APPLICATION OF SIMULATION TO DETAIL DESIGN OF A TELEPHONE DIRECTORY ASSISTANCE SYSTEM COMPUTER NUMBER 68

John A. Noecker
Bell Telephone Laboratories
P. O. Box 2020, Room 1B-346
New Brunswick, New Jersey 08903
Telephone 201-463-6640

INTRODUCTION

Directory Assistance operators provide telephone number information to telephone customers who dial 411, 555-1212, or any other code which may be used in their local calling area. Approximately 25,000 operator positions are required to provide this service in the Bell System. Operators search for the requested telephone numbers in frequently updated directories.

A Directory Assistance System Computer (DAS/C) is being considered as a solution to the problem of meeting rapidly increasing demands for this service. DAS/C is a real time information retrieval system designed nominally to serve 500 terminals from a data base of 2 million telephone number listings with an average response time of 1/2 second, and capable of being configured at less than 1/2 nominal size and greater than twice nominal size. The large potential value of this mechanization demands great care in design of the computer system and also supports extensive use of simulation in detailed design of that system.

This paper presents results of several DAS/C system design studies and shows how simulation is used to support these studies at the detail level while simultaneously monitoring the overall design to assure meeting the system objectives.

DAS/C SYSTEM OVERVIEW

A basic DAS/C entity (Figure 1) consists of one Information Retrieval Center (IRC), sixteen data links to eight Directory Assistance Bureaus (DAB), and on the order of 100 voice lines connecting a number of central offices to one DAB.

The DAB design includes the personnel system, the Directory Assistance operator job, and a hardware system for controlling the terminals used by the operators. The design of the IRC involves hardware-software design questions. Here two major computer functions are involved; inquiry and update. The inquiry function is required to service Directory Assistance operator requests for particular telephone numbers. The update function is required to effect the listing changes required as a result of the customer requests for installations changes or terminations of telephone service. The basic inquiry functional design is based on work by H. I. Rothrock.[1]

The Directory Assistance operator's job is to receive the customer's request, search for the requested telephone number, and respond to the customer with the requested number or nonexistence thereof. The computer performs only a partial search and presents a number of telephone number listings to the operator for final selection of the requested number. To initiate the computer search the operator selects search arguments from the details (surname, street name, next name, etc.) volunteered by the customer and normally submits three characters of each selected detail to the computer. The computer retrieves all customer listings matching the operator's search arguments and presents the first CRT page of these listings for selection. If more than one page is retrieved, the operator can see additional listings by submitting a page flip request to the computer.

Directory Assistance is characterized in the Bell System by extensive data describing Directory Assistance operator positions in terms of groupings, number and size of directories, types of requests serviced, human performance, and other details. Considerable data are also available to characterize the files and in particular the frequency distribution of the number of listings retrieved to service customer requests. An experimental system is being used to study the interaction of operators with a computer system as they serve live Directory Assistance traffic.

This paper is concerned with the systems analysis associated with the IRC inquiry function. The analyses of the IRC update function and the operator's job is currently under way and may be reported in the future.

A WORD ON APPROACH

Design questions are defined with an objective to establish the significance of controllable variables and uncontrollable variables within the performance range of interest to the DAS/C project. Controllable variables (file block size, buffer pool size, spare space in the block) are those which the system designer or system operator has alternatives to choose from in controlling system performance. Uncontrollable variables (file

growth rate, call load, some operator behavior) are those that are difficult or impossible to affect.

A continual effort is made to avoid abstract or all-encompassing design problem definitions because of difficulties encountered relating results to design and operations practices. The approach used to avoid abstract problem definitions begins with an assignment of portions of the maximum allowed average response time to each of the major functions; e.g., program execution, disk accessing (See Figure 2). Experience with early simulations leads to the assignments shown. With a breakdown of the system into functions and with these maximum response time assignments, a function design question can be pursued in an understandable and manageable way. As each function design progresses, its performance characteristics are continually tested in a total system model. A version of the total system model was presented at the Third Conference on Applications of Simulation.[2]

This functional approach to design also facilitates modeling. A total system model can be maintained which is concerned with major effects of each function at a level of detail most suitable for modeling efficiency. At the same time separate function models can be concerned with a level of detail commensurate with the needs of the function design decision. This balance between modeling efficiency and level of detail is very difficult if not impossible in an all encompassing model. Many times the greatest difficulty resides in human inability to simultaneously comprehend all the ramifications of a large model or system.

Each design study is pursued to a completion defined as follows: that set of alternatives, and ranges in variables of these alternatives, imagined at this point in design to encompass most of the likely design choices to be made through system implementation. For example, the communication line control study deals with five control methods ranging from a simple direction switching method to one incorporating priority and pre-emption. Each method is analyzed for communication line utilization ranging from 0 to 90 percent. Though this "completion" is not perfect, "complete" design problem study in this sense provides a good base for a fuller study or reorientation at a later stage of system development.

The results of subsystem design studies are considered in the complete environment of the design decision to be made. For example, program modularity studies suggest the elimination of priority and pre-emption. To avoid the use of priority and pre-emption requires waits for periods of several milliseconds before preserving interrupt information. The risk of overwriting an interrupt with another interrupt in that time period is too great. The analysis serves only to discourage use of processing priority and pre-emption of program execution except where it is logically required.

DESIGN STUDIES

A summary of systems analysis results is given near the end of the memorandum. The detailed reviews are given now.

Communication Line Control

DAS/C communication involves half duplex 50 kilobit data transmission between the Information Retrieval Center and the eight Directory Assistance Centers. The choice of a 50 kilobit line is made to facilitate transmission of messages up to 1500 bytes long from the Information Retrieval Center to the Directory Assistance Center and the operator's terminal within the response time limitation. The input message is between 10 and 30 bytes long. The designer is led quite naturally, because of the large difference between the length of the input and the output messages, to the use of priority for the input message in order to avoid relatively long delays. This design study involves the performance of two nonpriority and three priority communication control methods:

1. Single Server
2. First-Wait-First-Served
3. Input Priority
4. Input Pre-emption - Resume
5. Input Pre-emption - Restart

The single server method involves switching the direction of transmission whenever the queue for the active direction is exhausted. The first-wait-first-served discipline provides for servicing of messages in the order of arrival without regard to the transmission direction. Giving priority to the input message means that output messages wait until the input queue is empty before being serviced. The input queue is checked at the end of each output message transmission. Pre-emption - resume involves priority for the input message. With this method, however, the arrival of an input message causes a pre-emption of transmission of an output message. After the input queue is empty the output transmission resumes at the point of interruption. The pre-empt - restart method is the same as the pre-empt - resume method with the exception that output restarts from the beginning when the input queue is empty.

Separate GPSS simulation models were developed for each of the five communication line control methods. Each model consists of about 50-100 GPSS blocks and represents only the significant queueing features of the communication subsystem. These models were exercised for refinement of the control method and then finally to develop the data points for the graphs shown in Figure 3.

Figure 3 shows for each method the total communication time as a function of message load. On this chart if a horizontal line is drawn at the 200 millisecond response time assignment taken from Figure 2, the capacity of the communication line is specified for each method of control by the intersection of the respective response time curve with the 200 millisecond line. It can be seen that, with the exception of the pre-empt - restart method, all methods are relatively close together in performance and well above the nominal design load of each DAC (2.5 requests per second).

Viewing only this chart, Figure 3, one is led to a choice of the pre-empt - resume method. However, control program complexity should be considered. The bar chart of Figure 4 shows the relative control program complexity versus relative performance of each method. The vertical heights of the bars indicating complexity are rough estimates. If the estimates are accurate, a choice of a simple control method involves a relatively small sacrifice of performance with the advantage of less programming complexity. Further study of the programming complexity will probably be required before a control method is chosen.

Main Storage Requirements and Buffer Allocations

Work areas required in main storage can be classified four ways: 1) data pertinent to one operator request, 2) data pertaining to a single call, 3) data associated with an operator or operator positions, and 4) data used to control the computer system. Work area requirements of the last two classifications are usually fixed in size and assignment over periods of an hour or more. Of course, it is desirable to keep data in these fixed areas to a minimum. Thus data is placed in the first two classes as much as possible, and dynamic allocation of work areas is used in order to conserve space. Dynamic allocation is particularly valuable because 500 operators average less than 7 requests in the computer system simultaneously (21 requests/sec * 0.3 sec average processing time).

Four dynamically allocated work (buffer) areas are defined in the Directory Assistance System/Computer:

1. Second level index retrieval
2. Listings retrieval
3. Page information
4. Communications output

Four allocation methods were considered along with several minor variations. At one extreme buffer elements are statically assigned in pairs to each file channel and communication line. This method requires 90,000 to 100,000 bytes for buffers. At the other extreme elements are allocated when required in the size required but from a common pool. This method requires the least space for buffer; 14,000 bytes.

The common pool allocation method uses main storage space most efficiently but the allocation programs are very complex. Complexity is reduced greatly if all allocations from a pool are for the same size buffer element. Overhead processing time is reduced linearly as the number of allocations are reduced. The third allocation method involves two pools; one for index and listings and the other for page formation and output communications. Use of the index work area does not overlap, in time, use of the listings work area so one allocation can serve both uses. This also applies to the page formation and output communications areas. This method requires 15,000 bytes of main storage space. Finally for the fourth allocation method all buffer space required to service a request is allocated at the beginning of the request processing. This method, though quite similar in programming complexity, reduces the overhead 50 percent over the third method. The buffer space requirement is 30,000 bytes.

A final decision on buffer allocation method requires a more rigorous consideration of allocation programs. This may, in turn, call for additional performance analysis work.

All experiments associated with main storage requirements and buffer allocation were carried out on the total system model. It is interesting to note that buffer pools act as multiple server queueing systems. Thus the maximum request load for a finite queue situation is almost coincident with the request load limit set by the maximum function response time (Figure 2).

Listing File Blocking

The listing (name, address, and telephone number) file on the disk packs is grouped into blocks of listings. Within a block of listings the listings are arranged in a continuous string with spare space provided at the end of the block to allow insertion of new listings. Two block sizes are considered in this study: a full block of 60 listing spaces and a half block of 30 listing spaces. Since, for a typical file, 95 percent of the retrievals involve less than 24 listings, the half track block seems desirable.

Choice of a full track block will require a minimum additional 10,000 bytes of main storage and as much as 36,000 bytes depending on the buffer allocation method chosen. The most likely amount of additional main storage required is 20,000 bytes. Use of a full track block would, however, result in a 1 million byte savings of fixed head file device space. Main storage costs are approximately 100 times the fixed head file device costs (1 cent per thousand bytes).

Spare space allocation in the disk pack listing blocks has been shown in a separate study to be an effective control of the frequency of file reorganization. Since overflow can be very detrimental to retrieval performance, it is assumed for this study that a reorganization takes place each day an overflow occurs during updating of the file. A sensitivity study (Figure 5) compares the effects of file growth rate, file size in listings, and spare space, on the time between reorganizations for a file using a full track block. From Figure 5 it can be seen that a three listing range of spare space (8 to 11) can offset the effects of either file size or growth rate over ranges anticipated in the Bell System. For this discussion spare space in a full track block is set at 9. This provides an average reorganization interval of 20 working days under the most likely conditions of implementation of DAS/C.

A similar study of the half-track block size shows a requirement of seven spare spaces for a 21-day average reorganization interval. Thus, a track blocked in full track blocks can hold 51 listings but if it is blocked in half-track blocks the track capacity is 46 listings. In the typical file this difference results in a requirement for 80 million bytes additional disk pack storage space to allow for half-track blocking, at about $250,000 purchase.

These studies of file reorganization intervals and spare space assume an even distribution of update activity throughout the file. The effects of high concentrations of activity in small areas of the file should be studied. The effect of a higher activity rate, a shorter interval, is offset by a smaller effective file size (area of high concentration). Only complete elimination of overflow areas are given consideration in this study. The effects of a small overflow area (say less than 100 listings) should be considered. The reorganization interval might be increased but so also might the inquiry response time be increased with the occasional additional access to the overflow area.

The disk pack requirements appear dominant in this design decision. However, other aspects of the choices must be considered before a decision can be made.

The total system model was used to determine the amount of storage required in core for each of the various listing file blocking sizes. In addition, a PL-I simulation model was used along with an analytical approximation of the reorganization interval in order to determine the effects of file growth rate, file size in listings, and spare space on the time between reorganizations of a file. The approximation was used to explore the effects of the mean interval between reorganizations and the PL-I model was used to develop a distribution of reorganization intervals about the mean.

Listing Retrieval and Search Strategy

The computer search for telephone number listings begins with indexing and retrieval of listings from disk using a basic six character key. To form the basic key, characters are taken from the most productive details (finding name, street name, next name, etc.) submitted by the Directory Assistance Operator. After listings have been retrieved into core a search is performed on any additional details submitted by the operator. If indexing on the basic six character key results in a requirement for a retrieval of multiple blocks of listings, one alternate plan is to retrieve all blocks at one time (Figure 6, left). If additional details are available, a search is performed on all the retrieved listings. The matching listings are then formed into pages and processing continues to move these pages to the page queueing and the first page to the communications control programs.

Another alternative to this approach in multiblock retrievals is to retrieve one block of listings at a time and, if additional details are available, search that block of listings (Figure 6, right). At the end of the search the matching listings are moved out of the search area into a page formation area before another block is retrieved into core for searching.

Some side design problems were encountered in the preparation of the simulations for these two approaches. For example it was discovered that five or six tracks might be retrieved and searched to form a single page. On the other hand, it may be necessary to form as many as three pages in main storage out of a search of two tracks.

The main effects on the system are seen in the file buffer requirements and disk channel occupancy. The disk channel occupancy is increased approximately 15 percent as a result of going to retrievals of a block at a time, (from .50 to .58, (.58/4 channels = 14% utilization each channel)). Changing from retrieval of all blocks of listings to retrieval of one block at a time results in a reduction of 10,000 bytes of file buffer space needs. The response time change is less than 10 percent and all other utilizations in the system stay the same. With the increased channel load, however, the disk access function will reach its maximum allowed response time (Figure 2) at a lower system load.

Early in the simulation effort it was discovered that very long searches seriously degrade performance. This precipitated a realization of the effect of the associated page flipping activity on the operator call service time. Therefore the search was limited during early simulation activity to 200 listings, i.e., if keyed input results in a retrieval longer than 200 listings, the listings are not displayed. Additional details are requested by

the operator and another search key is given or the customer is told that we are unable to help without more information. This listing limit was tested to see how an increase to 1,000 listings affects performance. The increase to 1,000 listings is desirable in order to reduce the "not founds" and the operator requests to the customer for additional information.

Raising the search limit to 1,000 listings added 10 to 12 percent to the average response time and 5,000 to 8,000 bytes to the file buffer needs. The fixed head file device occupancy increased from 0.65 to 0.75 and the disk channel occupany increase is negligible.

Later in the design effort it became apparent that the listing limit is not practical if more than 2 details (basic key) are given since it is not possible to know how many listings match the request until all the listings are retrieved and searched. Thus two limits were established; 1) three tracks (180 to 210 listings) if two details are given, and 2) ten tracks retrieved and searched if three details are given. The effects of this change on DAS/C performance and component occupancies were all negligible.

The total system model was used to determine the main effects on the system for the two alternative strategies. A separate simulation which operated only on the nine distributions of number of listings retrieved for various file configurations was used to develop the experimental parameter values to be used in the total system model.

Program Modularity, Priority, and Pre-emption

The DAS/C load is characterized by a few types of requests (tasks), all with similar processing program execution times. Most of these request types are of the same priority. This contrasts with the load on most large computer systems which serve tasks with processing requirements of wide variation in length and urgency. Programs with small requirements are threatened with long waits for completion of programs with large requirements. To alleviate this threat, programs with large requirements are sometimes broken into modules to allow smaller programs access to the computer within reasonable intervals. Priority and more recently pre-emption are used extensively to keep delays for small programs commensurate with their processing requirements. However, use of modularity, priority, and pre-emption in DAS/C warrants study to determine the effects on performance of this system with similar processing requirements.

In this study program modularity is investigated first for a continuous line of program execution, i.e., uninterrupted by I/O. Breaking the program into nine equal modules increases the response time (program execution elapsed time) 16 percent when the processor utilization is 30 percent. At 60 percent processor utilization the response time increases 32 percent when this modularity is introduced, and the response time increases 62 percent at 90 percent processor utilization. Introduction of a descending priority scheme changes these increases to 32 percent, 90 percent, and 328 percent respectively. Descending priority is used commonly as, for example, in processing of a communications interrupt through various operating control system routines to finally give control to the execution of application programs as depicted in Figure 7. Figure 7 shows 7 processing modules at three priorities; 90, 80, and 20. In this scheme all transactions arriving at the first module will be processed to the "Activate Program" module before execution of that module for any preceding transaction.

Priority and pre-emption is quite often necessary in order to insure preservation of the interrupt information. A multiplexor I/O processor, for example, will set up line addresses and interrupt status in its registers and then raise an interrupt line to the computer. A good design minimizes chances of losing this first interrupt if, for example, a second cause for an interrupt arrives before information for the first interrupt is preserved in the computer. Priority and pre-emption is usually required to assure that at least enough of the common interrupt processor is executed to stack the interrupt and its information in a queue.

The designer is then faced with a decision to return to the pre-empted program or continue with administrative work on the pre-empting transaction to place it in a queue for the pre-empted program. For example, in Figure 7 let us assume transaction 1 had partially processed the presearch module when the interrupt associated with transaction 2 arrived at the common interrupt handler module thereby causing a pre-emption of transaction 1 processing. The decision must be made after processing of transaction 2 in the common interrupt handler is complete and the transactions resides in queue for the interrupt processing module. The simulations indicate a strong advantage to continuing with the presearch processing until it is completed for transaction 1 before proceeding with transaction 2. If the pre-empted program execution time is eight to ten times greater than the time to execute the pre-empting transaction, the effect on total response-time is minimal. However, if the situation is reversed, i.e., pre-empting ten times greater than pre-empted, the total response-time is increased nearly 100 percent at 60 percent processor utilization and about 300 percent at 90 percent processor utilization.

Initial studies were made on the effects of module size in a modularized string of processing. Equal size, increasing size, and decreasing size strings were studied and the effects were relatively insignificant.

Initial studies were also made on the effects of I/O (file accessing) between strings of processing. It appears that I/O operations exceeding processing strings in time by a factor of ten or more cause the processing strings to behave independently from a queuing standpoint.

To summarize, at this point in our studies it appears that modularity, priority, and pre-emption in a continuous line of program execution is undesirable. If pre-emption is required, it is desirable to pre-empt for as short a time as possible. If modularity is used choice of module size is relatively ineffective. If an I/O operation occurs the strings of processing on either side of the I/O operation can be treated independently. However, the designer must watch for possible interaction effects.

The effects of program modularity, priority, and pre-emption were studied with several small simulations representing in general the variations that can be used in configuration of program modules and how these configurations interact with file activity. Upon completion of the work with the separate models two or three of the best program configurations were applied to the total system model.

DESIGN STUDIES SUMMARY

Before proceeding to a look at the future, it seems desirable to attempt a summary of results. At this point the design studies probably appear to be a collection of independent activities. Actually they serve to advance the design of the inquiry subsystem and in this way they are closely related. The results of the studies are presented here in one paragraph to give a sense of overall system design.

The communications line-control study suggests use of a simple first-in/first-out queuing method or a line direction switching control method. A best choice for file block size appears to be a full track (approximately 60 listing spaces). Listing retrieval and searching should be done a block at a time. Dynamic main storage allocation should occur only once per operator request and in sufficient quantity to provide for all buffering needs to process the request. And finally, modularity, priority, and pre-emption are discouraged for use in continuous strings of program execution during processing of operator requests.

FUTURE SYSTEMS ANALYSIS

Systems analysis may continue on a lower key with computer system design problems and may expand in areas of the operator's job and the complete network of Directory Assistance. Work on the operator's job might well lead to a redesign of the computer system. Work on computer system design problems should give reasonable consideration to this possibility as it has in the past.

Computer system design problems presently under consideration are the page queuing functions and the update functions. Anticipation of problems to be considered is frustrated by the infinity of problems available and not knowing which will appear relevant at each point of the design process. This requires flexibility and a capability to complete a design study within a few weeks after the problem appears relevant. The approach described here, which uses small models to represent only significant aspects of the function involved in the design study, facilitates the flexibility required to respond quickly to design questions. At the same time overall system performance can be assured with the total system model and adjustments can be made in the function maximum response time (Figure 2) if necessary.

Work on the operator's job should involve a comparison of DAS/C, several alternative systems, and the present Directory Assistance operator job to determine the effects of each alternative on Directory Assistance in the Bell System.

Work on the complete DAS network will probably begin with renewed use of a market analysis program which was developed to help characterize the population of Directory Assistance offices and relate these characteristics to the design characteristics of the several Directory Assistance systems. This work views the subsystems (switching equipment, DAB's, and IRC) as black boxes described functionally. For example, the Directory Assistance Bureau is viewed as a voice to data concentrator.

ACKNOWLEDGMENTS

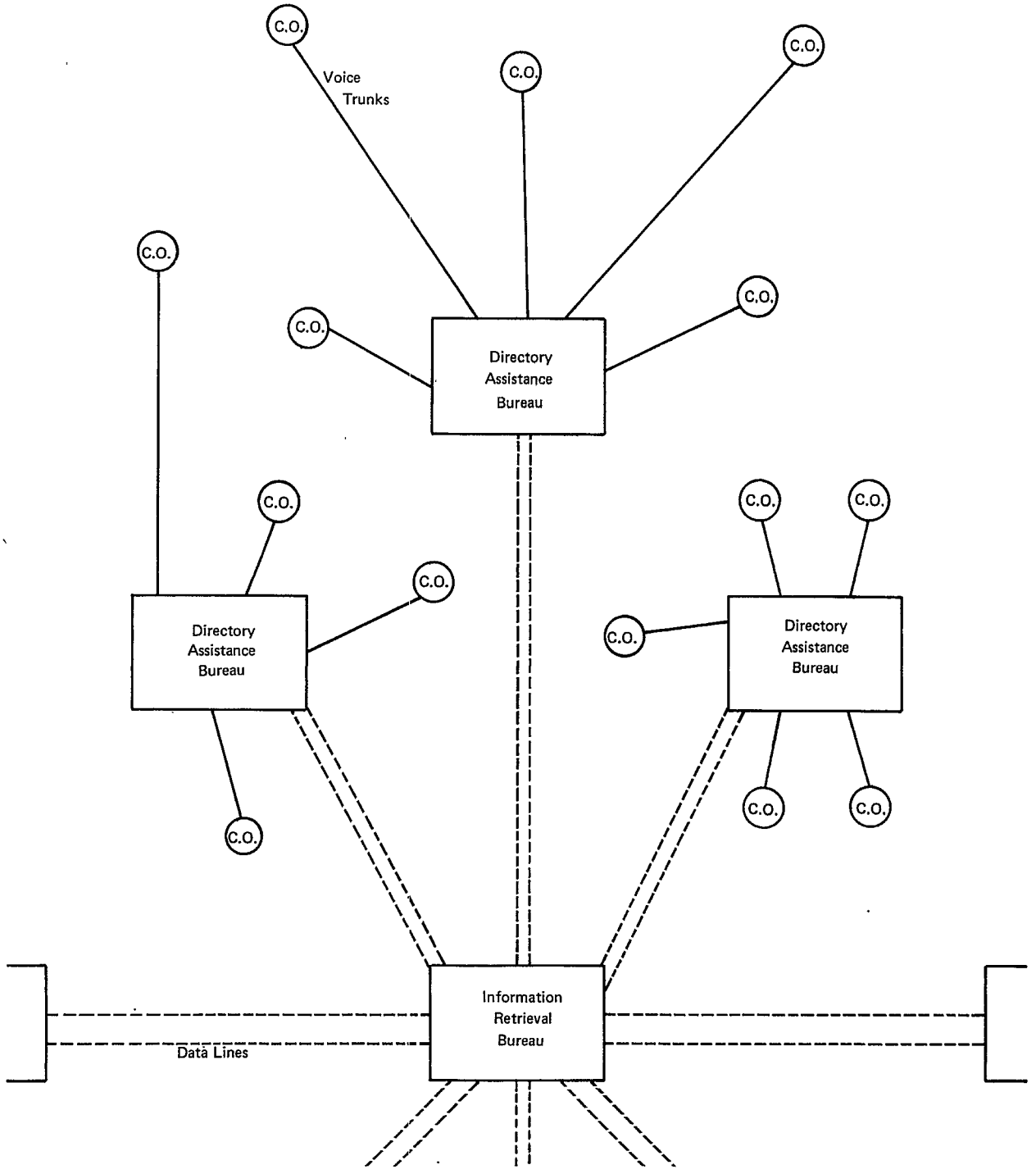
Mr. W. A. Hall of the Program Development and Training Department and Mr. Y. B. Eng of the DAS/C Design Department, both at Bell Laboratories, prepared all simulation models associated with these design studies. Specifically, Mr. Hall prepared models used to study listing file blocking and reorganization, listing retrieval and search methods, and main storage allocation methods. He also modified the total system model as required to check the overall effects of design decisions. Mr. Eng developed models for studies of communications line control, file reorganization, and program modularity. He also developed a first version simulation of the DAB cluster Controller. Our several close associates in the DAS/C Design, the Operating Control System, and the Systems Analysis Departments must be acknowledged for their important and helpful encouragement and support.

BIBLIOGRAPHY

1. H. I. Rothrock, "Computer Assisted Directory Search," PHD Thesis, University of Pennsylvania, May 1968. Avail: University Microfilm Order No. 69-164.

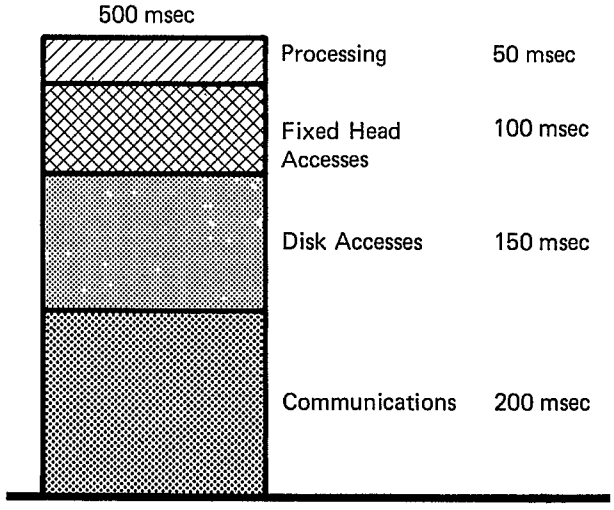
2. W. A. Hall, "A General Simulation Model of an On-Line Telephone Directory Assistance Retrieval System, "Proceedings of the Third Conference on Applications of Simulation," December, 1969.

DIRECTORY ASSISTANCE SYSTEM COMPUTER
BASIC ENTITY
FIGURE 1

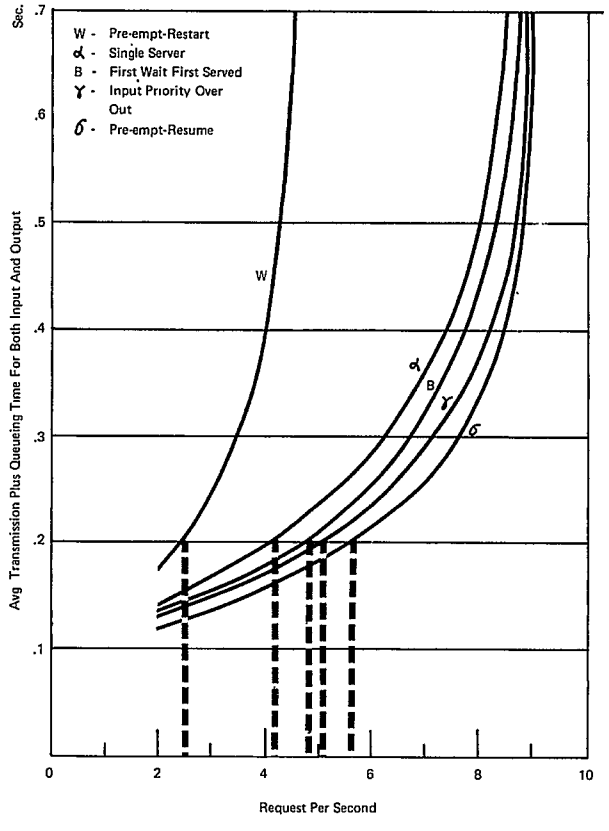


Directory Assistance System/Computer
Response Time Assignments
Figure 2

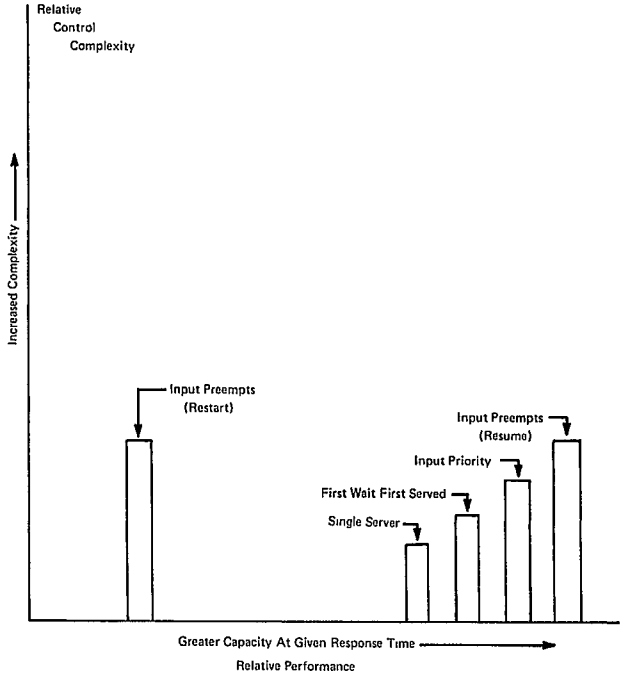
Allocation of Response Time . . .



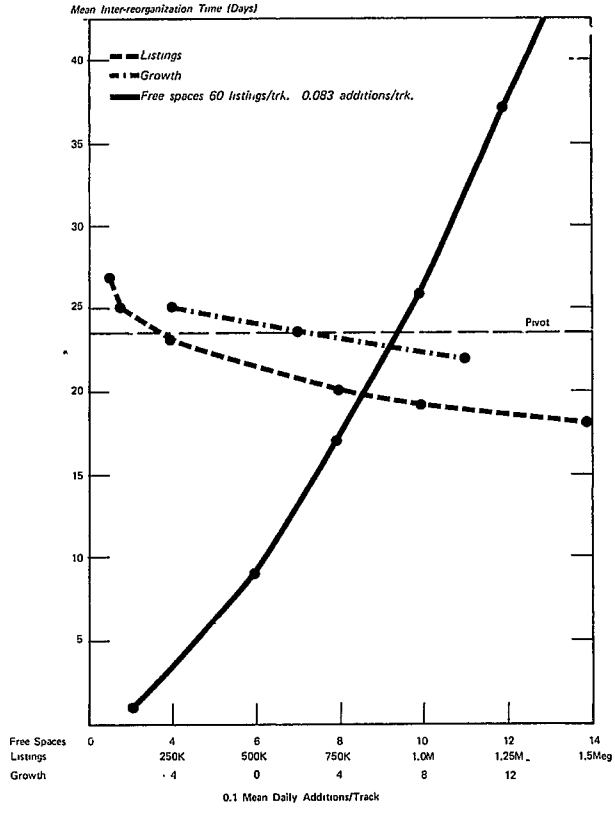
DIRECTORY ASSISTANCE SYSTEM/COMPUTER
COMMUNICATIONS CONTROL-LOAD PERFORMANCE CURVES
FIGURE 3



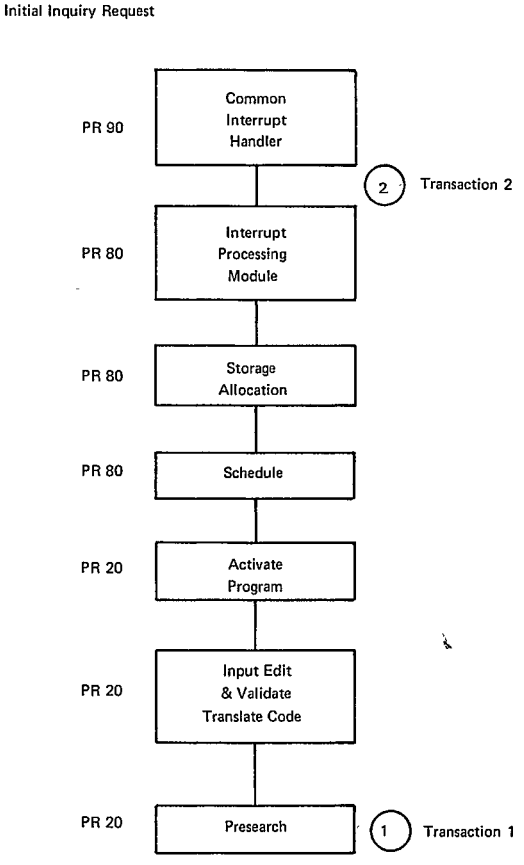
DIRECTORY ASSISTANCE SYSTEM/COMPUTER
COMMUNICATIONS CONTROL
PROGRAM COMPLEXITY vs PERFORMANCE
FIGURE 4



Directory Assistance System/Computer
Figure 5
File Inter-reorganization Time
Sensitiuity Study



DIRECTORY ASSISTANCE SYSTEM/COMPUTER
PROGRAM MODULARITY, PRIORITY, AND PREEMPTION
FIGURE 7



DIRECTORY ASSISTANCE SYSTEM/COMPUTER
LISTING RETRIEVAL AND SEARCH STRATEGY
FIGURE 6