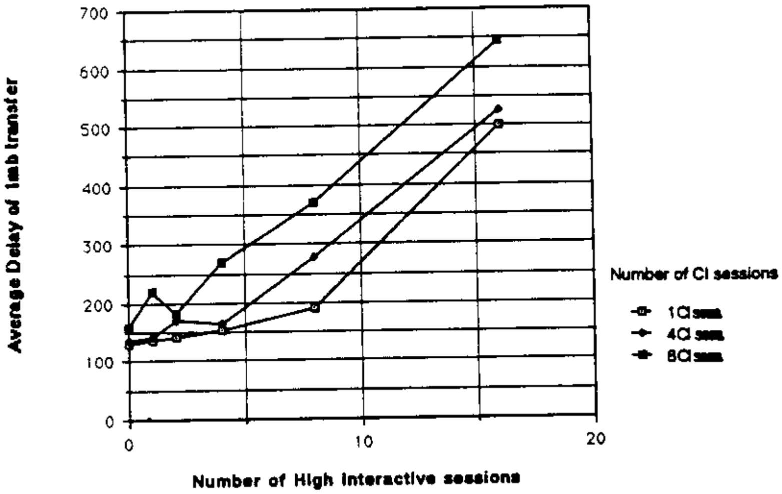
4381 Comm. Int. Delay with 16 int. x High Int. ses.



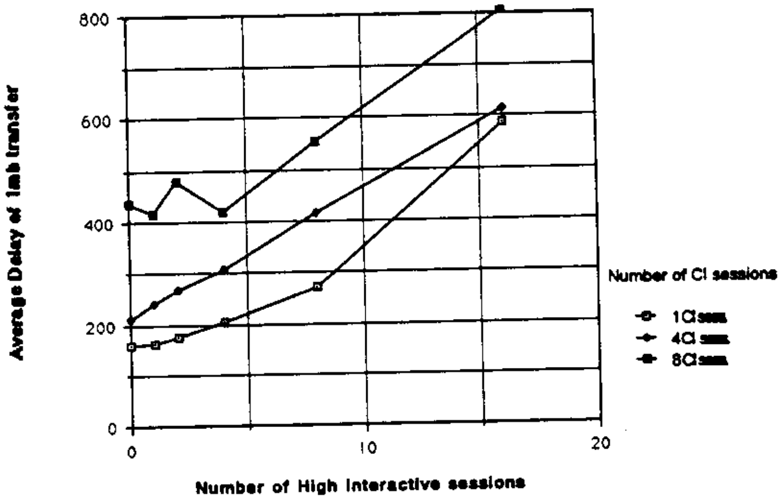
4361 Comm. Intensive Delay with 16 int x High Int. ses.



4381 Comm. Int. Delay with 32 int x High Int. ses.



4361 Comm. Intensive Delay with 32 int. x High Int. ses.



APPENDIX F

Optimization Program and Optimization Results

We present in this appendix the complete listing of the sub optimal configuration synthesis program. The program computes delay and costs and sub-optimal session assignment for PCI service with the objective function of minimum delay per Dollar spent..

We also present sone listings of the program computations, showing the various backbone numbers, costs, and convergence to local minima (or divergence in absence of local minima).

Lastly we present resulting cost/delay optimizations for a network of 6 and 10 backbone servers.

```
#include "stdio.h"
#define links 10

#define maxss 18
#define maxb  1
#define osci 10
#define tmax   1000.0 /* 100 per session - 90 sessions on 6 sites */
#define epslon 1.0

FILE *outfile;
float delays0_2[36],delays0_3[36],delays0_5[36],delays0_6[36],delays0_7[36],
      delays4_2[36],delays4_3[36],delays4_5[36],delays4_6[36],delays4_7[36],
      delays8_2[36],delays8_3[36],delays8_5[36],delays8_6[36],delays8_7[36],

      delays12_1[36],delays12_4[36],delays12_8[36],
      delays12_2[36],delays12_3[36],delays12_5[36],delays12_6[36],delays12_7[36],

      delays16_2[36],delays16_3[36],delays16_5[36],delays16_6[36],delays16_7[36];

float delays0_1 [] =  /* 0 interactive 1 CI x hi */
{76.6, 80.3,86.2,91.6,97.0,105.2, 113.5,121.8,130.0,145.75,161.5,177.2,193.0,208.7,
 224.5,240.3, 256.0,999.0,/*4361*/

 /* 0      1     2   3    4     5      6    7      8     9     10    11     12     13
  14    15      16 */

 62.5, 65.0,67.5,70.5,73.5,78.7 , 83.9,89.1,   94.3,108.55,122.8,137.0,  151.3,165.5,
 179.8 , 194.0 , 208.3,999.0 /* 4381*/
                  };
 float delays0_4 [] =  /* 0 interactive 4 CI x hi */
 {

 109.8,117.6,126.5,136.7,147.0,158.7,170.5,182.25,194.0,211.0,228.1,245.1,262.2,279.2,
   296.3, 313.4 , 330.5 ,999.0,/*4361*/
/*  0      1     2   3    4     5      6    7      8     9     10    11     12     13
  14    15      16 */
  71.0,75.0,78.4,83.2 ,88.0,95.4,102.8,110.2,117.6, 134.1  ,150.7, 167.2 ,183.8,200.4,
  217.0, 233.4,250.0,999.0 /* 4381*/
              };
 float delays0_8 [] =  /* 0 interactive 8 CI x hi */
 {
 188.6,200.0,212.7,225.35,238.0,252.0,266.0,280.0,294.1,314.0,334.0,354.0 ,374.0,394.0
  414.0,434.0,454.4,999.0,/*4361*/
/*  0      1     2   3    4     5      6    7      8     9     10    11     12     13
  14    15      16 */
 92.5,98.0,102.0, 109.8,117.6, 129.8,142.1,154.35,166.6,184.2,201.8 ,219.5,237.1,254.7
  272.3,290.0,307.6 ,999.0 /* 4381*/
              };
  float delays4_1 [] =  /* 4 interactive 1 CI x hi */
 {
 83.05,88.50,90.9,97.0,103.1,112.05,121.0,130.0,138.9,156.25,173.6,190.95,208.3,225.65
 243.0,260.35,277.7,999.0,/*4361*/
/*  0      1     2   3    4     5      6    7      8     9     10    11     12     13
  14    15      16 */
 68.0, 70.5,72.9, 75.9,78.9,83.85,88.8,  93.75 ,98.7,114.8,131.0, 147.0 ,163.0,179.1,
 195.1,211.2,227.2,999.0 /* 4381*/
```

```
              };

 float delays4_4 [] =  /* 4 interactive 4 CI x hi */
 {
 117.0,128.2,137.0,144.5,152.0,167.2,182.4,198.0,212.7,231.0,249.0,267.0,285.0,303.0,
  321.0,339.0,357.1,999.0,/*4361*/
/*  0     1    2   3    4     5       6    7       8      9      10     11      12      13
  14    15      16 */
    78.4,84.0,89.2,91.7,94.3,100.8,107.3,113.85,120.4, 138.5 ,156.6, 174.7 ,192.8,210.
  229.0, 247.1, 265.2,999.0 /* 4381*/
              };


 float delays4_8 [] =  /* 4 interactive 8 CI x hi */
 {
 200.0,215.0,217.0,233.5,250.0,263.2,276.5,290.0,303.0,323.2,343.5,364.0,384.0,404.0,
  424.5,445.0,465.1,999.0,/*4361*/
/*  0     1    2   3    4     5       6    7       8      9      10     11      12      13
  14    15      16 */
 105.2,106.0,107.5,114.7,122.0, 132.1,142.3,152.4,162.6,183.9,205.2,226.5,247.8,269.1,
  290.4,311.7,333.0,999.0 /* 4381*/
              };

 float delays8_1 [] =  /* 8 interactive 1 CI x hi */
 {
 84.3,95.1,96.1,102.3,108.5,118.1,127.8,137.4,147.0,164.3,181.7,199.0,216.35,233.7,
  251.0,268.4,285.7,999.0,/*4361*/
/*  0  1    2      3     4     5       6    7       8      9      10     11      12   13
  14    15      16 */
 70.3,72.6,78.7,81.35,84.0, 90.2,96.35,102.5,108.7,124.6,140.6,156.55,172.5, 188.5,
  204.5,220.4,236.4 ,999.0 /* 4381*/
              };

  float delays8_4 [] =  /* 8 interactive 4 CI x hi */
 {
 124.22,138.8,147.6,155.4,163.2,179.2,195.2,211.2,227.2,245.1,263.0, 280.85,298.7,316.
  334.5,352.4,370.3,999.0,/*4361*/
/*  0      1    2      3     4     5       6    7       8      9      10     11      12   13
  14    15      16 */
  78.7,  82.0,89.6, 99.1,108.7,114.0,119.3,124.5,130.0,151.0,172.0, 193.0,213.8,234.7,
  255.7,276.65,297.6,999.0 /* 4381*/
              };

  float delays8_8 [] =  /* 8 interactive 8 CI x hi */
 {
 223.5,223.7,229.8,253.7,277.7,291.6,305.5,319.4 ,333.3, 354.0,375.0,396.0,416.7,437.5
   458.3,479.1,500.0,999.0,/*4361*/
/*  0      1    2      3     4     5       6    7       8      9      10     11      12   13
  14    15      16 */
 102.7,107.0,112.2,120.0,127.7,142.0,156.4,170.75,185.1,205.3, 225.6,245.9 ,266.15,286
   306.7,327.0,347.2,999.0 /* 4381*/
              };

 float delays16_1 [] =  /* 16 interactive 1 CI x hi */
 {
  107.6,109.5,115.4,120.5,126.0,136.2,146.3,156.5,166.6,195.0,223.4,251.8,280.15,308.6
```

```
    337.0,365.0,393.7,999.0,/*4361*/
/*  0       1     2      3      4      5        6     7        8       9       10      11     12     13
  14     15      16  */
   86.7,88.0, 90.0,  91.8,97.8,104.6,111.4,118.2,125.0, 143.0,162.0, 180.0,197.5,216.0
   234.0,252.0,270.0,999.0 /* 4381*/
                 };

  float delays16_4 [] =  /* 16 interactive 4 CI x hi */
 {
  145.0,156.2,162.0,173.5,185.1,206.4,227.7,249.0,270.2,292.0,313.0,334.0,355.1,377.0,
  398.0,419.0,440.0,999.0,/*4361*/
/*  0       1     2      3      4      5        6     7        8       9       10      11      12    13
  14     15      16 */
  102.8,106.0,111.1,112.2,113.3,127.5,142.0,156.2,170.4, 190.2,210.0,230.0,249.2,269.0
  288.5,308.2,327.8,999.0 /* 4381*/
                 };


    float delays16_8 [] =  /* 16 interactive 8 CI x hi */
  {
   232.0,277.7,250.0,276.5,303.0,323.5,344.0,364.3,384.6,408.0,430.5,442.0,476.4,499.0,
    522.3,545.0,568.1,999.0,/*4361*/
/*  0       1     2      3      4      5        6     7        8       9       10      11      12    13
  14     15      16 */
  106.0,120.0,133.3,145.0,156.0,168.0,180.0,192.0,204.0,219.0,233.0,247.0,261.0,276.0,
  290.0,304.0,318.0,999.0 /* 4381*/
                 };


 float cost[] = {300.0, /*4361 */
                  700.0};/*4381 */
 int backbone[links]={0}; /* backbone assigned to link i */
 /* total 90 */
 /*48 - li
   24 - hi
   18 - ci  */
 int /* nseshi[links]={   0, 0,   0,   0 , 8 , 16 },
     nsesli[links]={  16, 16 , 16  ,0, 0 , 0  },
     nsesci[links]={   0,  0,   0,  8 ,8 , 2 } ;*/  /*6 sites allocation vector of flow


 /*  nseshi[links]={    0, 0,   12, 12  },
     nsesli[links]={    16, 16 ,  8  ,8  },
     nsesci[links]={    4, 4,  5, 5 } ; 4 sites allocation vector of flows to links*/


  nseshi[links]={1, 1, 1, 1, 1, 5,  5, 5 ,5,    5 },
     nsesli[links]={8, 8, 8 ,8, 8, 0, 0,  0, 0,   0},
     nsesci[links]={2, 2, 2, 2, 2, 2, 2,  2, 2,   2} /*10 sites*/; /* allocation vector
 int  deltci = 1,
      delthi = 1,
      deltli = 4;
      cmax;
 float delaylink(seshi,sesli,sesci,backbone)
    int seshi,sesli,sesci,backbone;
```

```
{
  int i;
   if (backbone == -1) {
     printf("INVALID INDEX IN DELAYLINK no backbone\n");
     exit();
  }

  i = (maxss*backbone+seshi);
 /* printf("delaylinks seshi = %d sesci = %d sesli = %d backbone = %d  \n",
            seshi,sesci,sesli,backbone); */
  if  ( (i > 35)  || ( i < 0) )  {
     printf("INVALID INDEX IN DELAYLINK\n");
     exit();
  }
  switch (sesci)
  {
   case 0:
   case 1:
      switch (sesli)
      {
        case 0:return delays0_1[i]; break;
        case 4:return delays4_1[i]; break;
        case 8:return delays8_1[i]; break;
        case 12:return delays12_1[i]; break;
        case 16:return delays16_1[i]; break;
     /*    case 20:return delays20_1[i]; break;
        case 24:return delays24_1[i]; break;
        case 28:return delays28_1[i]; break;
        case 32:return delays32_1[i]; break; */
        default: /* printf("invalid sesli\n");
                 exit();*/
             return (delays16_1[i]*2); break;

       }
       break;
   case 2:
    switch (sesli)
      {
        case 0:return delays0_2[i]; break;
        case 4:return delays4_2[i]; break;
        case 8:return delays8_2[i]; break;
        case 12:return delays12_2[i]; break;
        case 16:return delays16_2[i]; break;
      /*  case 20:return delays20_2[i]; break;
        case 24:return delays24_2[i]; break;
        case 28:return delays28_2[i]; break;
        case 32:return delays32_2[i]; break; */
        default:/*  printf("invalid sesli\n");
                 exit(); */
               return (delays16_2[i]*2); break;
       }
       break;
   case 3:
    switch (sesli)
      {
         case 0:return delays0_3[i]; break;
         case 4:return delays4_3[i]; break;
         case 8:return delays8_3[i]; break;
         case 12:return delays12_3[i]; break;
         case 16:return delays16_3[i]; break;
       /*  case 20:return delays20_3[i]; break;
         case 24:return delays24_3[i]; break;
         case 28:return delays28_3[i]; break;
         case 32:return delays32_3[i]; break; */
         default:/* printf("invalid sesli\n");
                  exit(); */
                return (delays16_3[i]*2); break;
        }
        break;
    case 4:
    switch (sesli)
       {
         case 0:return delays0_4[i]; break;
         case 4:return delays4_4[i]; break;
         case 8:return delays8_4[i]; break;
         case 12:return delays12_4[i]; break;
         case 16:return delays16_4[i]; break;
       /*  case 20:return delays20_4[i]; break;
         case 24:return delays24_4[i]; break;
         case 28:return delays28_4[i]; break;
         case 32:return delays32_4[i]; break; */
         default: /* printf("invalid sesli\n");
                  exit();*/
                 return (delays16_4[i]*2); break;
        }
        break;
    case 5:
     switch (sesli)
       {
         case 0:return delays0_5[i]; break;
         case 4:return delays4_5[i]; break;
         case 8:return delays8_5[i]; break;
         case 12:return delays12_5[i]; break;
         case 16:return delays16_5[i]; break;
       /*   case 20:return delays20_5[i]; break;
         case 24:return delays24_5[i]; break;
         case 28:return delays28_5[i]; break;
         case 32:return delays32_5[i]; break; */
          default: /* printf("invalid sesli\n");
                  exit();*/
                 return (delays16_5[i]*2); break;
        }
        break;
    case 6:
     switch (sesli)
       {
         case 0:return delays0_6[i]; break;
         case 4:return delays4_6[i]; break;
         case 8:return delays8_6[i]; break;
         case 12:return delays12_6[i]; break;
         case 16:return delays16_6[i]; break;
```

```
   /*   case 20:return delays20_6[i]; break;
     case 24:return delays24_6[i]; break;
     case 28:return delays28_6[i]; break;
     case 32:return delays32_6[i]; break; */
     default: /* printf("invalid sesli\n");
              exit();*/
            return (delays16_6[i]*2); break;
    }
    break;
case 7:
 switch (sesli)
   {
     case 0:return delays0_7[i]; break;
     case 4:return delays4_7[i]; break;
     case 8:return delays8_7[i]; break;
     case 12:return delays12_7[i]; break;
     case 16:return delays16_7[i]; break;
 /*      case 20:return delays20_7[i]; break;
     case 24:return delays24_7[i]; break;
     case 28:return delays28_7[i]; break;
     case 32:return delays32_7[i]; break; */
     default: /* printf("invalid sesli\n");
              exit();*/
             return (delays16_7[i]*2); break;
    }
    break;
case 8:
 switch (sesli)
   {
     case 0:return delays0_8[i]; break;
     case 4:return delays4_8[i]; break;
     case 8:return delays8_8[i]; break;
     case 12:return delays12_8[i]; break;
     case 16:return delays16_8[i]; break;
   /*   case 20:return delays20_8[i]; break;
     case 24:return delays24_8[i]; break;
     case 28:return delays28_8[i]; break;
     case 32:return delays32_8[i]; break; */
     default: /* printf("invalid sesli\n");
              exit();*/
             return (delays16_8[i]*2); break;
    }
    break;
default:
    switch (sesli)
   {
     case 0:return (delays0_8[i]*2); break;
     case 4:return (delays4_8[i]*2); break;
     case 8:return (delays8_8[i]*2); break;
     case 12:return (delays12_8[i]*2); break;
     case 16:return (delays16_8[i]*2); break;
   /*   case 20:return delays20_8[i]*2); break;
     case 24:return delays24_8[i]*2; break;
     case 28:return delays28_8[i]*2; break;
     case 32:return delays32_8[i]*2; break; */
     default: /* printf("invalid sesli\n");
```

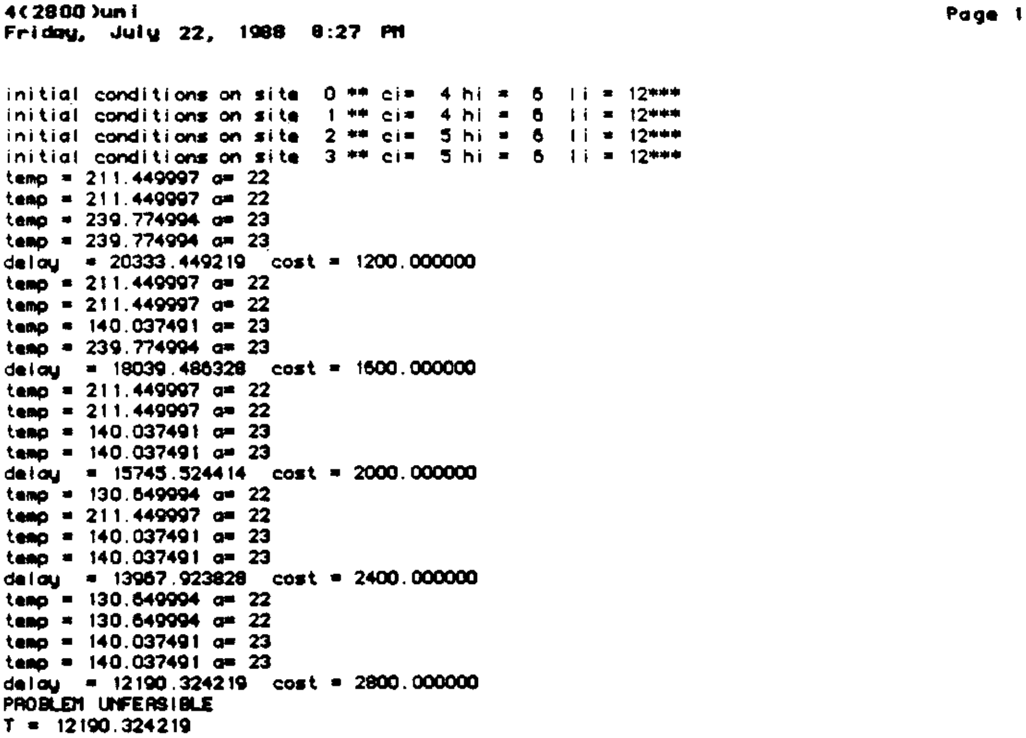
```
                exit();*/
               return (delays16_8[i]*2); break;
       }

   }

}

 /* calculate the total response time*/
 float rt()
 {
  float ttotal,temp;
  int i,a;
  ttotal = .0;
     for (i = 0 ; i < links ; i++) {
                             /*sessions,backbone current allocated*/

        if  ( (backbone[i] != -1) ) {
           a = nseshi[i] + nsesli[i] + nsesci[i] ;
           temp = delaylink(nseshi[i],nsesli[i],nsesci[i],backbone[i]);
           /* printf("temp = %f a= %d\n",temp,a);*/
           fprintf(outfile,"temp = %f a= %d\n",temp,a);
           ttotal += temp*a;
        }
     }
   return ttotal;
   }


  /* flow deviation alg. for PCI */
  float fd()
  {
   int k,a,i,imax,imin;
   float dt,temp,ttotal,max,min,
         totalant,/* total response time in previous iteration */
         dthi[links], /* derivatives of response time */
         dtli[links],
         dtci[links];
   int ccontt = 0;/* number of steps*/

   totalant = 100000.0;
        for ( ; ; ) {
              ccontt++ ;
             /* printf("Tant = %f \n",totalant);*/
              ttotal = rt();
              /* step 1 -- stop rule */
               if ( ( ttotal - totalant ) > 0  )  {
                 if ((nseshi[imax] < 0) || (nsesci[imax] < 0) || (nsesli[imax] < 0))
                 {
                   switch (k)
                  {
                   case 1:
                    nseshi[imax] +=delthi  ;
                    nseshi[imin] -=delthi  ;
                   break;
                   case 2:
```

```
       nsesli[imax] +=deltli  ;
       nsesli[imin] -=deltli  ;
      break;
      case 3:
       nsesci[imax] +=deltci  ;
       nsesci[imin] -=deltci  ;
      break;
      default:printf("Something is stinking\n");
      }
      ccontt = 21;
     }
     fprintf(outfile,"RESPONSE TIME IS INCREASING Tant = %f"
      ,totalant);
     printf("RESPONSE TIME IS INCREASING Tant = %f ",totalant);
     printf("T = %f\n",ttotal);
     fprintf(outfile,"T = %f\n",ttotal);
     printf("stop --- ttotal-- converges in %d steps  \n",ccontt);
     for( i = 0 ; i < links ; i++)
             fprintf(outfile,"op. %d (sehi=%d sesli=%d sesci=%d ,bb= %d) \n",
                             i,nseshi[i],nsesli[i],nsesci[i],backbone[i]);

   if (ccontt > osci)   return ttotal;
   else fprintf(outfile,"Try to get out of local minima++++\n");
 }
/* printf("step 2 \n----ttotal = %f ",ttotal);*/
 fprintf(outfile,"delay = %f\n",ttotal);
 if (  ( ( totalant - ttotal ) < 0.1  )  &&( (ttotal - totalant) < 0.1) ){
     printf("T = %f\n",ttotal);
     fprintf(outfile,"Converges in %d steps(epslon) Optimal delay  = %f\n",cc
     printf("stop --- ttotal-- converges in %d steps  \n",ccontt);
     for( i = 0 ; i < links ; i++)
     fprintf(outfile,"op. %d (sehi=%d sesli=%d sesci=%d ,bb= %d) \n",
                             i,nseshi[i],nsesli[i],nsesci[i],backbone[i]);
     return ttotal;
 }
 totalant = ttotal;
 /*   calculate the minimum and maximum  delay link using  the numerical deri
 min = 32767.0;
 max = 0.0;
 for (i = 0 ; i < links ; i++) {
      a = nseshi[i] + nsesli[i] + nsesci[i];
      if  (a <= 0 ) {
         dt = 32767.0 ;
      }
      else {
           temp =  delaylink(nseshi[i],nsesli[i],nsesci[i],backbone[i]) ;
           dthi[i] = (delaylink((nseshi[i]+delthi),nsesli[i],nsesci[i],
                                       backbone[i]) - temp)*nseshi[i]/delthi;
           dthi[i] += temp;
           dtli[i] = (delaylink(nseshi[i],(nsesli[i]+deltli),nsesci[i],
                                       backbone[i]) - temp)*nsesli[i]/(deltli
           dtli[i] += temp;
           dtci[i] = (delaylink(nseshi[i],nsesli[i],(nsesci[i]+deltci),
                                       backbone[i]) - temp)*nsesci[i]/deltci;
            /*  printf("*****dtci[%d] = %f *****\n",i,dtci[i]);*/
           dtci[i] += temp;
      /*  printf("derivative = %f, %f, %f, \n",dthi[i],dtli[i],dtci[i]); */
        fprintf(outfile,"derivative = %f, %f, %f, \n",dthi[i],dtli[i],dtci[


      /*   dtmin = (   dthi[i] < dtli[i] ?
         ( (dthi[i] < dtci[i]) ? dthi[i]:dtci[i] ) :
         ( (dtli[i] < dtci[i]) ? dtli[i]:dtci[i] )
                 ) ;
         dtmax = (dthi[i] > dtli[i] ?
         ( (dthi[i] > dtci[i]) ?  dthi[i]:dtci[i] ) :
         ( (dtli[i] > dtci[i]) ?  dtli[i]:dtci[i] )
                 ); */
         dt = dtci[i]+dtli[i]+dthi[i];
    }
     if (dt < min)  {
       /*  printf("change previous = %d current = %d \n",imax,i); */
        imin = i;
        min = dt;
     }
     if  ((dt > max) && (dt < 32767.0)) {
        /* printf("change previous = %d current = %d \n",imax,i);*/
        imax = i;
        max = dt;
     }
}
/* step  deviate delta from the maximum delay link to the minimum delay */
if ( (imin == imax) )  {
      printf("T = %f\n",ttotal);
      fprintf(outfile,"Converges in %d steps max = min T = %f\n",ccontt,tto
      printf("stop bottom --- nochange-- converges in %d steps  \n",ccontt)
      for( i = 0 ; i < links ; i++) {
          fprintf(outfile,"optimal design site %d (sehi=%d sesli=%d sesci=%
          ,i,nseshi[i],nsesli[i],nsesci[i],backbone[i]);
           printf("optimal design site %d (sehi=%d sesli=%d sesci=%d  ,b=%d
          ,i,nseshi[i],nsesli[i],nsesci[i] ,backbone[i]);
      }
      return ttotal;
}
if  (  (backbone[imax] == -1) || (backbone[imin] == -1) ) {
    printf("error backbone selected as max or min has no flow assigned\n");
    fprintf(outfile,"error backbone selected as max or min has no flow assi
    exit();
}



if ( dthi[imax] >=
    (dtli[imax] > dtci[imax] ? dtli[imax] : dtci[imax]) ){
     printf("deviates from class hi from %d to %d\n",imax,imin);
     fprintf(outfile,"deviates from class hi from %d to %d\n",imax,imin);

     k = 1;
     nseshi[imax] -=delthi  ;
     nseshi[imin] +=delthi  ;
```

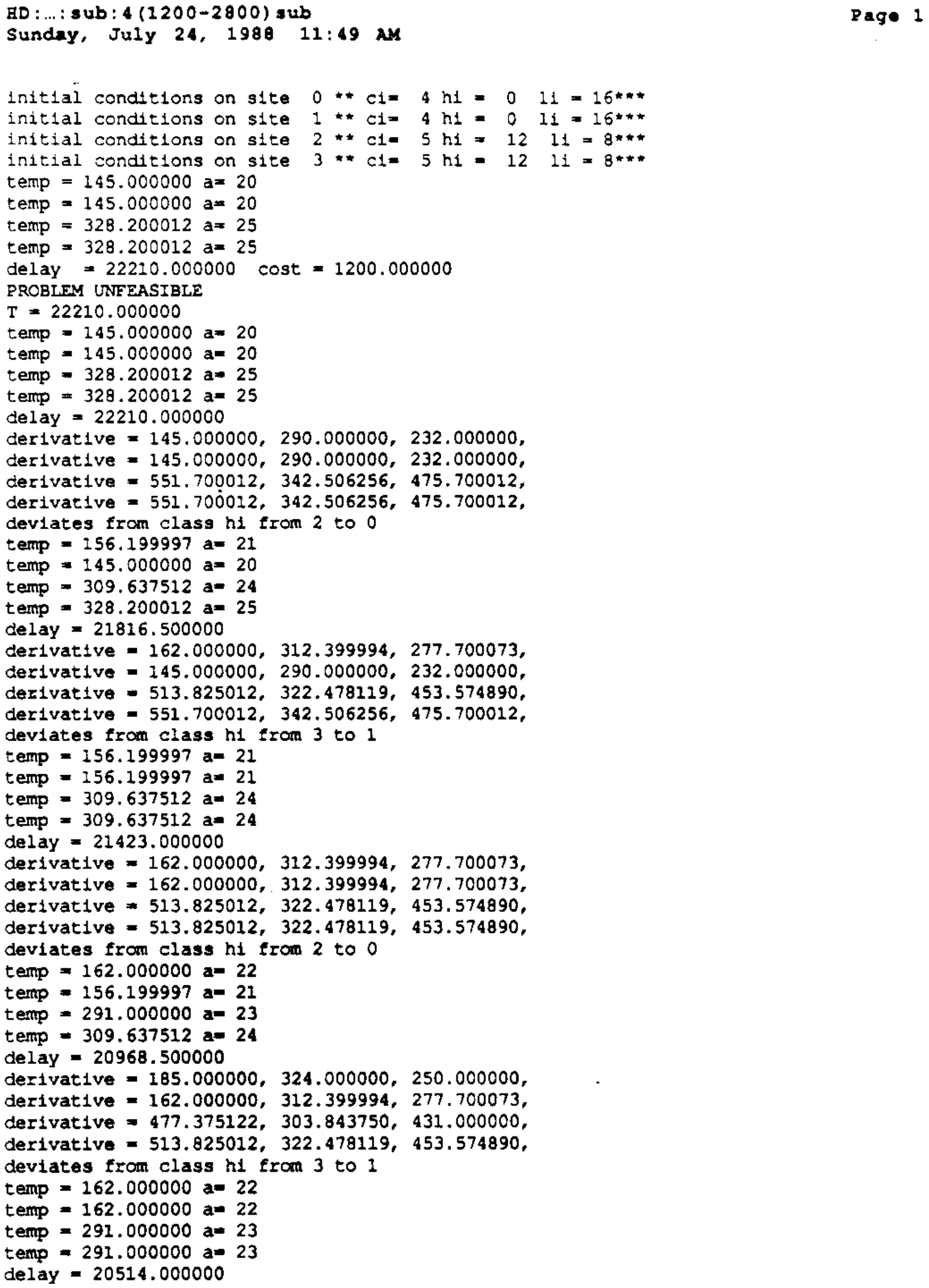
```
            } else if  (dtli[imax] >=
               (dthi[imax] > dtci[imax] ? dthi[imax] : dtci[imax]) ){
                printf("deviates from class li from %d to %d\n",imax,imin);
                fprintf(outfile,"deviates from class li from %d to %d\n",imax,imin);
                k = 2;
                nsesli[imax] -=deltli  ;
                nsesli[imin] +=deltli  ;
            }else  if   (dtci[imax] >=
               (dtli[imax] > dthi[imax] ? dtli[imax] : dthi[imax]) ){
                printf("deviates from class cifrom %d to %d\n",imax,imin);
                fprintf(outfile,"deviates from class cifrom %d to %d\n",imax,imin);

                k = 3;
                nsesci[imax] -=deltci  ;
                nsesci[imin] +=deltci  ;
            } else {
                printf("NO deviation ---- ERROR\n dci %f dhi %f dhi %f \n",dthi[imax],
                dtci[imax],dtli[imax]);
                fprintf(outfile,"NO deviation ---- ERROR\n dci %f dhi %f dhi %f \n",dt
                dtci[imax],dtli[imax]);
                exit();
            }
            a =  nseshi[imax] + nsesli[imax] + nsesci[imax] ;
           if (a == 0) backbone[imax] = -1;
     }
}

float ca()
{
 float tcost = 0.0;
 float ttotal,
        r,
        temp,
        rmax;
 int a,i,b,imax;



     /* compute minimum fit assignment and total cost */
     for (i = 0 ; i < links ; i++) {
        a =  nseshi[i] + nsesli[i] + nsesci[i] ;
        if ( a == 0 ) {
          printf("no backbone in link = %d \n",i);
          backbone[i] = -1;
        }
        else {
              printf("backbone assigned to link %d\n a = %d\n",i,a);
              backbone[i] = 0;
              tcost += cost[0];
        }
     }
     for ( ; ; ) {
        ttotal = rt();
     /*  printf("delay  = %f  cost = %f \n",ttotal,tcost); */
       fprintf(outfile,"delay  = %f  cost = %f \n",ttotal,tcost);

       if (tcost >= cmax) {
          printf("PROBLEM UNFEASIBLE\n");
          fprintf(outfile,"PROBLEM UNFEASIBLE\n");
          return ttotal;
       }

       if (ttotal < tmax) {
           printf("-----suboptimum capacity assignment is :\n");
           /* for ( i = 0 ; i < links ; i++) {
               printf("b[ %d ] = %d \n",i,backbone[i]);
               fprintf(outfile,"b[ %d ] = %d \n",i,backbone[i]);
           }*/
           return ttotal; /* problem feasible and capacity assignment  is  suboptimum
       }
       rmax = -1;
       imax = -1;
       for (i = 0 ; i < links ; i++) {
          if (  (backbone[i] == maxb) ) r = 0.0 ;
          else {
                 temp =  delaylink(nseshi[i],nsesli[i],nsesci[i] ,backbone[i]) ;
                  r = (delaylink((nseshi[i]+delthi),(nsesli[i]+deltli),(nsesci[i]+delt
                                    backbone(i]) - temp);
                  b = backbone[i];
                  r /= cost[b+1] - cost[b];
          }
        /*  printf("r = %f \n",r); */
         /* fprintf(outfile,"r = %f \n",r); */
          if  (r  > rmax)  {
              /*  printf("r  = %f rmax = %f \n",r,rmax); */
                imax = i;
                rmax = r;
           }
        }
        if ( ( imax == -1) || ( rmax == 0.0) )  {
         /*  printf("no improvement possible\n"); */
           for ( i = 0 ; i < links ; i++) {
               printf("b[ %d ] = %d \n",i,backbone[i]);
               fprintf(outfile,"b[ %d ] = %d \n",i,backbone[i]);
           }
           return ttotal;
        }
        tcost -= (cost[backbone[imax]]);
        backbone[imax]++;
        tcost += (cost[backbone[imax]]);
        printf("tcost = ------%f\n",tcost);
     }
}


main()
{
 int i,j;
 float t,tp;
```

```
   outfile = fopen("analout","w");
   /* Initialize delays [illegible] interpolation*/
   for    (i = 0; i < 36 ; i++) {
       delays0_2[i] = (delays0_1[i] + delays0_4[i]) / 2.5 ;
       delays0_3[i] = (delays0_2[i] + delays0_4[i]) / 2 ;
       delays0_6[i] = (delays0_8[i] + delays0_4[i]) / 2 ;
       delays0_5[i] = (delays0_6[i] + delays0_4[i]) / 2 ;
       delays0_7[i] = (delays0_6[i] + delays0_8[i]) / 2 ;

       delays4_2[i] = (delays4_1[i] + delays4_4[i]) / 2.5 ;
       delays4_3[i] = (delays4_2[i] + delays4_4[i]) / 2 ;
       delays4_6[i] = (delays4_8[i] + delays4_4[i]) / 2 ;
       delays4_5[i] = (delays4_6[i] + delays4_4[i]) / 2 ;
       delays4_7[i] = (delays4_6[i] + delays4_8[i]) / 2 ;

       delays8_2[i] = (delays8_1[i] + delays8_4[i]) / 2.5 ;
       delays8_3[i] = (delays8_2[i] + delays8_4[i]) / 2 ;
       delays8_6[i] = (delays8_8[i] + delays8_4[i]) / 2 ;
       delays8_5[i] = (delays8_6[i] + delays8_4[i]) / 2 ;
       delays8_7[i] = (delays8_6[i] + delays8_8[i]) / 2 ;

       delays16_2[i] = (delays16_1[i] + delays16_4[i]) / 2.5 ;
       delays16_3[i] = (delays16_2[i] + delays16_4[i]) / 2 ;
       delays16_6[i] = (delays16_8[i] + delays16_4[i]) / 2 ;
       delays16_5[i] = (delays16_6[i] + delays16_4[i]) / 2 ;
       delays16_7[i] = (delays16_6[i] + delays16_8[i]) / 2 ;


       delays12_1[i] = (delays8_1[i] +  delays16_1[i])  / 2;
       delays12_4[i] = (delays8_4[i] +  delays16_4[i])  / 2;
       delays12_8[i] = (delays8_8[i] +  delays16_8[i])  / 2;


       delays12_2[i] = (delays12_1[i] + delays12_4[i]) / 2.5 ;
       delays12_3[i] = (delays12_2[i] + delays12_4[i]) / 2 ;
       delays12_6[i] = (delays12_8[i] + delays12_4[i]) / 2 ;
       delays12_5[i] = (delays12_6[i] + delays12_4[i]) / 2 ;
       delays12_7[i] = (delays12_6[i] + delays12_8[i]) / 2 ;


   }
 /*  printf("BEGIN-----\n"); */

for (cmax = 7000; cmax >=3000;cmax-=400) {
 t = 32000;
 tp = 32767.0;
 for( ; ( (t > tmax) && (tp - t) > epslon)  ; ) {
    tp = t;
    for (j=0; j<links;j++) {
       printf("initial conditions on site  %d ** ci=  %d hi =  %d  li = %d***\n"
                      ,j,nsesci[j],nseshi[j],nsesli[j]);
       fprintf(outfile,"initial conditions on site  %d ** ci=  %d hi =  %d  li = %d**
                      ,j,nsesci[j],nseshi[j],nsesli[j]);

     }
```

```
      t = ca();
              /*  printf(" END ___ CA Phase\n--------------"); */
      fprintf(outfile,"T = %f\n",t);
      printf("T = %f\n",t);
      t = fd();
      printf(" END ___ fd Phase\n--------------");

      for( i = 0 ; i < links ; i++) {
              fprintf(outfile,"optimal design site %d (sehi=%d sesli=%d sesci=%d  ,b=%d
                      ,i,nseshi[i],nsesli[i],nsesci[i],backbone[i]);
                    /*  printf("optimal design site %d (sehi=%d sesli=%d sesci=%d  ,b=
                      ,i,nseshi[i],nsesli[i],nsesci[i] ,backbone[i]); */
       }
       printf("ANOTHER ONE\n");
       fprintf(outfile,"ANOTHER ONE\n");

    }
 }

   if ((tp - t) <= epslon)
     if (t > tmax) {
         printf("converges to local minimum -- greater than tmax\n");
        fprintf(outfile,"converges to local minimum -- greater than tmax\n");
     }
     else {
         printf("converges to local minimum -- less than tmax\n");
         fprintf(outfile,"converges to local minimum -- less than tmax\n");
     }



 }
```



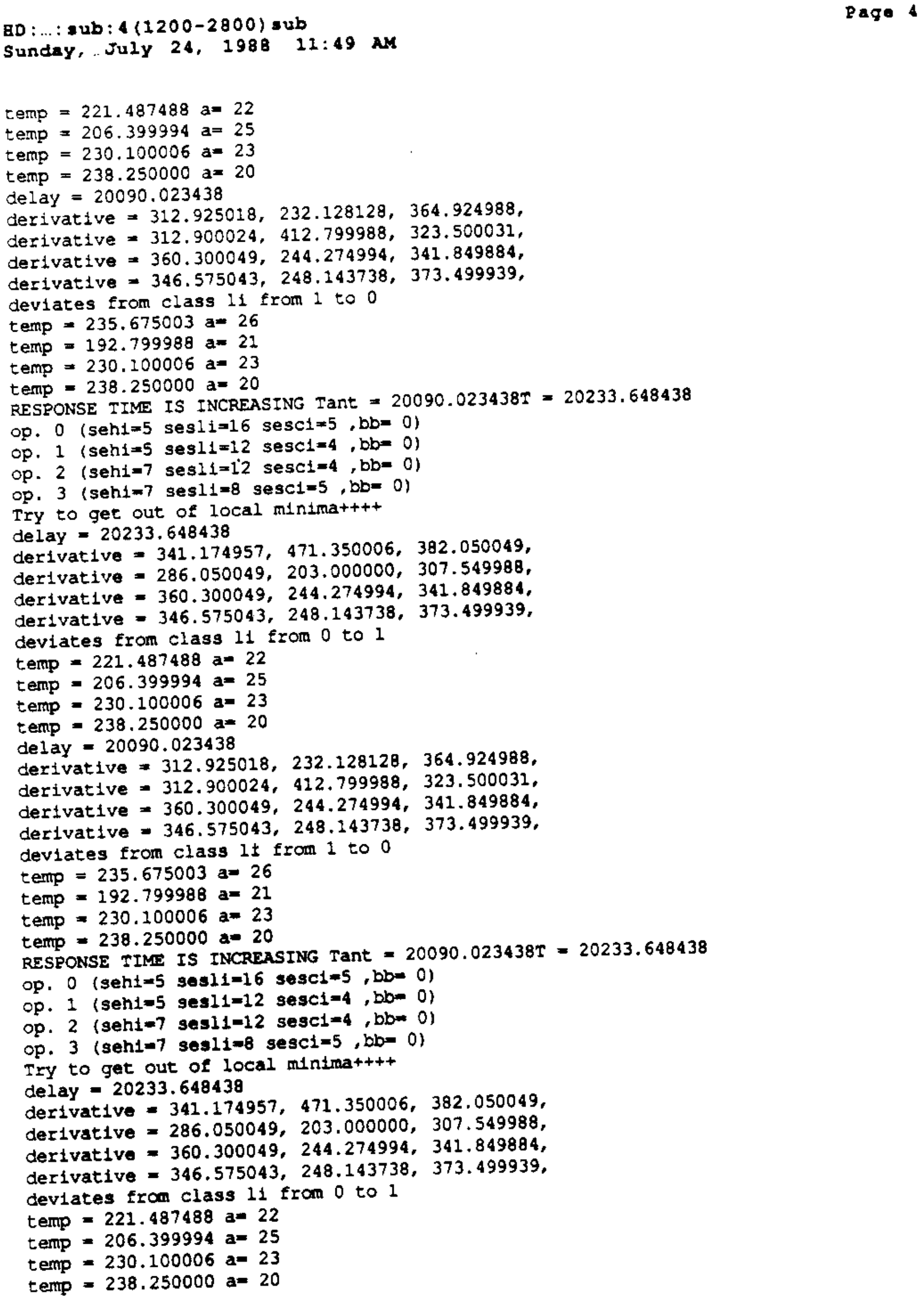
```
4(2800)uni                                                                    Page 1
Friday, July 22, 1988 8:27 PM

initial conditions on site  0 ** ci=  4 hi =  6  li = 12***
initial conditions on site  1 ** ci=  4 hi =  6  li = 12***
initial conditions on site  2 ** ci=  5 hi =  6  li = 12***
initial conditions on site  3 ** ci=  5 hi =  6  li = 12***
temp = 211.449997 a= 22
temp = 211.449997 a= 22
temp = 239.774994 a= 23
temp = 239.774994 a= 23
delay  = 20333.449219  cost = 1200.000000
temp = 211.449997 a= 22
temp = 211.449997 a= 22
temp = 140.037491 a= 23
temp = 239.774994 a= 23
delay  = 18039.486328  cost = 1600.000000
temp = 211.449997 a= 22
temp = 211.449997 a= 22
temp = 140.037491 a= 23
temp = 140.037491 a= 23
delay  = 15745.524414  cost = 2000.000000
temp = 130.649994 a= 22
temp = 211.449997 a= 22
temp = 140.037491 a= 23
temp = 140.037491 a= 23
delay  = 13967.923828  cost = 2400.000000
temp = 130.649994 a= 22
temp = 130.649994 a= 22
temp = 140.037491 a= 23
temp = 140.037491 a= 23
delay  = 12190.324219  cost = 2800.000000
PROBLEM UNFEASIBLE
T = 12190.324219
```



```
HD:…:sub:4(1200-2800)sub                                                       Page 1
Sunday, July 24, 1988  11:49 AM

initial conditions on site  0 ** ci=  4 hi =  0  li = 16***
initial conditions on site  1 ** ci=  4 hi =  0  li = 16***
initial conditions on site  2 ** ci=  5 hi =  12  li = 8***
initial conditions on site  3 ** ci=  5 hi =  12  li = 8***
temp = 145.000000 a= 20
temp = 145.000000 a= 20
temp = 328.200012 a= 25
temp = 328.200012 a= 25
delay  = 22210.000000  cost = 1200.000000
PROBLEM UNFEASIBLE
T = 22210.000000
temp = 145.000000 a= 20
temp = 145.000000 a= 20
temp = 328.200012 a= 25
temp = 328.200012 a= 25
delay = 22210.000000
derivative = 145.000000, 290.000000, 232.000000,
derivative = 145.000000, 290.000000, 232.000000,
derivative = 551.700012, 342.506256, 475.700012,
derivative = 551.700012, 342.506256, 475.700012,
deviates from class hi from 2 to 0
temp = 156.199997 a= 21
temp = 145.000000 a= 20
temp = 309.637512 a= 24
temp = 328.200012 a= 25
delay = 21816.500000
derivative = 162.000000, 312.399994, 277.700073,
derivative = 145.000000, 290.000000, 232.000000,
derivative = 513.825012, 322.478119, 453.574890,
derivative = 551.700012, 342.506256, 475.700012,
deviates from class hi from 3 to 1
temp = 156.199997 a= 21
temp = 156.199997 a= 21
temp = 309.637512 a= 24
temp = 309.637512 a= 24
delay = 21423.000000
derivative = 162.000000, 312.399994, 277.700073,
derivative = 162.000000, 312.399994, 277.700073,
derivative = 513.825012, 322.478119, 453.574890,
derivative = 513.825012, 322.478119, 453.574890,
deviates from class hi from 2 to 0
temp = 162.000000 a= 22
temp = 156.199997 a= 21
temp = 291.000000 a= 23
temp = 309.637512 a= 24
delay = 20968.500000
derivative = 185.000000, 324.000000, 250.000000,
derivative = 162.000000, 312.399994, 277.700073,
derivative = 477.375122, 303.843750, 431.000000,
derivative = 513.825012, 322.478119, 453.574890,
deviates from class hi from 3 to 1
temp = 162.000000 a= 22
temp = 162.000000 a= 22
temp = 291.000000 a= 23
temp = 291.000000 a= 23
delay = 20514.000000
```

```
derivative = 185.000000, 324.000000, 250.000000,
derivative = 185.000000, 324.000000, 250.000000,
derivative = 477.375122, 303.843750, 431.000000,
derivative = 477.375122, 303.843750, 431.000000,
deviates from class hi from 2 to 0
temp = 173.500000 a= 23
temp = 162.000000 a= 22
temp = 272.325012 a= 22
temp = 291.000000 a= 23
delay = 20238.650391
derivative = 208.300018, 347.000000, 276.500000,
derivative = 185.000000, 324.000000, 250.000000,
derivative = 440.399902, 284.493744, 408.449890,
derivative = 477.375122, 303.843750, 431.000000,
deviates from class hi from 3 to 1
temp = 173.500000 a= 23
temp = 173.500000 a= 23
temp = 272.325012 a= 22
temp = 272.325012 a= 22
delay = 19963.300781
derivative = 208.300018, 347.000000, 276.500000,
derivative = 208.300018, 347.000000, 276.500000,
derivative = 440.399902, 284.493744, 408.449890,
derivative = 440.399902, 284.493744, 408.449890,
deviates from class hi from 2 to 0
temp = 185.100006 a= 24
temp = 173.500000 a= 23
temp = 253.725006 a= 21
temp = 272.325012 a= 22
delay = 19752.275391
derivative = 270.299957, 370.200012, 303.000031,
derivative = 208.300018, 347.000000, 276.500000,
derivative = 402.525055, 264.993774, 386.349976,
derivative = 440.399902, 284.493744, 408.449890,
deviates from class hi from 3 to 1
temp = 185.100006 a= 24
temp = 185.100006 a= 24
temp = 253.725006 a= 21
temp = 253.725006 a= 21
delay = 19541.251953
derivative = 270.299957, 370.200012, 303.000031,
derivative = 270.299957, 370.200012, 303.000031,
derivative = 402.525055, 264.993774, 386.349976,
derivative = 402.525055, 264.993774, 386.349976,
deviates from class hi from 2 to 0
temp = 206.399994 a= 25
temp = 185.100006 a= 24
temp = 238.250000 a= 20
temp = 253.725006 a= 21
RESPONSE TIME IS INCREASING Tant = 19541.251953T = 19695.625000
op. 0 (sehi=5 sesli=16 sesci=4 ,bb= 0)
op. 1 (sehi=4 sesli=16 sesci=4 ,bb= 0)
op. 2 (sehi=7 sesli=8 sesci=5 ,bb= 0)
op. 3 (sehi=8 sesli=8 sesci=5 ,bb= 0)
Try to get out of local minima++++
delay = 19695.625000
```

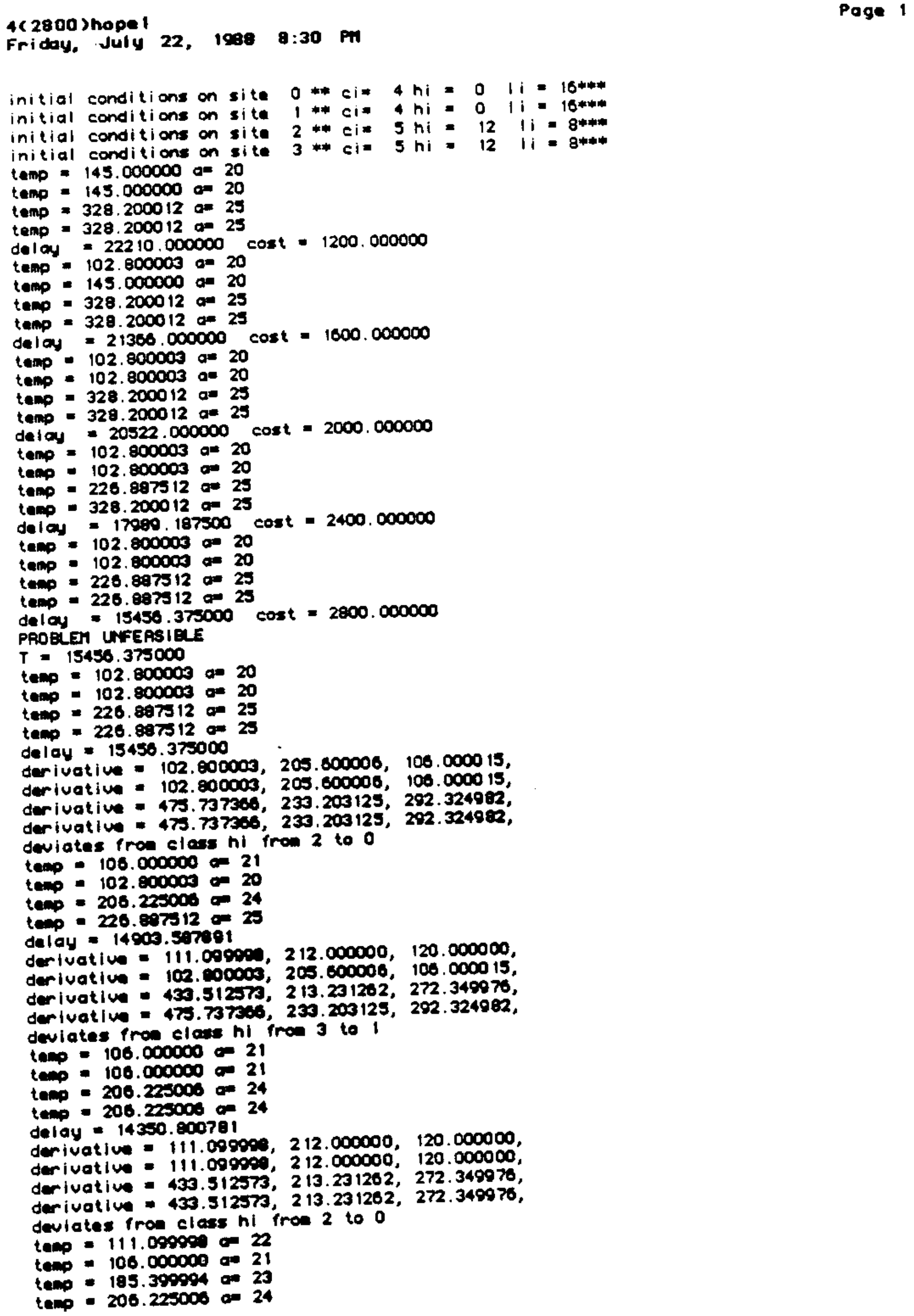
```
derivative = 312.900024, 412.799988, 323.500031,
derivative = 270.299957, 370.200012, 303.000031,
derivative = 346.575043, 248.143738, 373.499939,
derivative = 402.525055, 264.993774, 386.349976,
deviates from class hi from 3 to 1
temp = 206.399994 a= 25
temp = 206.399994 a= 25
temp = 238.250000 a= 20
temp = 238.250000 a= 20
RESPONSE TIME IS INCREASING Tant = 19695.625000T = 19850.000000
op. 0 (sehi=5 sesli=16 sesci=4 ,bb= 0)
op. 1 (sehi=5 sesli=16 sesci=4 ,bb= 0)
op. 2 (sehi=7 sesli=8 sesci=5 ,bb= 0)
op. 3 (sehi=7 sesli=8 sesci=5 ,bb= 0)
optimal design site 0 (sehi=5 sesli=16 sesci=4  ,b=0)
optimal design site 1 (sehi=5 sesli=16 sesci=4  ,b=0)
optimal design site 2 (sehi=7 sesli=8 sesci=5  ,b=0)
optimal design site 3 (sehi=7 sesli=8 sesci=5  ,b=0)
ANOTHER ONE
initial conditions on site  0 ** ci=  4 hi =  5  li = 16***
initial conditions on site  1 ** ci=  4 hi =  5  li = 16***
initial conditions on site  2 ** ci=  5 hi =  7  li = 8***
initial conditions on site  3 ** ci=  5 hi =  7  li = 8***
temp = 206.399994 a= 25
temp = 206.399994 a= 25
temp = 238.250000 a= 20
temp = 238.250000 a= 20
delay  = 19850.000000  cost = 1200.000000
PROBLEM UNFEASIBLE
T = 19850.000000
temp = 206.399994 a= 25
temp = 206.399994 a= 25
temp = 238.250000 a= 20
temp = 238.250000 a= 20
delay = 19850.000000
derivative = 312.900024, 412.799988, 323.500031,
derivative = 312.900024, 412.799988, 323.500031,
derivative = 346.575043, 248.143738, 373.499939,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class li from 0 to 2
temp = 192.799988 a= 21
temp = 206.399994 a= 25
temp = 258.037476 a= 24
temp = 238.250000 a= 20
RESPONSE TIME IS INCREASING Tant = 19850.000000T = 20166.699219
op. 0 (sehi=5 sesli=12 sesci=4 ,bb= 0)
op. 1 (sehi=5 sesli=16 sesci=4 ,bb= 0)
op. 2 (sehi=7 sesli=12 sesci=5 ,bb= 0)
op. 3 (sehi=7 sesli=8 sesci=5 ,bb= 0)
Try to get out of local minima++++
delay = 20166.699219
derivative = 286.050049, 203.000000, 307.549988,
derivative = 312.900024, 412.799988, 323.500031,
derivative = 385.612732, 272.878113, 397.724976,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class cifrom 2 to 0
```



Page 4

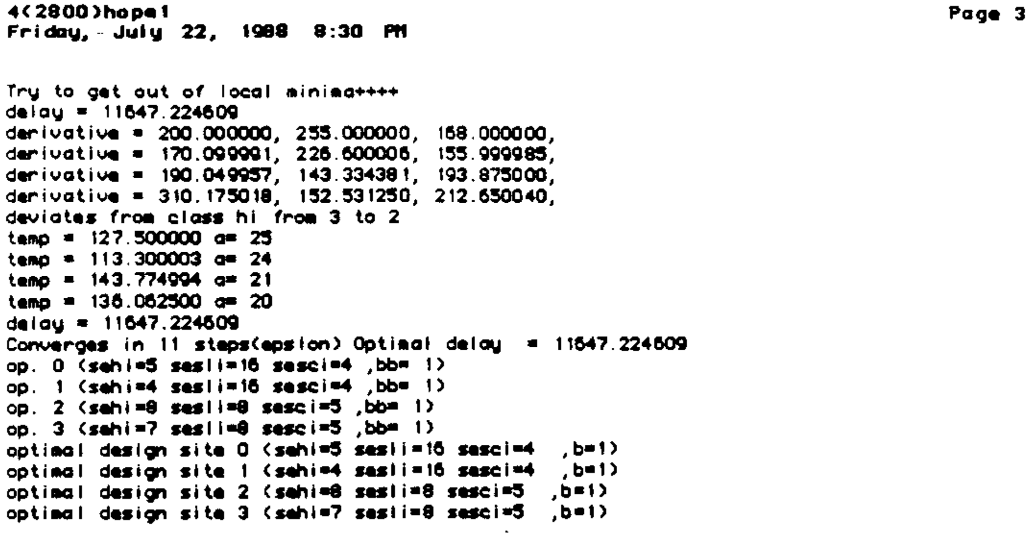
HD:...:sub:4(1200-2800)sub
Sunday, July 24, 1988 11:49 AM

```
temp = 221.487488 a= 22
temp = 206.399994 a= 25
temp = 230.100006 a= 23
temp = 238.250000 a= 20
delay = 20090.023438
derivative = 312.925018, 232.128128, 364.924988,
derivative = 312.900024, 412.799988, 323.500031,
derivative = 360.300049, 244.274994, 341.849884,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class li from 1 to 0
temp = 235.675003 a= 26
temp = 192.799988 a= 21
temp = 230.100006 a= 23
temp = 238.250000 a= 20
RESPONSE TIME IS INCREASING Tant = 20090.023438T = 20233.648438
op. 0 (sehi=5 sesli=16 sesci=5 ,bb= 0)
op. 1 (sehi=5 sesli=12 sesci=4 ,bb= 0)
op. 2 (sehi=7 sesli=12 sesci=4 ,bb= 0)
op. 3 (sehi=7 sesli=8 sesci=5 ,bb= 0)
Try to get out of local minima++++
delay = 20233.648438
derivative = 341.174957, 471.350006, 382.050049,
derivative = 286.050049, 203.000000, 307.549988,
derivative = 360.300049, 244.274994, 341.849884,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class li from 0 to 1
temp = 221.487488 a= 22
temp = 206.399994 a= 25
temp = 230.100006 a= 23
temp = 238.250000 a= 20
delay = 20090.023438
derivative = 312.925018, 232.128128, 364.924988,
derivative = 312.900024, 412.799988, 323.500031,
derivative = 360.300049, 244.274994, 341.849884,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class li from 1 to 0
temp = 235.675003 a= 26
temp = 192.799988 a= 21
temp = 230.100006 a= 23
temp = 238.250000 a= 20
RESPONSE TIME IS INCREASING Tant = 20090.023438T = 20233.648438
op. 0 (sehi=5 sesli=16 sesci=5 ,bb= 0)
op. 1 (sehi=5 sesli=12 sesci=4 ,bb= 0)
op. 2 (sehi=7 sesli=12 sesci=4 ,bb= 0)
op. 3 (sehi=7 sesli=8 sesci=5 ,bb= 0)
Try to get out of local minima++++
delay = 20233.648438
derivative = 341.174957, 471.350006, 382.050049,
derivative = 286.050049, 203.000000, 307.549988,
derivative = 360.300049, 244.274994, 341.849884,
derivative = 346.575043, 248.143738, 373.499939,
deviates from class li from 0 to 1
temp = 221.487488 a= 22
temp = 206.399994 a= 25
temp = 230.100006 a= 23
temp = 238.250000 a= 20
```



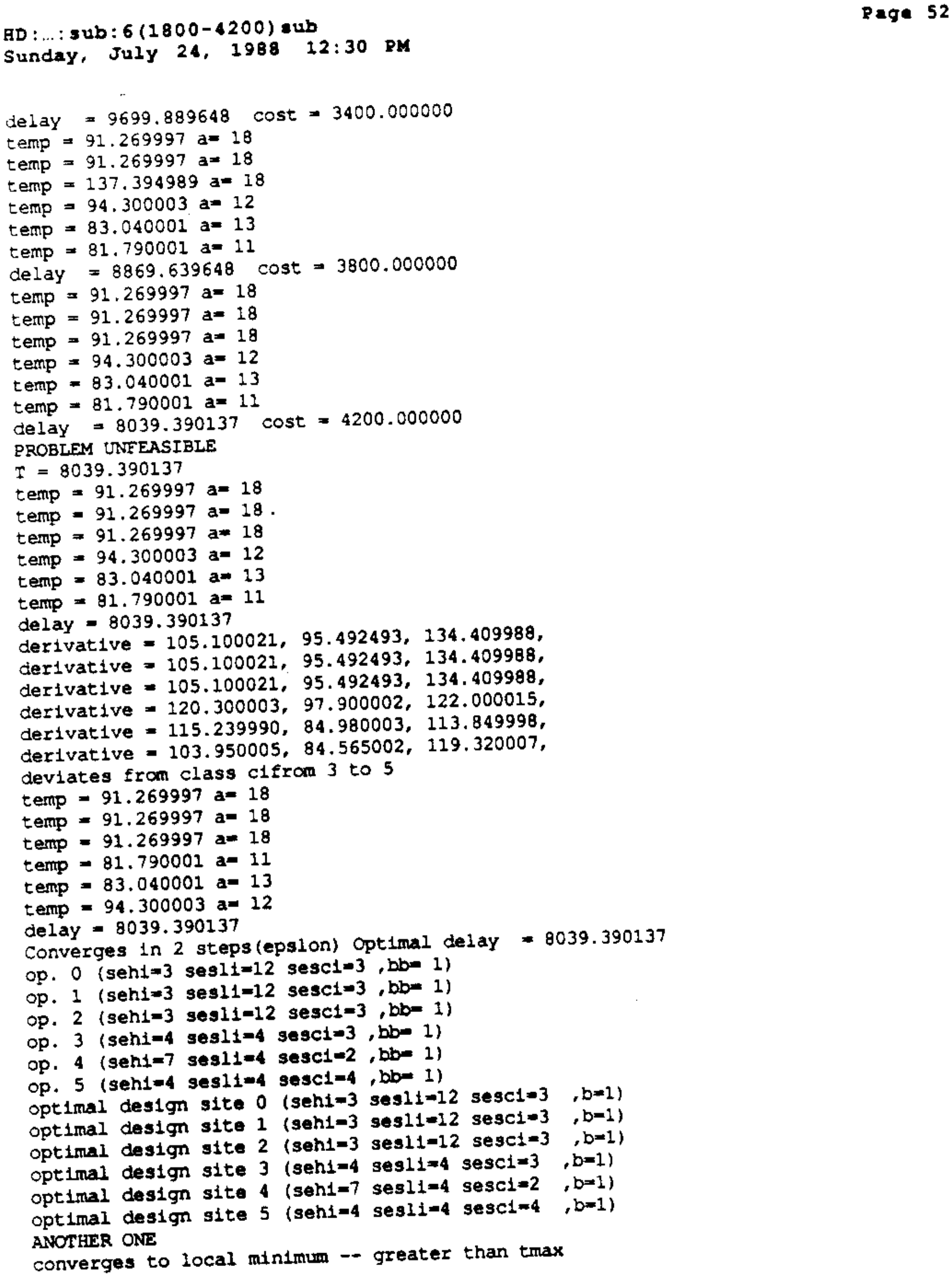
```
4(2800)hopel                                                       Page 1
Friday, July 22, 1988  8:30  PM

initial conditions on site  0 ** ci=  4 hi =  0  li = 16***
initial conditions on site  1 ** ci=  4 hi =  0  li = 16***
initial conditions on site  2 ** ci=  5 hi =  12  li = 8***
initial conditions on site  3 ** ci=  5 hi =  12  li = 8***
temp = 145.000000 a= 20
temp = 145.000000 a= 20
temp = 328.200012 a= 25
temp = 328.200012 a= 25
delay  = 22210.000000  cost = 1200.000000
temp = 102.800003 a= 20
temp = 145.000000 a= 20
temp = 328.200012 a= 25
temp = 328.200012 a= 25
delay  = 21366.000000  cost = 1600.000000
temp = 102.800003 a= 20
temp = 102.800003 a= 20
temp = 328.200012 a= 25
temp = 328.200012 a= 25
delay  = 20522.000000  cost = 2000.000000
temp = 102.800003 a= 20
temp = 102.800003 a= 20
temp = 226.887512 a= 25
temp = 328.200012 a= 25
delay  = 17989.187500  cost = 2400.000000
temp = 102.800003 a= 20
temp = 102.800003 a= 20
temp = 226.887512 a= 25
temp = 226.887512 a= 25
delay  = 15456.375000  cost = 2800.000000
PROBLEM UNFEASIBLE
T = 15456.375000
temp = 102.800003 a= 20
temp = 102.800003 a= 20
temp = 226.887512 a= 25
temp = 226.887512 a= 25
delay = 15456.375000
derivative = 102.800003, 205.600006, 106.000015,
derivative = 102.800003, 205.600006, 106.000015,
derivative = 475.737366, 233.203125, 292.324982,
derivative = 475.737366, 233.203125, 292.324982,
deviates from class hi  from 2 to 0
temp = 106.000000 a= 21
temp = 102.800003 a= 20
temp = 206.225006 a= 24
temp = 226.887512 a= 25
delay = 14903.587891
derivative = 111.099998, 212.000000, 120.000000,
derivative = 102.800003, 205.600006, 106.000015,
derivative = 433.512573, 213.231262, 272.349976,
derivative = 475.737366, 233.203125, 292.324982,
deviates from class hi  from 3 to 1
temp = 106.000000 a= 21
temp = 106.000000 a= 21
temp = 206.225006 a= 24
temp = 206.225006 a= 24
delay = 14350.800781
derivative = 111.099998, 212.000000, 120.000000,
derivative = 111.099998, 212.000000, 120.000000,
derivative = 433.512573, 213.231262, 272.349976,
derivative = 433.512573, 213.231262, 272.349976,
deviates from class hi  from 2 to 0
temp = 111.099998 a= 22
temp = 106.000000 a= 21
temp = 185.399994 a= 23
temp = 206.225006 a= 24
```

```
delay = 13883.800781
derivative = 113.299995, 222.199997, 133.299988,
derivative = 111.099998, 212.000000, 120.000000,
derivative = 393.650116, 192.987488, 252.400040,
derivative = 433.512573, 213.231262, 272.349976,
deviates from class hi from 3 to 1
temp = 111.099998 a= 22
temp = 111.099998 a= 22
temp = 185.399994 a= 23
temp = 185.399994 a= 23
delay = 13416.799805
derivative = 113.299995, 222.199997, 133.299988,
derivative = 113.299995, 222.199997, 133.299988,
derivative = 393.650116, 192.987488, 252.400040,
derivative = 393.650116, 192.987488, 252.400040,
deviates from class hi from 2 to 0
temp = 112.199997 a= 23
temp = 111.099998 a= 22
temp = 164.574997 a= 22
temp = 185.399994 a= 23
delay = 12909.650391
derivative = 115.500015, 224.399994, 145.000015,
derivative = 113.299995, 222.199997, 133.299988,
derivative = 351.999969, 172.781250, 232.449982,
derivative = 393.650116, 192.987488, 252,400040,
deviates from class hi from 3 to 1
temp = 112.199997 a= 23
temp = 112.199997 a= 23
temp = 164.574997 a= 22
temp = 164.574997 a= 22
delay = 12402.500000
derivative = 115.500015, 224.399994, 145.000015,
derivative = 115.500015, 224.399994, 145.000015,
derivative = 351.999969, 172.781250, 232.449982,
derivative = 351.999969, 172.781250, 232.449982,
deviates from class hi from 2 to 0
temp = 113.300003 a= 24
temp = 112.199997 a= 23
temp = 143.774994 a= 21
temp = 164.574997 a= 22
delay = 11939.725586
derivative = 170.099991, 226.600006, 155.999985,
derivative = 115.500015, 224.399994, 145.000015,
derivative = 310.175018, 152.531250, 212.650040,
derivative = 351.999969, 172.781250, 232.449982,
deviates from class hi from 3 to 1
temp = 113.300003 a= 24
temp = 113.300003 a= 24
temp = 143.774994 a= 21
temp = 143.774994 a= 21
delay = 11476.949219
derivative = 170.099991, 226.600006, 155.999985,
derivative = 170.099991, 226.600006, 155.999985,
derivative = 310.175018, 152.531250, 212.650040,
derivative = 310.175018, 152.531250, 212.650040,
deviates from class hi from 2 to 0
temp = 127.500000 a= 25
temp = 113.300003 a= 24
temp = 136.062500 a= 20
temp = 143.774994 a= 21
RESPONSE TIME IS INCREASING Tant = 11476.949219T = 11647.224609
op. 0 (sehi=5 sesli=16 sesci=4 ,bb= 1)
op. 1 (sehi=4 sesli=16 sesci=4 ,bb= 1)
op. 2 (sehi=7 sesli=8 sesci=5 ,bb= 1)
op. 3 (sehi=8 sesli=8 sesci=5 ,bb= 1)
```



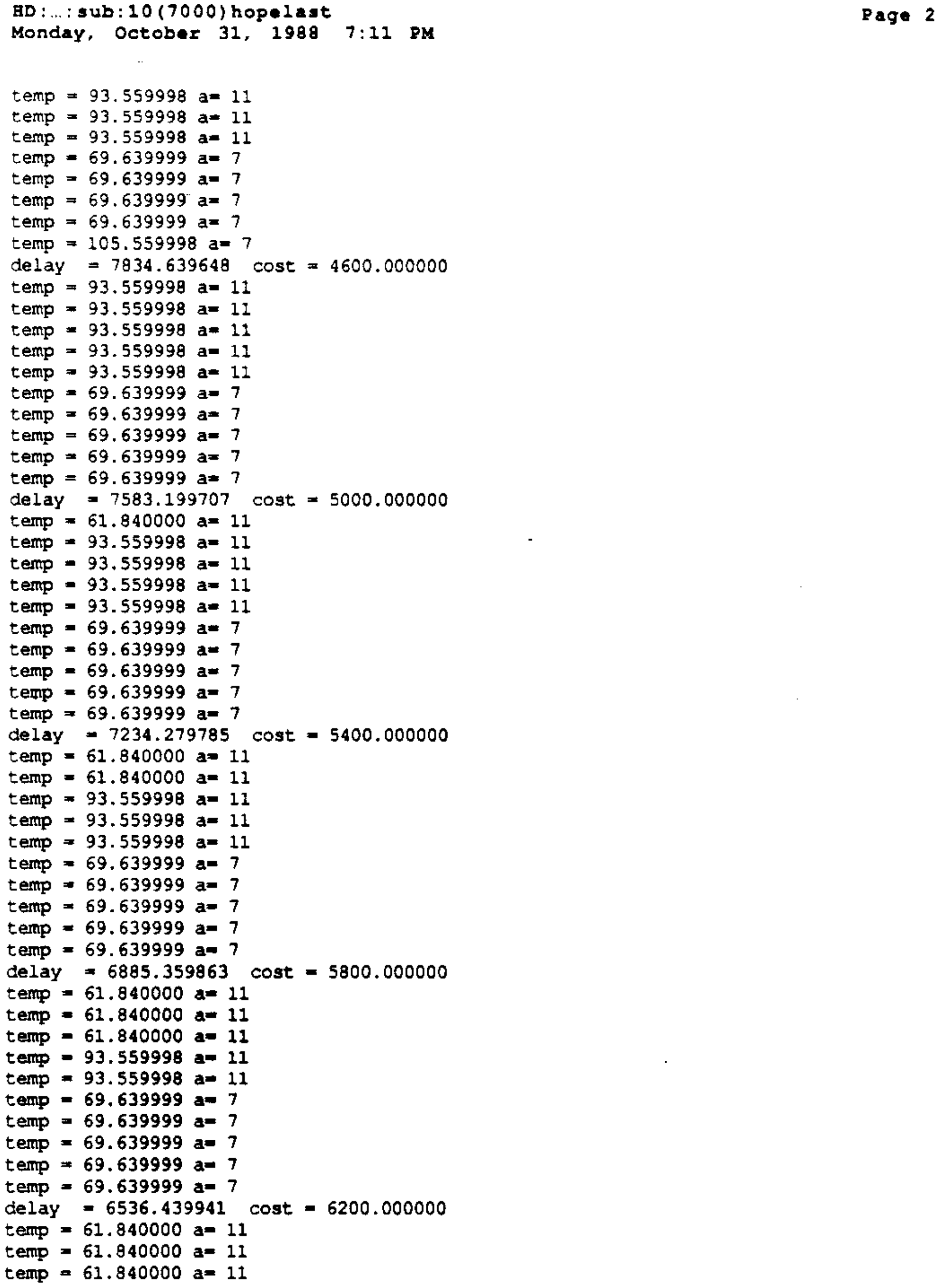
```
4(2800)hope1                                                          Page 3
Friday, July 22,  1988  8:30  PM


Try to get out of local minima++++
delay = 11647.224609
derivative = 200.000000, 255.000000, 168.000000,
derivative =  170.099991, 226.600006, 155.999985,
derivative =  190.049957, 143.334381, 193.875000,
derivative = 310.175018, 152.531250, 212.650040,
deviates from class hi from 3 to 2
temp = 127.500000 a= 25
temp = 113.300003 a= 24
temp = 143.774994 a= 21
temp = 136.062500 a= 20
delay = 11647.224609
Converges in 11 steps(epsion) Optimal delay  = 11647.224609
op. 0 (sehi=5 sesli=16 sesci=4 ,bb= 1)
op. 1 (sehi=4 sesli=16 sesci=4 ,bb= 1)
op. 2 (sehi=8 sesli=8 sesci=5 ,bb= 1)
op. 3 (sehi=7 sesli=8 sesci=5 ,bb= 1)
optimal design site 0 (sehi=5 sesli=16 sesci=4  ,b=1)
optimal design site 1 (sehi=4 sesli=16 sesci=4  ,b=1)
optimal design site 2 (sehi=8 sesli=8 sesci=5  ,b=1)
optimal design site 3 (sehi=7 sesli=8 sesci=5  ,b=1)
```

```
temp = 81.790001 a= 11
delay = 8039.390137
Converges in 4 steps(epslon) Optimal delay  = 8039.390137
op. 0 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 1 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 2 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 3 (sehi=4 sesli=4 sesci=4 ,bb= 1)
op. 4 (sehi=7 sesli=4 sesci=2 ,bb= 1)
op. 5 (sehi=4 sesli=4 sesci=3 ,bb= 1)
optimal design site 0 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 1 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 2 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 3 (sehi=4 sesli=4 sesci=4  ,b=1)
optimal design site 4 (sehi=7 sesli=4 sesci=2  ,b=1)
optimal design site 5 (sehi=4 sesli=4 sesci=3  ,b=1)
ANOTHER ONE
initial conditions on site  0 ** ci=  3 hi =  3  li = 12***
initial conditions on site  1 ** ci=  3 hi =  3  li = 12***
initial conditions on site  2 ** ci=  3 hi =  3  li = 12***
initial conditions on site  3 ** ci=  4 hi =  4  li = 4***
initial conditions on site  4 ** ci=  2 hi =  7  li = 4***
initial conditions on site  5 ** ci=  3 hi =  4  li = 4***
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 152.000000 a= 12
temp = 131.199997 a= 13
temp = 127.020004 a= 11
delay  = 12346.149414  cost = 1800.000000
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 152.000000 a= 12
temp = 83.040001 a= 13
temp = 127.020004 a= 11
delay  = 11720.069336  cost = 2200.000000
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 127.020004 a= 11
delay  = 11027.668945  cost = 2600.000000
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 81.790001 a= 11
delay  = 10530.139648  cost = 3000.000000
temp = 91.269997 a= 18
temp = 137.394989 a= 18
temp = 137.394989 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 81.790001 a= 11
```

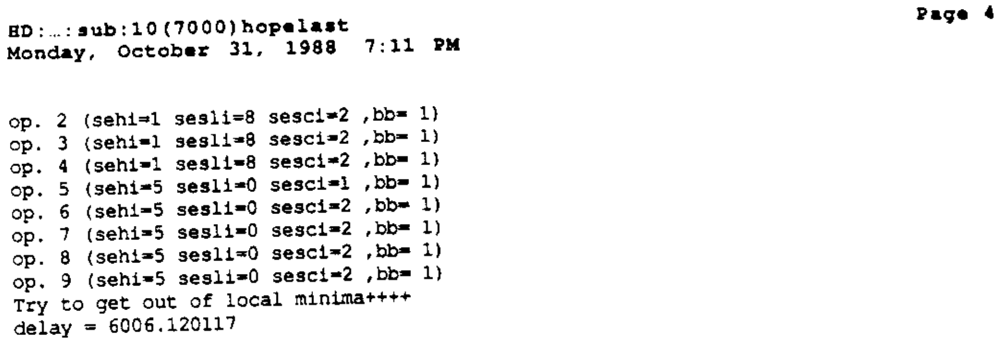


```
delay  = 9699.889648  cost = 3400.000000
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 137.394989 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 81.790001 a= 11
delay  = 8869.639648  cost = 3800.000000
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 81.790001 a= 11
delay  = 8039.390137  cost = 4200.000000
PROBLEM UNFEASIBLE
T = 8039.390137
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 94.300003 a= 12
temp = 83.040001 a= 13
temp = 81.790001 a= 11
delay = 8039.390137
derivative = 105.100021, 95.492493, 134.409988,
derivative = 105.100021, 95.492493, 134.409988,
derivative = 105.100021, 95.492493, 134.409988,
derivative = 120.300003, 97.900002, 122.000015,
derivative = 115.239990, 84.980003, 113.849998,
derivative = 103.950005, 84.565002, 119.320007,
deviates from class cifrom 3 to 5
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 91.269997 a= 18
temp = 81.790001 a= 11
temp = 83.040001 a= 13
temp = 94.300003 a= 12
delay = 8039.390137
Converges in 2 steps(epslon) Optimal delay  = 8039.390137
op. 0 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 1 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 2 (sehi=3 sesli=12 sesci=3 ,bb= 1)
op. 3 (sehi=4 sesli=4 sesci=3 ,bb= 1)
op. 4 (sehi=7 sesli=4 sesci=2 ,bb= 1)
op. 5 (sehi=4 sesli=4 sesci=4 ,bb= 1)
optimal design site 0 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 1 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 2 (sehi=3 sesli=12 sesci=3  ,b=1)
optimal design site 3 (sehi=4 sesli=4 sesci=3  ,b=1)
optimal design site 4 (sehi=7 sesli=4 sesci=2  ,b=1)
optimal design site 5 (sehi=4 sesli=4 sesci=4  ,b=1)
ANOTHER ONE
converges to local minimum -- greater than tmax
```

```
HD:…:sub:10(7000)hopelast                                                  Page 1
Monday, October 31, 1988  7:11 PM


initial conditions on site  0 ** ci=  2 hi =  1  li = 8***
initial conditions on site  1 ** ci=  2 hi =  1  li = 8***
initial conditions on site  2 ** ci=  2 hi =  1  li = 8***
initial conditions on site  3 ** ci=  2 hi =  1  li = 8***
initial conditions on site  4 ** ci=  2 hi =  1  li = 8***
initial conditions on site  5 ** ci=  2 hi =  5  li = 0***
initial conditions on site  6 ** ci=  2 hi =  5  li = 0***
initial conditions on site  7 ** ci=  2 hi =  5  li = 0***
initial conditions on site  8 ** ci=  2 hi =  5  li = 0***
initial conditions on site  9 ** ci=  2 hi =  5  li = 0***
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
delay  = 8840.399414  cost = 3000.000000
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
delay  = 8588.959961  cost = 3400.000000
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
delay  = 8337.519531  cost = 3800.000000
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 105.559998 a= 7
temp = 105.559998 a= 7
delay  = 8086.079590  cost = 4200.000000
temp = 93.559998 a= 11
temp = 93.559998 a= 11
```



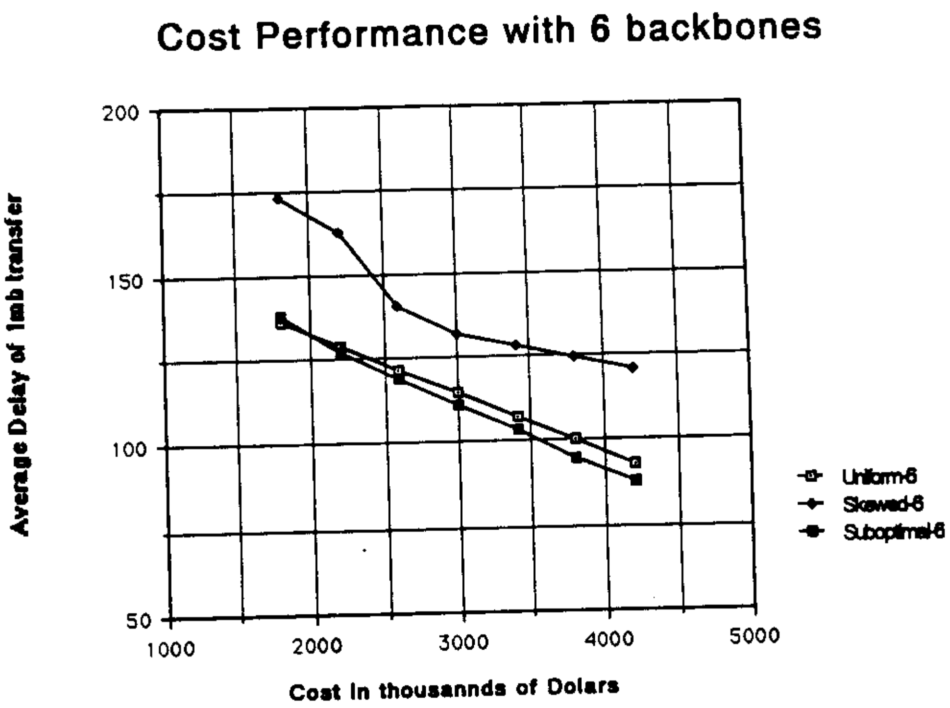
```
HD:…:sub:10(7000)hopelast                                         Page 2
Monday, October 31, 1988  7:11 PM

temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 105.559998 a= 7
delay  = 7834.639648  cost = 4600.000000
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 7583.199707  cost = 5000.000000
temp = 61.840000 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 7234.279785  cost = 5400.000000
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 6885.359863  cost = 5800.000000
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 93.559998 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 6536.439941  cost = 6200.000000
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 93.559998 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 6187.520020  cost = 6600.000000
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay  = 5838.600098  cost = 7000.000000
PROBLEM UNFEASIBLE
T = 5838.600098
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
delay = 5838.600098
derivative = 67.320000, 65.779999, 82.000000,
derivative = 67.320000, 65.779999, 82.000000,
derivative = 67.320000, 65.779999, 82.000000,
derivative = 67.320000, 65.779999, 82.000000,
derivative = 67.320000, 65.779999, 82.000000,
derivative = 94.840004, 69.639999, 95.400009,
derivative = 94.840004, 69.639999, 95.400009,
derivative = 94.840004, 69.639999, 95.400009,
derivative = 94.840004, 69.639999, 95.400009,
derivative = 94.840004, 69.639999, 95.400009,
deviates from class cifrom 5 to 0
temp = 71.919998 a= 12
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 61.840000 a= 11
temp = 78.699997 a= 6
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
temp = 69.639999 a= 7
RESPONSE TIME IS INCREASING Tant = 5838.600098T = 6006.120117
op. 0 (sehi=1 sesli=8 sesci=3 ,bb= 1)
op. 1 (sehi=1 sesli=8 sesci=2 ,bb= 1)
```



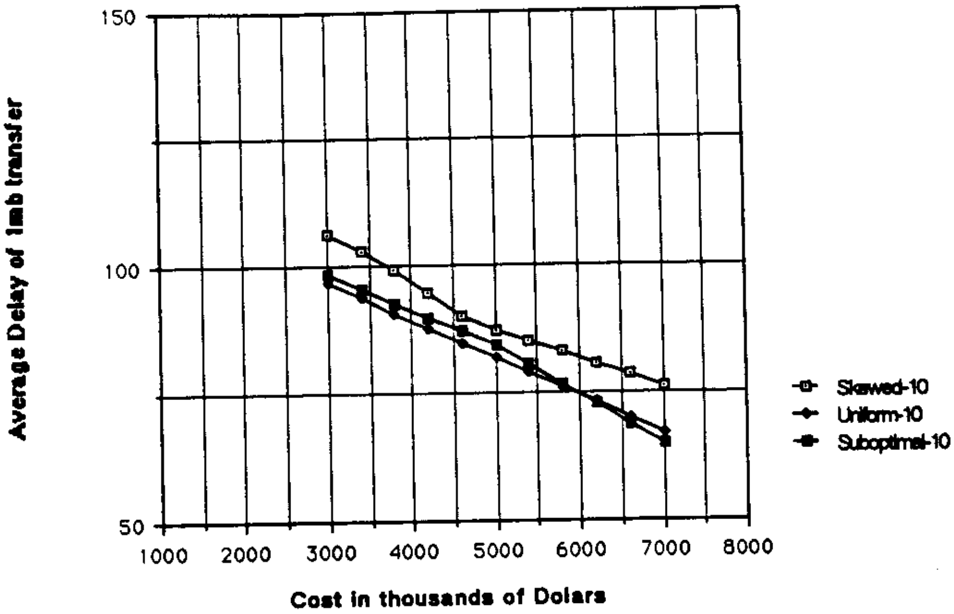
HD:…:sub:10(7000)hopelast Page 4

Monday, October 31, 1988 7:11 PM

```
op. 2 (sehi=1 sesli=8 sesci=2 ,bb= 1)
op. 3 (sehi=1 sesli=8 sesci=2 ,bb= 1)
op. 4 (sehi=1 sesli=8 sesci=2 ,bb= 1)
op. 5 (sehi=5 sesli=0 sesci=1 ,bb= 1)
op. 6 (sehi=5 sesli=0 sesci=2 ,bb= 1)
op. 7 (sehi=5 sesli=0 sesci=2 ,bb= 1)
op. 8 (sehi=5 sesli=0 sesci=2 ,bb= 1)
op. 9 (sehi=5 sesli=0 sesci=2 ,bb= 1)
Try to get out of local minima++++
delay = 6006.120117
```


Cost Performance with 6 backbones
200
150
100
50
1000
2000
3000
4000
5000
Average Delay of 1mb transfer
Cost In thousannds of Dolars
Uniform-6
Skewed-6
Suboptimal-6

Cost Performance with 10 backbones



APPENDIX G

User Model Derivation Data

We present here additional information that we have obtained in our user model investigation. This information pertains to courses tought, instructor interviews, and reduced network communication data.

88/05/10
10:35:08

courselist

```
----------------------------------------------------------
Nelson               CE165B          Fall   1985
Selna                CE167A          Fall   1985
Neethling            CE184D          Fall   1985 *
Nelson               CE265A          Fall   1985
Dyer                 CS163           Fall   1985
Catton               MA131A          Fall   1985
Karagozian           MA150P          Fall   1985 *
Yang                 MA162A          Fall   1985
Mingori              MA171A          Fall   1985
Fleury               MA197F          Fall   1985
Mal                  MA257A          Fall   1985
Gibson               MA271A          Fall   1985
----------------------------------------------------------
Neethling            CE184B          Winter 1986 *
Felio                CE185B          Winter 1986
Stenstrom            CE284H          Winter 1986
Selna                CE286A          Winter 1986
Manousiouthakis      CH138           Winter 1986
Kay/Wong(TA)         CS12            Winter 1986 *
Korf                 CS161           Winter 1986
Vidal                CS174           Winter 1986
Flowers              CS264A          Winter 1986 *
Klinger              CS267           Winter 1986
Wang                 EE122A          Winter 1986
Jacobsen             EE129A          Winter 1986 ^
Karagozian           MA150A          Winter 1986 *
Dinyavari            MA154A          Winter 1986
Mingori              MA164           Winter 1986
Gustafson            MA171A          Winter 1986
Orkent               MA236C          Winter 1986
Mal                  MA257B          Winter 1986
----------------------------------------------------------
Cohen                CE137F          Spring 1986
Felton/Coelho(TA)    CE165L          Spring 1986 *
Hart                 CE269           Spring 1986
Kay/Wong(TA)         CS13            Spring 1986 *
Lang                 CS151C          Spring 1986
Vidal                CS174           Spring 1986
Mak                  CS196B          Spring 1986
Cardenas             CS241B          Spring 1986
Luhmann              EE117M          Spring 1986
Hassul               EE122A          Spring 1986
Levine               MA131A          Spring 1986
Karagozian           MA250C          Spring 1986 *
----------------------------------------------------------
Neethling            CE184D          Fall   1986 *
Neethling            CE198           Fall   1986 ^
Fourney              CEM169L         Fall   1986
Cardenas             CS141           Fall   1986
Gerla                CS141           Fall   1986
Distefano            CS270C          Fall   1986
Jacobsen             EE103           Fall   1986 *
Martin               EE115A          Fall   1986
Mills                MA131A          Fall   1986
Wilcox               MA150A          Fall   1986
Karagozian           MA150P          Fall   1986 *
Yang                 MA162A          Fall   1986
Mingori              MA171A          Fall   1986
----------------------------------------------------------
Pfeiffer             CE106           Winter 1987
Felton               CE165B          Winter 1987 ^
Selna                CE167B          Winter 1987
Hart                 CE186           Winter 1987
```

88/05/10
10:35:08

courselist

```
Neethling            CE198           Winter 1987 ^
Nelson               CE265B          Winter 1987
Stenstrom            CE284H          Winter 1987
Mal                  CE286A          Winter 1987
Resnick              CH137E          Winter 1987
Manousiouthakis      CH138           Winter 1987
Kay/Wong(TA)         CS12            Winter 1987 *
Martin               CS132           Winter 1987
Inselberg            CS141           Winter 1987
Dechter              CS161           Winter 1987
Dyer                 CS163           Winter 1987
Vidal                CS174           Winter 1987
Cardenas             CS241AL         Winter 1987
Flowers              CS264A          Winter 1987 *
Jacobsen             EE103           Winter 1987 *
Giardini             EE136           Winter 1987
Elliot               EE141           Winter 1987
Karagozian           MA150A          Winter 1987 *
Dinyavari            MA154A          Winter 1987
Gustafson            MA171A          Winter 1987
Rizzo                MT141           Winter 1987
----------------------------------------------------------
Pfeiffer             CE106A          Spring 1987
Dong                 CE165C          Spring 1987
Felton/Coelho(TA)    CE165L          Spring 1987 *
Selna                CE167L          Spring 1987
Neethling            CE184F          Spring 1987 *
Neethling            CE198           Spring 1987 ^
Dong                 CE264B          Spring 1987
Selna                CE267C          Spring 1987
Perrine              CE281           Spring 1987
Hart                 CE286B          Spring 1987
Cohen                CH137F          Spring 1987
Kay/Wong(TA)         CS13            Spring 1987 *
Lang                 CS151C          Spring 1987
Korf                 CS161           Spring 1987
Flowers              CS264B          Spring 1987 *
Flowers              CS265           Spring 1987 *
Gurlitz              EE103           Spring 1987
Wiberg               EE141           Spring 1987
Levan                EE199           Spring 1987
Jacobsen             EE239BB         Spring 1987 ^
Mortensen            EE241C          Spring 1987 *
Rosenstein           EM176A          Spring 1987
Levine               MA131A          Spring 1987
Mills                MA131A          Spring 1987
Miu                  MA162L          Spring 1987
Mingori              MA164           Spring 1987
Karagosian           MA250C          Spring 1987 *
----------------------------------------------------------
Neethling            CE15A           Fall   1987 ^
Pfeiffer             CE106A          Fall   1987
Felio                CE120           Fall   1987
Fourney              CE137L          Fall   1987
Neethling            CE155           Fall   1987 *
Felton               CE180           Fall   1987 ^
Felio                CE223           Fall   1987
Dong                 CE237A          Fall   1987
Felton               CE240           Fall   1987 ^
Neethling            CE254J          Fall   1987 ^
Bagrodia             CS131           Fall   1987
Gerla                CS141           Fall   1987
Rennels              CS152A          Fall   1987 %
Cardenas             CS241AL         Fall   1987
```

88/05/10
10:35:08

courselist

Jacobsen	EE103	Fall	1987	^
Martin	EE115A	Fall	1987	
Jacobsen	EE236A	Fall	1987	^
Mortensen	EE241A	Fall	1987	*
Chir	MA105D	Fall	1987	
Stephenson	MA133A	Fall	1987	
Karagozian	MA150A	Fall	1987	*
Meyer	MA150P	Fall	1987	
Miu	MA162A	Fall	1987	
Forouhar/Youn(TA)	MA171A	Fall	1987	*
Wagner	MSE145A	Fall	1987	
Neethling	CE15A	Winter	1988	^
Neethling	CE15B	Winter	1988	^
Pfeiffer	CE106A	Winter	1988	
Fourney	CE130F	Winter	1988	
Selna	CE142	Winter	1988	
Felton	CE180	Winter	1988	^
Felio	CE221	Winter	1988	
Dong	CE235B	Winter	1988	
Felton	CE240	Winter	1988	^
Dracup	CE250A	Winter	1988	
Dracup	CE250B	Winter	1988	
Stenstrom	CE253	Winter	1988	
Neethling	CE255A	Winter	1988	^
Manousiouthakis	CH138	Winter	1988	§
Manousiouthakis	CH239AI	Winter	1988	
Kay/Wong(TA)	CS12	Winter	1988	*
Jefferson	CS111	Winter	1988	§
Bargodia	CS131	Winter	1988	
Cardenas	CS141	Winter	1988	
Rennels	CS152A	Winter	1988	
Klinger	CS161	Winter	1988	
Dyer	CS163	Winter	1988	
Aoki	CS170	Winter	1988	
Parker	CS239	Winter	1988	
Tyree	CS258BC	Winter	1988	
Flowers	CS264A	Winter	1988	*
Hillestad	EE103	Winter	1988	
Jacobsen	EE136	Winter	1988	^
Jacobsen	EE236B	Winter	1988	^
Wilcox	MA150A	Winter	1988	
Dinavari	MA154A	Winter	1988	
Yam/Youn(TA)	MA171A	Winter	1988	*
Friedmann	MA269B	Winter	1988	

* - Instructor or TA was interviewed w.r.t. this course.
^ - Instructor was interviewed but not w.r.t. this course.
§ - SEASnet accounts were not used for this course.
    The PC's wrer used as stand alone devices.

SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor Lewis Felton
Department:	Civil Engineering
Interviewer:	Geri Segal
Date:	February 17, 1988

Course:	CE165L (now 135L) - Structural Design and Testing Laboratory
Quarter:	Spring '87
No. of Students:	14-16
Hrs/wk in Lab:	5-10

Comments:

1. Now teaching CE 180
2. Teaches 135L every Spring Quarter.
3. Students used an optimization program called COPES provided by the professor. (Interviewer was provided with a copy of handout given to students to get them started using COPES and the PCs.)
4. Program was stored on the SEASnet class directory.
5. Students were required to write Fortran subroutines to interface with the program. These routines were up to about 2 pages (approx 110 lines) long. Students provided data of up to about 60 lines.
6. There was an introductory lesson for the student to become familiar with the system.
7. Students were encouraged to use the TURBO editor (resides on the PCs).
8. Several small assignments and one large project were given during the quarter.

(February 17, 1988) - Comments by Harold Coelho, TA for the course:

1. He estimated that at least 10 hrs/wk was spent by students in the PC lab.
2. The program was stored in the form of object modules on the E: drive. It was about 300K bytes.
3. There were many problems using the SEASnet facilities at the beginning of the quarter. It often took two minutes merely to log on to SEASnet.

4. The students were encouraged to work as much as possible on the C: drive as the E: drive was often slow and unreliable.

5. The task of the students was to:

   a. write Fortran interfacing routines (using an editor),

   b. compile the routines (using a compiler - Profort in this case),

   c. link the program and student object modules (using a linker),

   d. load and run the programs (using the executable file and data).

6. Most of the time spent by the students on the PCs was in doing steps a, b, and c above.

7. While the editor, compiler, and linker were stored on the C: drive, the program object modules, students source and object modules, executable file, and data files were stored on the E: drive, requiring file transfers over the net each time the above steps were performed.

8. When problems were encountered, the students were encouraged to download all files, including the program object modules, to the C: drive. Those who did this continued to do so even when the problems were cleared up.

9. In general, floppies were not used by the students in any phase of the work. PCs were used in lab work as stand alone machines.

Additional questions to Prof. Felton by the interviewer:

1. The following schedule denotes classes which were taught by the instructor and which used SEASnet. Is it correct?

Fa'85	*Wi'86*	*Sp'86*	*Fa'86*	*Wi'87*	*Sp'87*	*Fa'87*	*W'88*
-----	-----	CE165L	-----	CE165B (now 135)	CE165L	-----	CE180

Corrected 4/4/88 by G. Segal
OKed by Coelho - 4/8/88

SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor David Kay
Department:	Computer Science
Interviewer:	Geri Segal
Date:	February 19, 1988

General comments:

1. Two nets are used in this course, viz. SEASnet and PICnet (Mathematics Dept.).

2. The professor prefers PICnet since, among other reasons(4/4/88 - the hours are too short, the DOS environment is poor, and it's crowded), it permits automatic grading much more easily than SEASnet.

3. It was estimated that about 30% of the students (about 30-36 students) use SEASnet for their assignments.

4. All work is done on the E: drive. In general files are not copied to the C: drive.

5. Assignments are given in Lisp and C.

6. Editors used on the PCs were either the Turbo editor for C or the Edwin editor for Lisp.

7. There may be small data files involved.

8. The TA for the courses was Greg Wong in 3256 BH and e-mail address <gregwong>. (4/4/88 - also Jay Tobias <tobias> and Xinming Lin <xinming>)

9. When programs were being developed they may have been compiled, run, edited and replaced as much as every 5 minutes.

Course:	CS12 - Introduction to Computer Science II
Quarter:	Winter '86, '87, and '88
No. of Students:	100-120
Hrs/wk in Lab:	see below

Comments:

There were several small programming assignments (about 100 lines) given. There were two larger assignments (about 400 lines each) given, one in Lisp and one in C.

Course:	CS13 - Introduction to Computer Science III
Quarter:	Spring '86 and '87
No. of Students:	100-120
Hrs/wk in Lab:	see below

Comments:

1. Programming assignments were 1 small and 1 large program in each of Lisp and C languages. .The small programs were about 150 lines and the large programs were 1000-1500 lines.

2. On the average about 10 hours/wk were spent in the PC Labs. The long programs may have required as much as 20-30 hours on the PCs for 2 weeks each.

Additional questions to Prof. Kay by the interviewer:

1. The following schedule denotes classes which were taught by the instructor and which used SEASnet. Is it correct?

Fa'85	*Wi'86*	*Sp'86*	*Fa'86*	*Wi'87*	*Sp'87*	*Fa'87*	*Wi'88*
-----	CS12	CS13	-----	CS12	CS13	-----	CS12

(Feb 29, 1988) - Comments by Greg Wong, TA for the courses.

1. Mr. Wong has been working as a TA for the courses since Fall '86, i.e. for CS12 in Winter '87 and CS13 in Spring '87. He is now (Winter '88) a TA for CS12.

2. SEASnet accounts are given to students in CS12 and CS13.

3. The compilers/interpreters used were Microsoft C, which is a C compiler believed to reside on the C: drive, and PC Scheme. PC Scheme is a Lisp interpreter for the PC environment and is invoked by running a batch procedure, PCS, on the PCs. PC Scheme does not handle batch well and makes grading of the students assignments very difficult. It is not known as of this writing where PC Scheme resides, on the C: or E: drive. Jonathan Sin of SEASnet would have an idea of the size of PC Scheme and related files. (Note: Information from John Sin - PC Scheme is on the net.)

4. The editors that were used on SEASnet were Turbo and Edwin which is part of the PC Scheme environment. The problem with Edwin was that as the programs become larger, Edwin takes longer and longer (as much as 2 minutes) to read them in and write them out.

5. The programs read data interactively. Some of the students may have created small data files to avoid typing the same data repetitively.

6. All the students files are kept on the E: drive. No instructions were given to copy any files over and run them from the C: drive. Some students who have IBM PCs at home copied all their files to floppies and worked at home. There may be about 5 students who do this.

7. The computers are used more heavily towards the end of the quarter just before assignments are due. Students may sit down for sessions of 3-4 hours or more. PCS may be invoked every 10-15 minutes.

8. CS12

   A. Programming assignments were 1-2 medium programs (400-500 statements) in Lisp and/or C, and 1 short Lisp procedure (about 50 lines). (4/4/88 - Kay - actually 2, though only one was required to be done on the SEASnet)

   B. At least 1 assignment was given that required students to use both SEAS and PIC nets.

   C. In addition the computer facilities were available for use on the homework assignments to try out code that was written for homework. This was done by several students.

   D. There were estimated to be 90-100 students in this course.

   E. In Winter '87 the division between SEASnet and PICnet among the students was about 50-50.

   F. Note sent to interviewer by Greg Wong via e-mail:

   I just checked to see how many turned the last scheme program on the SEASnet. This program is a medium sized program, and 16 people turned in their programs on the SEASnet. 56 people turned their program in on the PICnet. This is out of a total of approx 90 students [Winter '88].

9. CS13

   A. There were estimated to be 75 students in this course.

   B. Programming assignments were 1 small and 1 large program in each of Lisp and C languages. The small programs were about 200 lines and the large programs were 1500-2000 lines.

Corrected 4/4/88 from Kay's response by G. Segal

SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor Johannes B. Neethling
Department:	Civil Engineering
Interviewer:	Geri Segal
Date:	February 24, 1988

General comments:

1. The professor uses SEASnet as a file server for the programs run in the course.

2. None of the work for any of the courses involved programming. In all cases a packaged program was used; either one purchased from the outside or one written in-house.

3. All programs were stored in the class directory on the backbone file server, and all work was done on the E: drive unless noted otherwise. No instructions were given to students to copy programs over to the C: drive and run them from there.

4. Except as noted (viz. CE 184D), none of the classes were taught in the Classroom Labs.

5. In general, the professor did not encounter problems with response time on the network.

Course:	CE184D (now 155) - Water Quality Control Systems
Quarter:	Fall '85 and '86
No. of Students:	25
Hrs/wk in Lab:	8 in Classroom Labs

Comments:

1. Packaged programs were used during classrom hours. These programs were stored as executable files (about 28 K bytes) on the E: drive and invoked approx. 2 times each classroom hour. They were interactive, menu driven programs.

2. In addition to the classroom exercises, 4 assignments were given during the quarter consisting of using packaged programs of a similar size in a similar manner. An estimate of 2-3 hours/assignment was given with the program being invoked about 5 times per assignment.

3. Lecture was taught in computer classroom.

Course:	CE184D (now 155) - Water Quality Control Systems
Quarter:	Fall '87
No. of Students:	40
Hrs/wk in Lab:	see below

Comments:

1. The same 4 assignments as 2. above were given.

2. There was a demo program (about 28K) available for the students to use to see how assignment programs were supposed to be run. It was estimated that about half the students used this program.

3. There was a help program called "browse" (about 1K) available for use. It is not known how much this was used.

Course:	CE184F (now 157) - Design of Water Quality Control Systems
Quarter:	Spring '87
No. of Students:	6
Hrs/wk in Lab:	see below

Comments:

1. The course used commercially available software - CAPDET. CAPDET is a water quality program which analyzes different options of treating water. The program goes through all possible paths in a multi-stage network of water treatment nodes.

2. The executable version of the program was stored on the E: drive and loaded to the PCs for execution. The size of CAPDET is as follows:

   Main - 300K
   Menu - 82K

3. In addition, 10 data files can be ("are" - Neethling, 4/4/88) used by the program. These files contain all the properties of the different options selected by the user at each node. For example they contain the properties associated with a particular type of filter. The size of the data files range from 30K to 90K with an average of 70K. They are overlay files (ask John Sin). These files are stored on drive E: and accessed by the program in a one of two ways: (1) Each time a node is encountered, the properties from that node are read from the data files; or (2) Properties for a particular node are read into memory once, and used from memory each time that node is encountered. It is not known in which way the files are used.

4. Each node takes approximately 5-8 minutes on the PC each time it is encountered.

5. The problems that students run are typically 3 parallel unconnected multi-stage networks. The first network is about 6 stages with 6 separate paths. The second net is about 4 stages with 4 paths. The third net is about 2 stages with 2 paths.

6. Student data files consisted of input files of about 1-2 pages and output files of 12-15 pages on the average, but may be as large as 50 pages.

7. CAPDET was used for 5 weeks during the quarter; 2 weeks intensively. It was run an estimated 5-10 times/week per student, 10 times/week during the intensive 2 weeks.

8. Lotus 1-2-3 (about 250K) was run. It was used for data analysis; there were 365 data points and about 10 different parameters. The program was invoked about 20 times per student.

Course:	CE184B (now 151) - Intro. to Water Resources Engineering
Quarter:	Winter '86
No. of Students:	20-25
Hrs/wk in Lab:	see below

Comments:

1. A special session of the class was set up at the beginning of the class to demonstrate use of the facilities, i.e. the PCs, SEASnet, etc. This totaled about 4 hours time.

2. One assignment consisted of running a small program (about 28K). This was estimated to take the students about 4 hours in which the program was invoked altogether 5 times.

3. Two assignments using Multiplan were given. This was estimated to take about 6 hours per assignment in which the program was invoked altogether about 20 times. (4/4/88 - Multiplan was executed from local floppy disks, not via the network.)

Additional questions by the interviewer:
Responses (4/4/88 - Neethling)

1. Was 184D in Fall '85 taught similar to Fall '86? Yes

2. How and when was 198 taught? Fall, Winter, Spring 1986/87.
   Is the course now numbered 15A? Yes.

3. Was Multiplan and/or Lotus installed locally on the PCs? Multiplan - yes, Lotus - no.

4. The following schedule denotes classes taught by the instructor and used SEASnet. Is it correct?

Fa'85	*Wi'86*	*Sp'86*	*Fa'86*	*Wi'87*	*Sp'87*	*Fa'87*	*Wi'88*
CE184D*	CE184B*	-----	CE184D	-----	CE184F*	CE184D	-----
			CE198	CE198	CE198		

5. Were there TAs for any of these courses? * above had partial TA support. CE198 in 1986/87 was taught by a TA.

Corrected 4/4/88 by G. Segal
Corrected 5/12/88 by G. Segal

SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor Stephen E. Jacobsen
Department:	Electrical Engineering
Interviewer:	Geri Segal and Joseph Betser
Date:	February 22, 1988

Course:	EE103 - Applied Numerical Computing
Quarter:	Fall '86 and '87
No. of Students:	100
Hrs/wk in Lab:	4 in Classroom Labs + see below

Comments:

1. There were 4 hours of Classroom Labs with 25 students each hour.

2. There were about 8 homework assignments for the course of which about 1/3 to 1/2 used computer resources (i.e. 2-4 assignments).

3. Homework assignments requiring the computer were one of three types:

   a. Students wrote small driver programs (about 20 lines), in Pascal mostly, and linked them to object modules located in the class directory. The object modules were small (1-5 Kbytes) and only one per student program was used. The Pascal compiler was provided by the professor and stored on the E: drive. It is a two-pass compiler and each pass uses a file that is about 94 Kbytes. Some students may have worked in Microsoft C which is installed locally. Students were told that the compiler could not be copied to the local hard disk (drive C:) so most probably used it from drive E:. In general it was estimated that at least half the class worked off of the E: drive exclusively. The problems involved solving systems of linear, nonlinear, and differential equations.

   b. Students used existing compiled and linked programs to solve problems. The program file sizes were on the order of 42 Kbytes (GAUSS, DGAUSS).

   c. There were one or two large graphics packages (about 243 Kbytes) which were used as stand alone programs to solve problems. These were accompanied by some small sample data files (< 1 Kbyte). One of the graphics packages was used to find the roots of nonlinear equations. The programs are menu driven. Although instructions were given to copy the package over to the local drive, it was estimated that at least half of the students used it from the E: drive.

4. There were one or two large programming projects (300-500 lines) for the course. These programs were all stand-alone, that is, they did not require any additional object modules. The main language being used was Pascal. It was estimated that about 24 hours was required to complete an assignment.

5. The TA's for the course, Sihem Ben Saad (4738 BH) or Z. Zhang, may have some additional information. Ms. Ben Saad is also located in 4803 BH and can be reached by SEASnet e-mail at <sihem>.

Additional questions by the interviewer:

1. The following schedule denotes classes taught by the instructor and used SEASnet. Is it correct?

Fa'85	*Wi'86*	*Sp'86*	*Fa'86*	*Wi'87*	*Sp'87*	*Fa'87*	*Wi'88*
-----	EE129A	-----	EE103	-----	EE239BS	EE103	-----

Corrected 6/20/88 by G. Segal

SURVEY OF SEASnet USER MODEL

Interview Summary with Instructor Who Used SEASnet in Classrom

Instructor:	Professor Richard E. Mortensen
Department:	Electrical Engineering
Interviewer:	Geri Segal
Date:	February 25, 1988
Course:	EE241C - Stochastic Control
Quarter:	Spring '87
No. of Students:	15
Hrs/wk in Lab:	see below
Course:	EE241A - Stochastic Processes
Quarter:	Fall '87
No. of Students:	35
Hrs/wk in Lab:	see below

Comments:

1. The EE Dept. has access to two computer facilities: (1) Pyramid computers which belong to EE; and (2) SEASnet.

2. Both classes used a canned program for matrix algebra manipulation called Matlab. This program was installed on SEASnet by Jim Brucker. The executable file was stored on the E: drive. The size of the program is about 360 K bytes with a 10 K Help file.

3. The program is run mainly in an interactive mode. It does, however, have the capability of reading from and writing to a data file.

4. Instructions were given to the students to copy the program on to the C: drive and run the it from there. Some students, perhaps five, may have made a copy to take home to run on their own personal computers.

5. The program was used for the homework assignments of which there were four per course. The assignments took from 1 to 6 hours each, probably toward the higher end at the beginning of the quarter when the students were becoming familiar with the facilities and the software. The students probably took one or two sessions at the terminals to complete the computer portion of the assignments.

6. The interviewer was provided with 2 sample homework assignments, one for each course, to give an indication of what was asked of the students. In addition, a listing of the data file produced by the program was provided.

7. The course EE237A, originally scheduled for Winter '87 and listed in the SEASnet Progress Report of 1987 as such, was cancelled.

Additional questions by the interviewer:

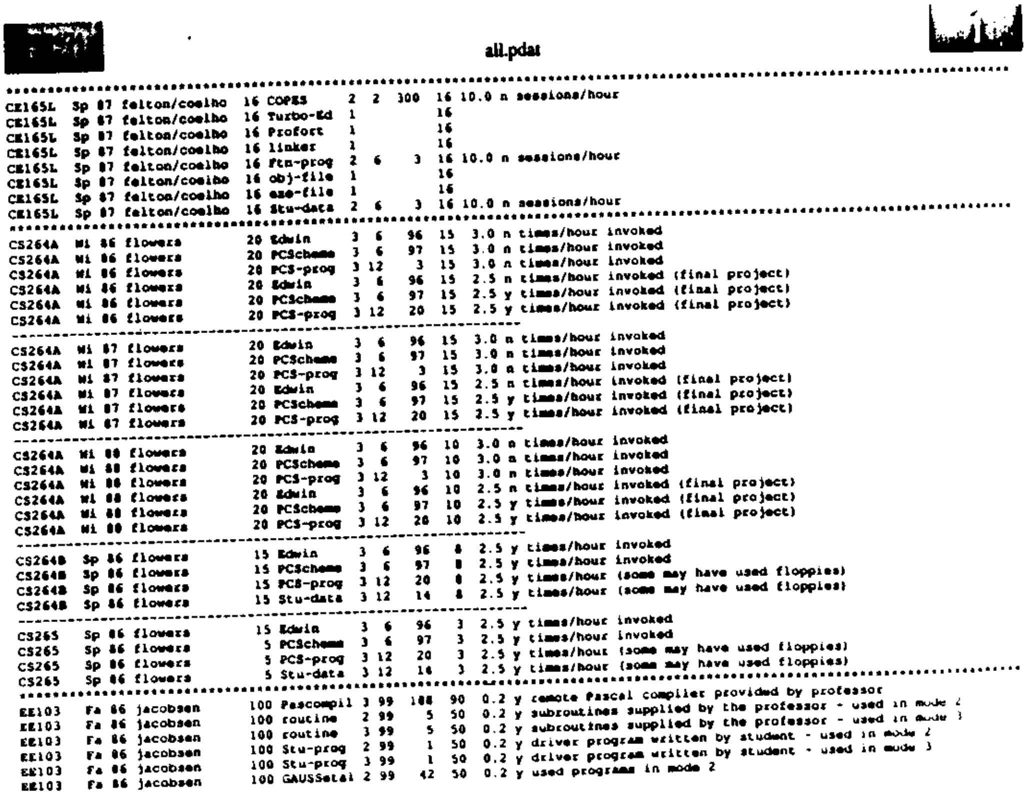
1. The following schedule denotes classes which were taught by the instructor and which used SEASnet. Is it correct?

Fa'85	*Wi'86*	*Sp'86*	*Fa'86*	*Wi'87*	*Sp'87*	*Fa'87*	*W'88*
-----	-----	-----	-----	-----	EE241C	EE241A	-----

2. Were all the students from both classes on SEASnet or were some using the Pyramids.

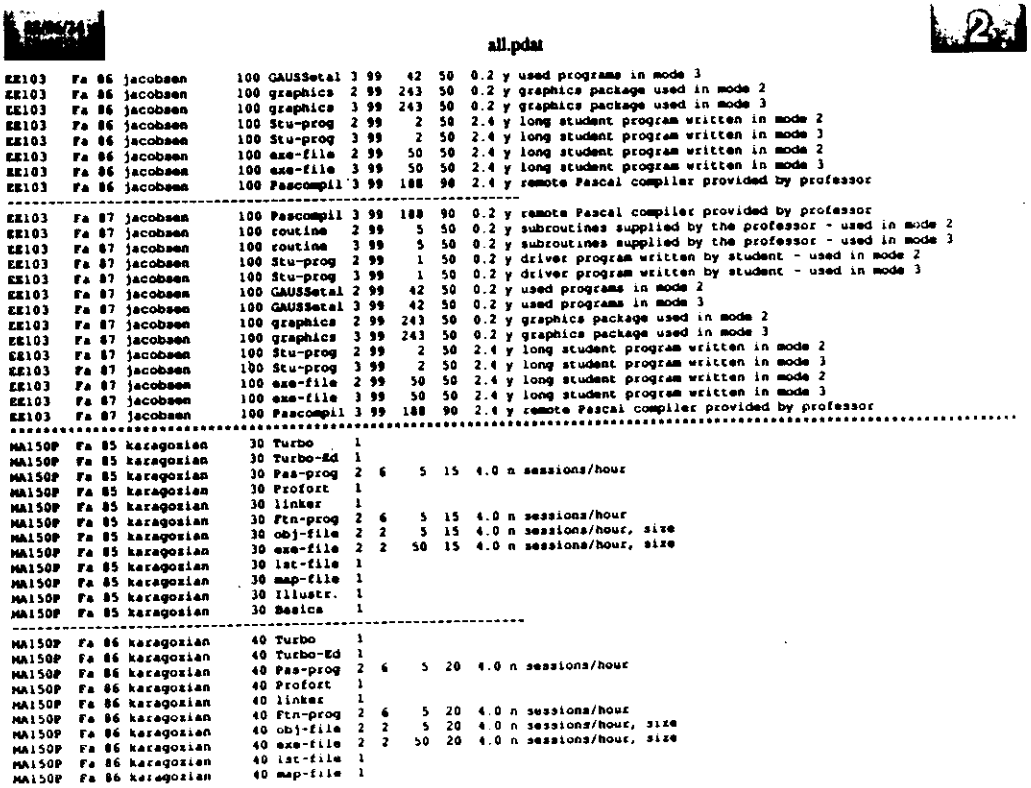
   Response (4/4/88): Essentially all were on SEASnet. Two or three may have used Pyramid.

Corrected 4/4/88 by G. Segal



all.pdat

```
**********************************************************************************************************
CE165L   Sp 87 felton/coelho  16 COPES     2  2  300  16 10.0 n sessions/hour
CE165L   Sp 87 felton/coelho  16 Turbo-Ed  1          16
CE165L   Sp 87 felton/coelho  16 Profort   1          16
CE165L   Sp 87 felton/coelho  16 linker    1          16
CE165L   Sp 87 felton/coelho  16 Ftn-prog  2  6    3  16 10.0 n sessions/hour
CE165L   Sp 87 felton/coelho  16 obj-file  1          16
CE165L   Sp 87 felton/coelho  16 exe-file  1          16
CE165L   Sp 87 felton/coelho  16 Stu-data  2  6    3  16 10.0 n sessions/hour
**********************************************************************************************************
CS264A   Wi 86 flowers          20 Edwin     3  6   96  15  3.0 n times/hour invoked
CS264A   Wi 86 flowers          20 PCScheme  3  6   97  15  3.0 n times/hour invoked
CS264A   Wi 86 flowers          20 PCS-prog  3 12    3  15  3.0 n times/hour invoked
CS264A   Wi 86 flowers          20 Edwin     3  6   96  15  2.5 n times/hour invoked (final project)
CS264A   Wi 86 flowers          20 PCScheme  3  6   97  15  2.5 y times/hour invoked (final project)
CS264A   Wi 86 flowers          20 PCS-prog  3 12   20  15  2.5 y times/hour invoked (final project)
--------------------------------------------------------------
CS264A   Wi 87 flowers          20 Edwin     3  6   96  15  3.0 n times/hour invoked
CS264A   Wi 87 flowers          20 PCScheme  3  6   97  15  3.0 n times/hour invoked
CS264A   Wi 87 flowers          20 PCS-prog  3 12    3  15  3.0 n times/hour invoked
CS264A   Wi 87 flowers          20 Edwin     3  6   96  15  2.5 n times/hour invoked (final project)
CS264A   Wi 87 flowers          20 PCScheme  3  6   97  15  2.5 y times/hour invoked (final project)
CS264A   Wi 87 flowers          20 PCS-prog  3 12   20  15  2.5 y times/hour invoked (final project)
--------------------------------------------------------------
CS264A   Wi 88 flowers          20 Edwin     3  6   96  10  3.0 n times/hour invoked
CS264A   Wi 88 flowers          20 PCScheme  3  6   97  10  3.0 n times/hour invoked
CS264A   Wi 88 flowers          20 PCS-prog  3 12    3  10  3.0 n times/hour invoked
CS264A   Wi 88 flowers          20 Edwin     3  6   96  10  2.5 n times/hour invoked (final project)
CS264A   Wi 88 flowers          20 PCScheme  3  6   97  10  2.5 y times/hour invoked (final project)
CS264A   Wi 88 flowers          20 PCS-prog  3 12   20  10  2.5 y times/hour invoked (final project)
--------------------------------------------------------------
CS264B   Sp 86 flowers          15 Edwin     3  6   96   8  2.5 y times/hour invoked
CS264B   Sp 86 flowers          15 PCScheme  3  6   97   8  2.5 y times/hour invoked
CS264B   Sp 86 flowers          15 PCS-prog  3 12   20   8  2.5 y times/hour (some may have used floppies)
CS264B   Sp 86 flowers          15 Stu-data  3 12   14   8  2.5 y times/hour (some may have used floppies)
--------------------------------------------------------------
CS265    Sp 86 flowers          15 Edwin     3  6   96   3  2.5 y times/hour invoked
CS265    Sp 86 flowers           5 PCScheme  3  6   97   3  2.5 y times/hour invoked
CS265    Sp 86 flowers           5 PCS-prog  3 12   20   3  2.5 y times/hour (some may have used floppies)
CS265    Sp 86 flowers           5 Stu-data  3 12   14   3  2.5 y times/hour (some may have used floppies)
**********************************************************************************************************
EE103    Fa 86 jacobsen        100 Pascompil 3 99  188  90  0.2 y remote Pascal compiler provided by professor
EE103    Fa 86 jacobsen        100 routine   2 99    5  50  0.2 y subroutines supplied by the professor - used in mode 2
EE103    Fa 86 jacobsen        100 routine   3 99    5  50  0.2 y subroutines supplied by the professor - used in mode 3
EE103    Fa 86 jacobsen        100 Stu-prog  2 99    1  50  0.2 y driver program written by student - used in mode 2
EE103    Fa 86 jacobsen        100 Stu-prog  3 99    1  50  0.2 y driver program written by student - used in mode 3
EE103    Fa 86 jacobsen        100 GAUSSetal 2 99   42  50  0.2 y used programs in mode 2
```

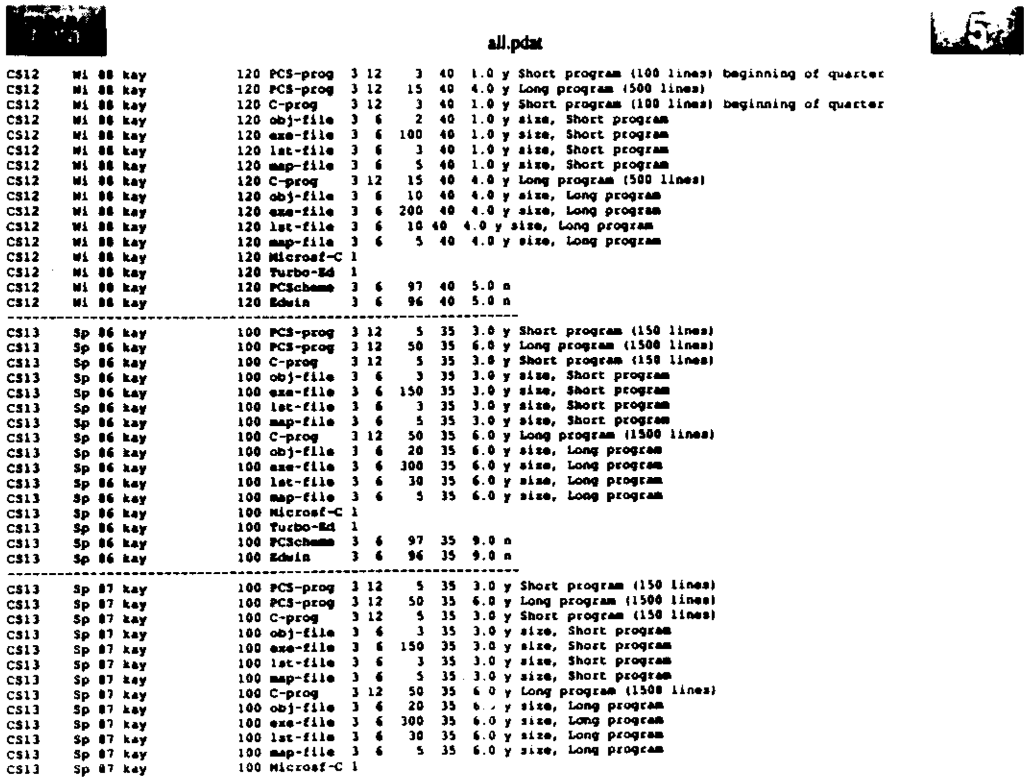


all.pdat

```
EE103    Fa 86 jacobsen        100 GAUSSetal 3 99   42  50  0.2 y used programs in mode 3
EE103    Fa 86 jacobsen        100 graphics  2 99  243  50  0.2 y graphics package used in mode 2
EE103    Fa 86 jacobsen        100 graphics  3 99  243  50  0.2 y graphics package used in mode 3
EE103    Fa 86 jacobsen        100 Stu-prog  2 99    2  50  2.4 y long student program written in mode 2
EE103    Fa 86 jacobsen        100 Stu-prog  3 99    2  50  2.4 y long student program written in mode 3
EE103    Fa 86 jacobsen        100 exe-file  2 99   50  50  2.4 y long student program written in mode 2
EE103    Fa 86 jacobsen        100 exe-file  3 99   50  50  2.4 y long student program written in mode 3
EE103    Fa 86 jacobsen        100 Pascompil 3 99  188  90  2.4 y remote Pascal compiler provided by professor
--------------------------------------------------------------
EE103    Fa 87 jacobsen        100 Pascompil 3 99  188  90  0.2 y remote Pascal compiler provided by professor
EE103    Fa 87 jacobsen        100 routine   2 99    5  50  0.2 y subroutines supplied by the professor - used in mode 2
EE103    Fa 87 jacobsen        100 routine   3 99    5  50  0.2 y subroutines supplied by the professor - used in mode 3
EE103    Fa 87 jacobsen        100 Stu-prog  2 99    1  50  0.2 y driver program written by student - used in mode 2
EE103    Fa 87 jacobsen        100 Stu-prog  3 99    1  50  0.2 y driver program written by student - used in mode 3
EE103    Fa 87 jacobsen        100 GAUSSetal 2 99   42  50  0.2 y used programs in mode 2
EE103    Fa 87 jacobsen        100 GAUSSetal 3 99   42  50  0.2 y used programs in mode 3
EE103    Fa 87 jacobsen        100 graphics  2 99  243  50  0.2 y graphics package used in mode 2
EE103    Fa 87 jacobsen        100 graphics  3 99  243  50  0.2 y graphics package used in mode 3
EE103    Fa 87 jacobsen        100 Stu-prog  2 99    2  50  2.4 y long student program written in mode 2
EE103    Fa 87 jacobsen        100 Stu-prog  3 99    2  50  2.4 y long student program written in mode 3
EE103    Fa 87 jacobsen        100 exe-file  2 99   50  50  2.4 y long student program written in mode 2
EE103    Fa 87 jacobsen        100 exe-file  3 99   50  50  2.4 y long student program written in mode 3
EE103    Fa 87 jacobsen        100 Pascompil 3 99  188  90  2.4 y remote Pascal compiler provided by professor
**********************************************************************************************************
MA150P   Fa 85 karagozian       30 Turbo     1
MA150P   Fa 85 karagozian       30 Turbo-Ed  1
MA150P   Fa 85 karagozian       30 Pas-prog  2  6    5  15  4.0 n sessions/hour
MA150P   Fa 85 karagozian       30 Profort   1
MA150P   Fa 85 karagozian       30 linker    1
MA150P   Fa 85 karagozian       30 Ftn-prog  2  6    5  15  4.0 n sessions/hour
MA150P   Fa 85 karagozian       30 obj-file  2  2    5  15  4.0 n sessions/hour, size
MA150P   Fa 85 karagozian       30 exe-file  2  2   50  15  4.0 n sessions/hour, size
MA150P   Fa 85 karagozian       30 lst-file  1
MA150P   Fa 85 karagozian       30 map-file  1
MA150P   Fa 85 karagozian       30 Illustr.  1
MA150P   Fa 85 karagozian       30 Basics    1
--------------------------------------------------------------
MA150P   Fa 86 karagozian       40 Turbo     1
MA150P   Fa 86 karagozian       40 Turbo-Ed  1
MA150P   Fa 86 karagozian       40 Pas-prog  2  6    5  20  4.0 n sessions/hour
MA150P   Fa 86 karagozian       40 Profort   1
MA150P   Fa 86 karagozian       40 linker    1
MA150P   Fa 86 karagozian       40 Ftn-prog  2  6    5  20  4.0 n sessions/hour
MA150P   Fa 86 karagozian       40 obj-file  2  2    5  20  4.0 n sessions/hour, size
MA150P   Fa 86 karagozian       40 exe-file  2  2   50  20  4.0 n sessions/hour, size
MA150P   Fa 86 karagozian       40 lst-file  1
MA150P   Fa 86 karagozian       40 map-file  1
```

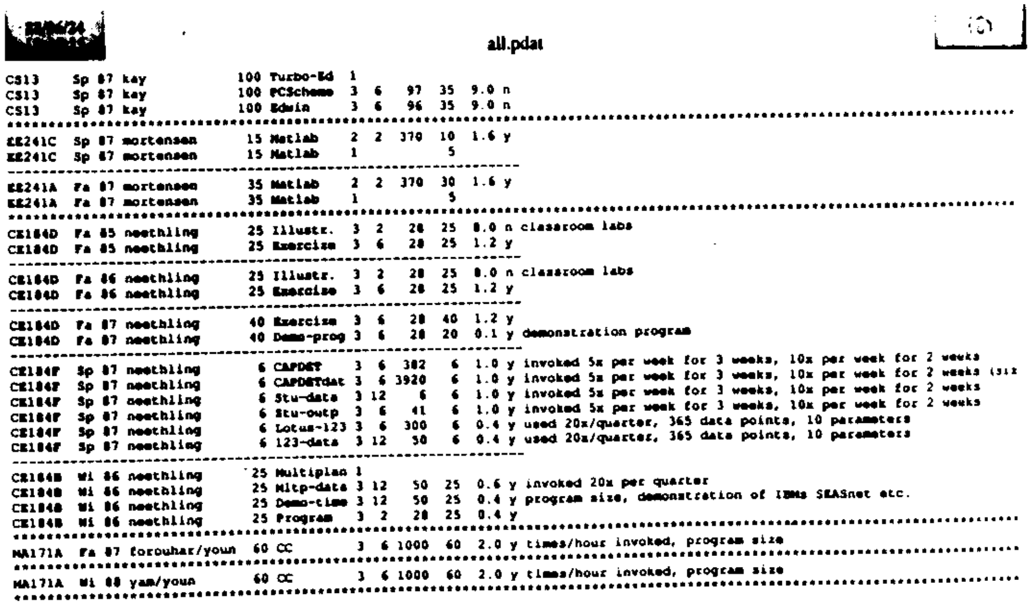
```
MA150P  Fa 86 karagozian      40 Illustr.  2  2   15  40  4.0 n sessions/hour
MA150P  Fa 86 karagozian      40 Basica    1
-----------------------------------------------------------------
MA150A  Wi 86 karagozian      70 Text      2  2   10  70  1.0 n sessions/hour, hours/week
MA150A  Wi 86 karagozian      70 Basica    1
MA150A  Wi 86 karagozian      70 Profort   1
MA150A  Wi 86 karagozian      70 Turbo     1
MA150A  Wi 86 karagozian      70 Turbo-Ed  1
MA150A  Wi 86 karagozian      70 linker    1
MA150A  Wi 86 karagozian      70 Ftn-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Wi 86 karagozian      70 Pas-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Wi 86 karagozian      70 obj-file  2  2    5  35  2.0 n sessions/hour, size
MA150A  Wi 86 karagozian      70 exe-file  2  2   50  35  2.0 n sessions/hour, size
MA150A  Wi 86 karagozian      70 lst-file  1
MA150A  Wi 86 karagozian      70 map-file  1
-----------------------------------------------------------------
MA150A  Wi 87 karagozian      70 Text      2  2   10  70  1.0 n sessions/hour, hours/week
MA150A  Wi 87 karagozian      70 Basica    1
MA150A  Wi 87 karagozian      70 Profort   1
MA150A  Wi 87 karagozian      70 Turbo     1
MA150A  Wi 87 karagozian      70 Turbo-Ed  1
MA150A  Wi 87 karagozian      70 linker    1
MA150A  Wi 87 karagozian      70 Ftn-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Wi 87 karagozian      70 Pas-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Wi 87 karagozian      70 obj-file  2  2    5  35  2.0 n sessions/hour, size
MA150A  Wi 87 karagozian      70 exe-file  2  2   50  35  2.0 n sessions/hour, size
MA150A  Wi 87 karagozian      70 lst-file  1
MA150A  Wi 87 karagozian      70 map-file  1
-----------------------------------------------------------------
MA150A  Fa 87 karagozian      70 Text      2  2   10  70  1.0 n sessions/hour, hours/week
MA150A  Fa 87 karagozian      70 Basica    1
MA150A  Fa 87 karagozian      70 Profort   1
MA150A  Fa 87 karagozian      70 Turbo     1
MA150A  Fa 87 karagozian      70 Turbo-Ed  1
MA150A  Fa 87 karagozian      70 linker    1
MA150A  Fa 87 karagozian      70 Ftn-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Fa 87 karagozian      70 Pas-prog  2  6    5  35  2.0 n sessions/hour
MA150A  Fa 87 karagozian      70 obj-file  2  2    5  35  2.0 n sessions/hour, size
MA150A  Fa 87 karagozian      70 exe-file  2  2   50  35  2.0 n sessions/hour, size
MA150A  Fa 87 karagozian      70 lst-file  1
MA150A  Fa 87 karagozian      70 map-file  1
-----------------------------------------------------------------
MA250C  Sp 86 karagozian      10 Turbo-Ed  1
MA250C  Sp 86 karagozian      10 Profort   1
MA250C  Sp 86 karagozian      10 linker    1
MA250C  Sp 86 karagozian      10 Ftn-prog  2  6   20  10  3.0 n sessions/hour, 3 large programs - so little time!
MA250C  Sp 86 karagozian      10 obj-file  2  2   20  10  3.0 n sessions/hour, size
```

```
MA250C  Sp 86 karagozian      10 exe-file  2  2  100  10  3.0 n sessions/hour, size
MA250C  Sp 86 karagozian      10 lst-file  1
MA250C  Sp 86 karagozian      10 map-file  1
-----------------------------------------------------------------
MA250C  Sp 87 karagozian      10 Turbo-Ed  1
MA250C  Sp 87 karagozian      10 Profort   1
MA250C  Sp 87 karagozian      10 linker    1
MA250C  Sp 87 karagozian      10 Ftn-prog  2  6   20  10  3.0 n sessions/hour, 3 large programs - so little time?
MA250C  Sp 87 karagozian      10 obj-file  2  2   20  10  3.0 n sessions/hour, size
MA250C  Sp 87 karagozian      10 exe-file  2  2  100  10  3.0 n sessions/hour, size
MA250C  Sp 87 karagozian      10 lst-file  1
MA250C  Sp 87 karagozian      10 map-file  1
*********************************************************************************************************
CS12    Wi 86 kay            120 PCS-prog  3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12    Wi 86 kay            120 PCS-prog  3 12   15  40  4.0 y Long program (500 lines)
CS12    Wi 86 kay            120 C-prog    3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12    Wi 86 kay            120 obj-file  3  6    2  40  1.0 y size, Short program
CS12    Wi 86 kay            120 exe-file  3  6  100  40  1.0 y size, Short program
CS12    Wi 86 kay            120 lst-file  3  6    3  40  1.0 y size, Short program
CS12    Wi 86 kay            120 map-file  3  6    5  40  1.0 y size, Short program
CS12    Wi 86 kay            120 C-prog    3 12   15  40  4.0 y Long program (500 lines)
CS12    Wi 86 kay            120 obj-file  3  6   10  40  4.0 y size, Long program
CS12    Wi 86 kay            120 exe-file  3  6  200  40  4.0 y size, Long program
CS12    Wi 86 kay            120 lst-file  3  6   10  40  4.0 y size, Long program
CS12    Wi 86 kay            120 map-file  3  6    5  40  4.0 y size, Long program
CS12    Wi 86 kay            120 Microsf-C 1
CS12    Wi 86 kay            120 Turbo-Ed  1
CS12    Wi 86 kay            120 PCScheme  3  6   97  40  5.0 n
CS12    Wi 86 kay            120 Edwin     3  6   96  40  5.0 n
-----------------------------------------------------------------
CS12    Wi 87 kay            120 PCS-prog  3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12    Wi 87 kay            120 PCS-prog  3 12   15  40  4.0 y Long program (500 lines)
CS12    Wi 87 kay            120 C-prog    3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12    Wi 87 kay            120 obj-file  3  6    2  40  1.0 y size, Short program
CS12    Wi 87 kay            120 exe-file  3  6  100  40  1.0 y size, Short program
CS12    Wi 87 kay            120 lst-file  3  6    3  40  1.0 y size, Short program
CS12    Wi 87 kay            120 map-file  3  6    5  40  1.0 y size, Short program
CS12    Wi 87 kay            120 C-prog    3 12   15  40  4.0 y Long program (500 lines)
CS12    Wi 87 kay            120 obj-file  3  6   10  40  4.0 y size, Long program
CS12    Wi 87 kay            120 exe-file  3  6  200  40  4.0 y size, Long program
CS12    Wi 87 kay            120 lst-file  3  6   10  40  4.0 y size, Long program
CS12    Wi 87 kay            120 map-file  3  6    5  40  4.0 y size, Long program
CS12    Wi 87 kay            120 Microsf-C 1
CS12    Wi 87 kay            120 Turbo-Ed  1
CS12    Wi 87 kay            120 PCScheme  3  6   97  40  5.0 n
CS12    Wi 87 kay            120 Edwin     3  6   96  40  5.0 n
-----------------------------------------------------------------
```



all.pdat

```
CS12     Wi 88 kay            120 PCS-prog  3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12     Wi 88 kay            120 PCS-prog  3 12   15  40  4.0 y Long program (500 lines)
CS12     Wi 88 kay            120 C-prog    3 12    3  40  1.0 y Short program (100 lines) beginning of quarter
CS12     Wi 88 kay            120 obj-file  3  6    2  40  1.0 y size, Short program
CS12     Wi 88 kay            120 exe-file  3  6  100  40  1.0 y size, Short program
CS12     Wi 88 kay            120 lst-file  3  6    3  40  1.0 y size, Short program
CS12     Wi 88 kay            120 map-file  3  6    5  40  1.0 y size, Short program
CS12     Wi 88 kay            120 C-prog    3 12   15  40  4.0 y Long program (500 lines)
CS12     Wi 88 kay            120 obj-file  3  6   10  40  4.0 y size, Long program
CS12     Wi 88 kay            120 exe-file  3  6  200  40  4.0 y size, Long program
CS12     Wi 88 kay            120 lst-file  3  6   10 40  4.0 y size, Long program
CS12     Wi 88 kay            120 map-file  3  6    5  40  4.0 y size, Long program
CS12     Wi 88 kay            120 Microsf-C 1
CS12     Wi 88 kay            120 Turbo-Ed  1
CS12     Wi 88 kay            120 PCScheme  3  6   97  40  5.0 n
CS12     Wi 88 kay            120 Edwin     3  6   96  40  5.0 n
-------------------------------------------------------------------
CS13     Sp 86 kay            100 PCS-prog  3 12    5  35  3.0 y Short program (150 lines)
CS13     Sp 86 kay            100 PCS-prog  3 12   50  35  6.0 y Long program (1500 lines)
CS13     Sp 86 kay            100 C-prog    3 12    5  35  3.0 y Short program (150 lines)
CS13     Sp 86 kay            100 obj-file  3  6    3  35  3.0 y size, Short program
CS13     Sp 86 kay            100 exe-file  3  6  150  35  3.0 y size, Short program
CS13     Sp 86 kay            100 lst-file  3  6    3  35  3.0 y size, Short program
CS13     Sp 86 kay            100 map-file  3  6    5  35  3.0 y size, Short program
CS13     Sp 86 kay            100 C-prog    3 12   50  35  6.0 y Long program (1500 lines)
CS13     Sp 86 kay            100 obj-file  3  6   20  35  6.0 y size, Long program
CS13     Sp 86 kay            100 exe-file  3  6  300  35  6.0 y size, Long program
CS13     Sp 86 kay            100 lst-file  3  6   30  35  6.0 y size, Long program
CS13     Sp 86 kay            100 map-file  3  6    5  35  6.0 y size, Long program
CS13     Sp 86 kay            100 Microsf-C 1
CS13     Sp 86 kay            100 Turbo-Ed  1
CS13     Sp 86 kay            100 PCScheme  3  6   97  35  9.0 n
CS13     Sp 86 kay            100 Edwin     3  6   96  35  9.0 n
-------------------------------------------------------------------
CS13     Sp 87 kay            100 PCS-prog  3 12    5  35  3.0 y Short program (150 lines)
CS13     Sp 87 kay            100 PCS-prog  3 12   50  35  6.0 y Long program (1500 lines)
CS13     Sp 87 kay            100 C-prog    3 12    5  35  3.0 y Short program (150 lines)
CS13     Sp 87 kay            100 obj-file  3  6    3  35  3.0 y size, Short program
CS13     Sp 87 kay            100 exe-file  3  6  150  35  3.0 y size, Short program
CS13     Sp 87 kay            100 lst-file  3  6    3  35  3.0 y size, Short program
CS13     Sp 87 kay            100 map-file  3  6    5  35  3.0 y size, Short program
CS13     Sp 87 kay            100 C-prog    3 12   50  35  6.0 y Long program (1500 lines)
CS13     Sp 87 kay            100 obj-file  3  6   20  35  6.0 y size, Long program
CS13     Sp 87 kay            100 exe-file  3  6  300  35  6.0 y size, Long program
CS13     Sp 87 kay            100 lst-file  3  6   30  35  6.0 y size, Long program
CS13     Sp 87 kay            100 map-file  3  6    5  35  6.0 y size, Long program
CS13     Sp 87 kay            100 Microsf-C 1
```



all.pdat

```
CS13     Sp 87 kay            100 Turbo-Ed  1
CS13     Sp 87 kay            100 PCScheme  3  6   97  35  9.0 n
CS13     Sp 87 kay            100 Edwin     3  6   96  35  9.0 n
**************************************************************************************************************
EE241C   Sp 87 mortensen       15 Matlab    2  2  370  10  1.6 y
EE241C   Sp 87 mortensen       15 Matlab    1           5
-------------------------------------------------------------------
EE241A   Fa 87 mortensen       35 Matlab    2  2  370  30  1.6 y
EE241A   Fa 87 mortensen       35 Matlab    1           5
**************************************************************************************************************
CE184D   Fa 85 neethling       25 Illustr.  3  2   28  25  8.0 n classroom labs
CE184D   Fa 85 neethling       25 Exercise  3  6   28  25  1.2 y
-------------------------------------------------------------------
CE184D   Fa 86 neethling       25 Illustr.  3  2   28  25  8.0 n classroom labs
CE184D   Fa 86 neethling       25 Exercise  3  6   28  25  1.2 y
-------------------------------------------------------------------
CE184D   Fa 87 neethling       40 Exercise  3  6   28  40  1.2 y
CE184D   Fa 87 neethling       40 Demo-prog 3  6   28  20  0.1 y demonstration program
-------------------------------------------------------------------
CE184F   Sp 87 neethling        6 CAPDET    3  6  382   6  1.0 y invoked 5x per week for 3 weeks, 10x per week for 2 weeks
CE184F   Sp 87 neethling        6 CAPDETdat 3  6 3920   6  1.0 y invoked 5x per week for 3 weeks, 10x per week for 2 weeks (siz
CE184F   Sp 87 neethling        6 Stu-data  3 12    6   6  1.0 y invoked 5x per week for 3 weeks, 10x per week for 2 weeks
CE184F   Sp 87 neethling        6 Stu-outp  3  6   41   6  1.0 y invoked 5x per week for 3 weeks, 10x per week for 2 weeks
CE184F   Sp 87 neethling        6 Lotus-123 3  6  300   6  0.4 y used 20x/quarter, 365 data points, 10 parameters
CE184F   Sp 87 neethling        6 123-data  3 12   50   6  0.4 y used 20x/quarter, 365 data points, 10 parameters
-------------------------------------------------------------------
CE184B   Wi 86 neethling       25 Multiplan 1
CE184B   Wi 86 neethling       25 Mltp-data 3 12   50  25  0.6 y invoked 20x per quarter
CE184B   Wi 86 neethling       25 Demo-time 3 12   50  25  0.4 y program size, demonstration of IBMs SEASnet etc.
CE184B   Wi 86 neethling       25 Program   3  2   28  25  0.4 y
**************************************************************************************************************
MA171A   Fa 87 forouhar/youn   60 CC        3  6 1000  60  2.0 y times/hour invoked, program size
**************************************************************************************************************
MA171A   Wi 88 yam/youn        60 CC        3  6 1000  60  2.0 y times/hour invoked, program size
**************************************************************************************************************
```

Bibliography

[AcadComp88] "Academic Computing", published by EDUCOM, Texas, 1988.

[ACIS 86} "Proceedings IBM Academic Information Systems", San Diego, 1986.

[ASEE87] "Proceedings 1987 ASEE (American Society for Engineering Education) Conference", Reno, Nevada, 1987.

[AvBeCaKa88] Alberto Avritzer, J. Betser, Jack W. Carlyle, Walter J. Karplus, "Potential for Load Sharing within Locus Server Clusters", UCLA Computer Science Department, October 1988.

[AvrGer88] A. Avritzer and M. Gerla "Load Balancing in a Distributed Transaction System", In preparation.

[BaChMuPa75] F. Baskett, K. Chandy, R. R. Muntz, and F. Palacios, " Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", JACM Col 22, pp 248-260, April 1975.

[Betser84] Joseph Betser, "Performance Modeling and Enhancement within the Locus Distributed System", M.S. Thesis, UCLA Computer Science Department, 1984.

[BetGerPop84] J. Betser, M. Gerla, and G. J. Popek, "A Dual priority MVA Model for a Large Distributed System : LOCUS", Performance '84, Proceedings of the 10th International Conference on Computer Performance (ed. E. Gelenbe), Paris France 19-21 December 1984, pp 51-66.

[Betser87] J. Betser, "Performance Evaluation and Prediction for Large Heterogeneous Distributed Systems", Ph.D. Dissertation Prospectus, UCLA Computer Science Department, Los Angeles, CA 90024, 1987.

[BeLaCaKa87] J. Betser, Nick Lai, Jack W. Carlyle, Walter J. Karplus, "LOCUS and SEASnet - Performance Analysis, Progress Report and Research Plan", Technical Report, UCLA Computer Science, 1987.

[BeCaKa88] J. Betser, Jack W. Carlyle, Walter J. Karplus, "Configuration Synthesis/Specificartion and Load Balancing for a Distributed TCF Cluster Environment", UCLA Computer Science Department, April 1988.

[BeAvCaKa88a] J. Betser, Alberto Avritzer, Jack W. Carlyle, Walter J. Karplus, "Performance Modeling and Analysis for a Large Heterogeneous Distributed System: UCLA-SEASnet", UCLA Computer Science Department, UCLA-CSD-880073, September 1988, to be presented at the ACM Computer Science Conference, 21-23 February, 1989, Louisville, Kentucky.

[BeAvCaKa88b] J. Betser, Alberto Avritzer, Jack W. Carlyle, Walter J. Karplus, "Configuration Synthesis for a Heterogeneous Backbone Cluster and PC-Interface Network", UCLA Computer Science Department, UCLA-CSD-880074, September 1988, to appear IEEE INFOCOM '89, Ottawa, Canada, 24-27 April 1989.

[BerSouMun87] S. Berson, E. de Souza e Silva, and R. R. Muntz, "An Object Oriented Methodology for the Specification of Markov Models" UCLA computer Science tech report CSD-870030, Los Angeles, CA 90024 1987 (submitted for publication).

[Cheriton88] David R. Cheriton, "The V Distributed System", CACM Vol 31, No 3, pp 314-333, Special Edition on Operating Systems, March 1988.

[CheCarKar86] Shun X. Cheung, Jack W. Carlyle, and Walter J. Karplus, "Asynchronous Distributed Simulation of a Communication Network", in Proceedings of the Summer Computer Simulation Conference, Society for Computer Simulation, July, 1986.

[DeSMunLav84] E. de Souza e Silva, R. R. Muntz, and S. S. Lavenberg, "A Clustering Approximation Technique for Queueing Network Models with a Large Number of Chains", IEEE Transactions on Computers, Vol C-35, No 5, May 1986.

[DeSoGerl84] E. de Souza e Silva and M. Gerla, "Load Balancing in Distributed Systems with Multiple Classes and Site Constraints", Proceedings Performance '84, E. Gelenbe (ed), Paris 1984, North Holland, pp 17-33.

[EagLazZah88] D. Eager, E. Lazowska, and J. Zahorjan, "The Limited Performance Benefits of Migrating Active Processes for Load Sharing", Sigmetrics 88, May 1988.

[Fox66] B. Fox, “Discrete Optimization Via Marginal Analysis”, Management Science, Nov 1966.

[Frank.etal69] H. Frank et al, “Design of Economical Offshore Natural Gas Pipeline Networks”, Office of Emergency Preparedness, Report R-1, Washington DC, Jan 1969.

[FraGerKle73] L. Fratta, M. Gerla and L. Kleinrock, “The Flow Deviation Method - An Approach to Store-and-Forward Communication Network Design”, Networks 3, 1973.

[GeleMunt76] Erol Gelenbe and Richard R. Muntz ”Probabilistic Models of Computer Systems, Part 1 (Exact Results)”, Acta Informatica 7, Springer Verlag 1976, pp 35-60.

[Gerla75] M. Gerla “Approximations and Bounds on the Topological Design of Distributed Computer Networks”, Proc. 4th Data Communications Symposium, Quebec, Canada, pp 4-7 to 4-15, October 1975.

[GifNeeSch88] David K. Gifford, Roger M. Needham, and Michael D. Schroeder, ”The Cedar File System”, CACM Vol 31, No 3, pp288-298, March 1988.

[HoKa.etal87] John H. Howard, Michael L. Kazar, Sherri G. Mences, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West, ”Scale and Performance in a Distributed File System”, technicasl report Information technology Center, Carnegie Mellon University, Pittsburgh, PA 15213, August 1987, and ACM Trans on Computer Systems Feb 1988.

[HeidLaks87] P. Heidelberger, and M. S. Lakshmi, “A Performance Comparison of Multi-Micro and Mainframe Database Architectures”, to appear in IEEE Tran. on Software Engineering, Presented in 1987 ACM Sigmetrics, Banff, Alberta, Canada, May 11-14, 1987 (Also IBM tech report RC 12230, T. J. Watson Research Center, Yorktown Heights, New York, 10598, February 1987).

[IBM88] IBM product announcement for AIX/370 Advanced Interactive Executive 370, March 1988.

[Jackson63] J. Jackson, ”Jobshop Like Queueing Systems”, Management Science, vol 10, pp 131-142, 1983.

[Kleinrock76] L. Kleinrock, “Queueing Systems, Volume II: Computer Applications”, Wiley-Interscience, New York, 1976.

[Lavenber83] S. S. Lavenberg (editor), “Computer Performance Modeling Handbook”, Academic Press, 1983.

[LaZaChZw84] Edward D. Lazowska, John Zahorjan, David R. Cheriton, and Willy Zwaenepoel, "File Access Performance of Diskless Workstations", Technical Report 84-06-01, Computer Science Department, Stanford University, Palo Alto, 1985.

[LaZaGrSe84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik, “Quantitative System Performance - Computer System Analysis Using Queueing Networks Models”, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1984.

[Liskov88] Barbara Liskov, "Distributed Programming in Argus", CACM Vol 31, No 3, pp 300-312, March 1988.

[MacNSaue85] Edward A. MacNair, and Charles H. Sauer "Elements of Practical Performance Modeling", Prentice Hall, Englewood Cliffs, New Jersey 07632, 1985.

[MoSa.etal86] James H. Morris, Mahadev Satyanarayanan, et al, "Andrew: A Distributed Personal Computing Environment", CACM, Vol 29, No 3 March 1986, pp 184-201.

[MoSa.etal86] James H. Morris, Mahadev Satyanarayanan, et al, "Andrew: A Distributed Personal Computing Environment", CACM, Vol 29, No 3 March 1986, pp 184-201.

[MorSouSoa87] Luis F. M. deMoraes, Edmundo deSouza e Silva, and Luiz F. G. Soares (editors), "Data Communication Systems and Their Performance", Conference Proceedings, Rio De Janeiro, Brazil, 1987.

[NelTowTan87] R. Nelson, D. Towsley, and A. N. Tantawi "Performance Analysis of Parallel Processing Systems", to appear IEEE Tran. on Software Engineering, Presented 1987 ACM Sigmetrics, Banff, Alberta, Canada, May 11-14 1987.

[NoBlLa.etal88] David Notkin, Andrew P. Black, Edward D. Lazowska, Henry M. Levy, Jan Sanislo, and John Zahorjan "Interconnecting Heterogeneouds Computer Systems" CACM Vol 31 No 3, pp 258-273, March 1988.

[Ousterhout.etal88] John K. Ousterhout, Andrew R. Cherenson, Frederick Douglis, Michael N. Nelson, and Brent B. Welch, “The Sprite Network Operating System”, IEEE Computer, Vol 21, No 2, pp 23-35, February 1988.

[PCI86] PC-Interface User's Manual, Locus Computing Corporation, Santa Monica (now Los Angeles), 1986.

[Pope.etal81] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, G. Thiel, "LOCUS: A Network Transparent, High Reliability Distributed System", Proc. of the 8th SOSP, Pacific Grove, CA, pp64-75, Dec 1981.

[PopeWalk85] Gerald J. Popek and Bruce J. Walker, "The LOCUS Distributed System Architecture" The MIT Press, Computer Science Series (Herb Schwetman, editor), Cambridge, Mass, 1985.

[RaghSilv86] C. Raghavendra and J. Silvester, "A Survey of Multi-Connected Loop Topologies for Local Computer Networks", Journal of Computer Networks - ISDN Sys., June 1986, 29-42.

[Rask78] Levy Raskin, "A Performance Evaluation of Multiple Processor Systems", Ph.D. Dissertation, Carnegie Mellon University, CMU-CS-78-141, August 1978.

[ReisLave80] Martin Reiser and Stephen S. Lavenberg, "Mean Value Analysis of Closed Multichain Networks", JACM Vol 27, No 2, April 1980, pp 313-322.

[SaHo.etal85] M. Satyanarayanan, J. H. Howard, D. N. Nichols, R. N. Sidebotham, A. Z. Spector, and M. J. West, "The ITC Distributed File System: Principles and Design", Proc. 10th Symposium on Operating Systems Principles, Orcas Island, Washington, 1-4 December, 1985, pp 35-50.

[SaueMacN83] Charles H. Sauer, and Edward A. MacNair "Simulation of Computer Communication Systems", Prentice Hall, Englewood Cliffs, New Jersey 07632, 1983.

[SEAS85] Michael K. Stenstrom, "SEASnet - Coursware Development Projects 1985", UCLA School of Engineering and Applied Science, 1985.

[SEAS86] Michael K. Stenstrom, "SEASnet - Coursware Development Projects 1986", UCLA School of Engineering and Applied Science, 1986.

[SEAS87] Michael K. Stenstrom, "SEASnet - Coursware Development Projects 1987", UCLA School of Engineering and Applied Science, 1987.

[Segal88] Geri Segal, "A Method for Estimating Traffic on a Heterogeneous Computer Network", M.S. Comprehensive Report, UCLA Computer Science Department, 1988.

[ShelPope86] Alan B. Sheltzer, and Gerald J. Popek, "Internet LOCUS : Extending Network Transparency to an Internet Environment" IEEE Trans on Software Engineering, 1986.

[Stenstro87a] Michael K. Stenstrom, "SEASnet - A Network for Educational Computing", Proceedings 1987 ASEE (American Society for Engineering Education) Conference, Reno, Nevada, pp. 1540-1543, 1987.

[Stenstro87b] Michael K. Stenstrom, "Teaching Numerical Methods in a Workstation/Server Environment", Proceedings 1987 ASEE (American Society for Engineering Education) Conference, Reno, Nevada, pp. 1544-1546, 1987.

[TantTows85] A. N. Tantawi and D. Towsley, "Optimal Static Load Balancing in Distributed Computer Systems, JACM, Vol 32, No 2 April 1985, pp 445-465.

[TanTowWol88] A. N. Tantawi, D. Towsley and J. Wolf, "Optimal Allocation of Multiple Class Resources in Computer Systems", Sigmetrics 88, May 1988, pp 253-260.

[WoodTrip86] C. M Woodside and S. K. Tripati, "Optimal Allocation of file Servers in a Local Network Environment", IEEE Trans. on Software Engineering, Vol 12, No 8, August 1986, pp 844-848.

[Zipf49] G. K. Zipf, "Human Behavior and the Principle of Least Effort",

Addison-Wesley Publishing Company, Reading, Mass, 1949.