A GENERAL SIMULATION MODEL OF AN ON-LINE TELEPHONE DIRECTORY ASSISTANCE INFORMATION RETRIEVAL SYSTEM

W. A. Hall
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey

Abstract

During the course of the development of the Directory Assistance System, it was desirable to study the expected performance of a wide variety of hardware configurations. Instead of creating a separate simulation program for each proposed design, it was decided to construct and implement a single, more general model. The model to be described is such that very little, if any, modification to the body of the program is necessary to describe all pertinent details of any given design to be simulated. The user need only modify the appropriate input parameters to experiment with a different configuration.

1. INTRODUCTION

A number of models have been constructed and implemented in GPSS/360 to simulate the performance of the Directory Assistance System (DAS), formerly known as the Semi-Automatic Information System (SIS). The early models were of a specific nature, representing systems using an IBM 360 central computer. As design considerations expanded to include the equipment of other manufacturers, the sheer volume of the data to be evaluated necessitated the development of a more general simulation model in which major changes in design configuration could be implemented with a minimum amount of modification to the GPSS program being necessary. This paper describes such a model and its implementation in some detail.

2. BASIC SYSTEM DESCRIPTION

DAS is intended to partially automate the handling of customer requests for directory information in large metropolitan areas. Studies have indicated that such a system should result in substantial financial savings, improved service, and better utilization of personnel over a period of time by increasing the rate at which customer requests can be satisfied.

When a customer desires directory information, his request is switched by an automatic call distributor (ACD) to one of several directory assistance bureaus (DAB's). Each DAB contains a number of operators, each of whom is seated at a console containing a keyboard and a cathode ray display screen. The customer supplies whatever pertinent information he may have available, such as last name and street name, business classification, etc. The operator then keys in certain portions of this information which will be transmitted over a communication line to the information retrieval center (IRC). The IRC contains a computer and several random-access storage devices on which the telephone directory is stored. A number of copies of the directory exist, each indexed by a different scheme, e.g., by list name and street name, list name and business class, etc. The particular copy which will be accessed for a given request is determined by the type of information supplied by the operator.

The computer locates the group of listings which match the search key determined from the operator's keyed-in initial request. If these listings occupy more than one "block" on the storage device, the operator is notified that a "long search" will be necessary. She may either direct the computer to proceed with the search by keying in a "long search demand," or she may choose to terminate the request and possibly try to obtain more information from the customer. If the listings occupy less than one "block" or if the operator has issued a long search demand, the listings are read into core storage. At this point an additional search is performed using the full text of the operator's request to further reduce the number of listings which might satisfy the request. The remaining listings are stored into alphabetical order on list name and formatted in groups of fifteen for output. (Fifteen listings represent one "page," i.e., a single CRT display). If fifteen or fewer listings remain after the final search, they are sent to the operator, who selects the one which satisfies the customer and gives him the information. If there are more than fifteen, all listings are written out, after formatting, on a random-access "page queue." Should the operator not find the listing she wants on the first display, she can key in a one-character page-flip request, which causes another page to be read from the page queue and sent directly to the CRT. As soon as the operator responds to the customer with the desired information, she becomes free to handle another call.

3. DESCRIPTION OF THE GENERAL MODEL

The design configurations which were under consideration for DAS, while different in detail and hardware characteristics, had enough in common overall to make desirable the development of a general model which could be made specific to any given configuration with a minimum amount of effort. The remainder of this paper describes the model which has been created to fulfill this need and its implementation in a GPSS/360 program. The model has the capability to simulate the behavior of systems involving multiple processors, multiple memory banks, and a great variety of combinations of random-access devices and channels. Figure 1 gives some idea of the range of designs which can be simulated. It should be noted that all previous work can be represented with the new model as well as the older, more specific models.

Figure 2 shows a somewhat more specific design which has been simulated with the general model. There are four memory banks, three of which can contain messages and listings, while the fourth contains the worker program. There are three processors, two of which serve as I/O dedicated processors to control the flow of data between the communication lines, the files, and the memory banks. These two can access only the three data banks. The third processor serves as a control/arithmetic unit and can access all four memory banks.

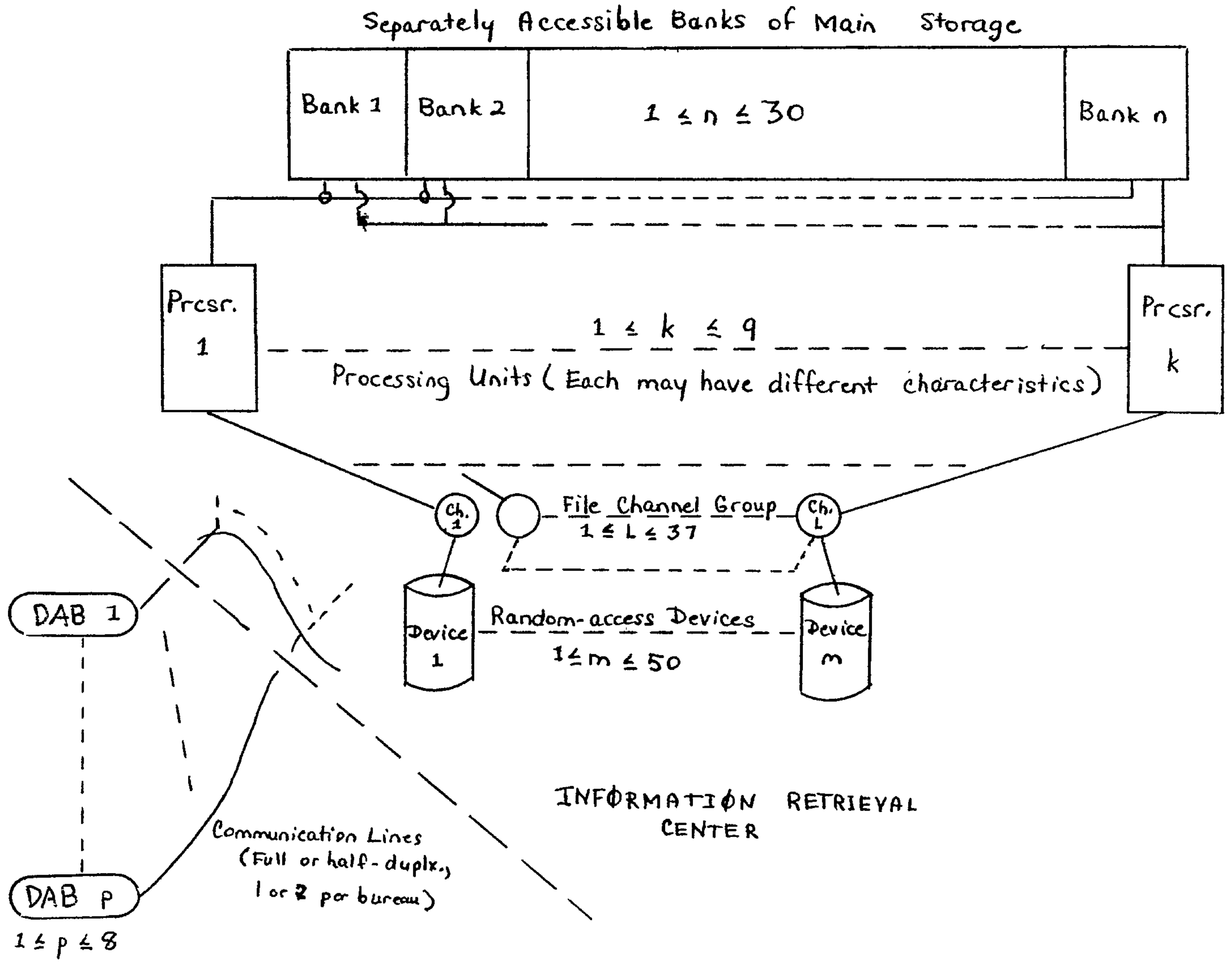
4. GPSS/360 REPRESENTATION OF SYSTEM COMPONENTS

The remainder of this paper requires that the reader have at least a minimal knowledge of the GPSS language. Most of the necessary concepts are explained briefly in the GPSS/360 Introductory User's Manual (IBM Publication #H20-0304-1).

Hardware items may be represented as "facilities" or "storages." The following representation is used in the general DAS model (see Figure 1):

(1) Facilities
- a. Communication lines
- b. Processors
  - Note: Each processor is represented in terms of six facilities; one master facility and five subfacilities, each of which represents the use of the master at a particular priority level.

(2) Storages
- a. Operators
- b. File devices
- c. Memory banks
  - Note: Each bank is represented as four storages. A more detailed description appears in Section 5.
- d. I/O-dedicated processors (multiplexors)



Separately Accessible Banks of Main Storage
Bank 1
Bank 2
1 ≤ n ≤ 30
Bank n
Prcsr. 1
1 ≤ k ≤ 9
Prcsr. k
Processing Units (Each may have different characteristics)
Ch. 1
File Channel Group
1 ≤ L ≤ 37
Ch. L
Device 1
Random-access Devices
1 ≤ m ≤ 50
Device m
DAB 1
DAB p
1 ≤ p ≤ 8
Communication Lines
(Full or half-duplx.,
1 or 2 per bureau)
INFORMATION RETRIEVAL CENTER

Fig. 1

A TYPICAL DAS CONFIGURATION

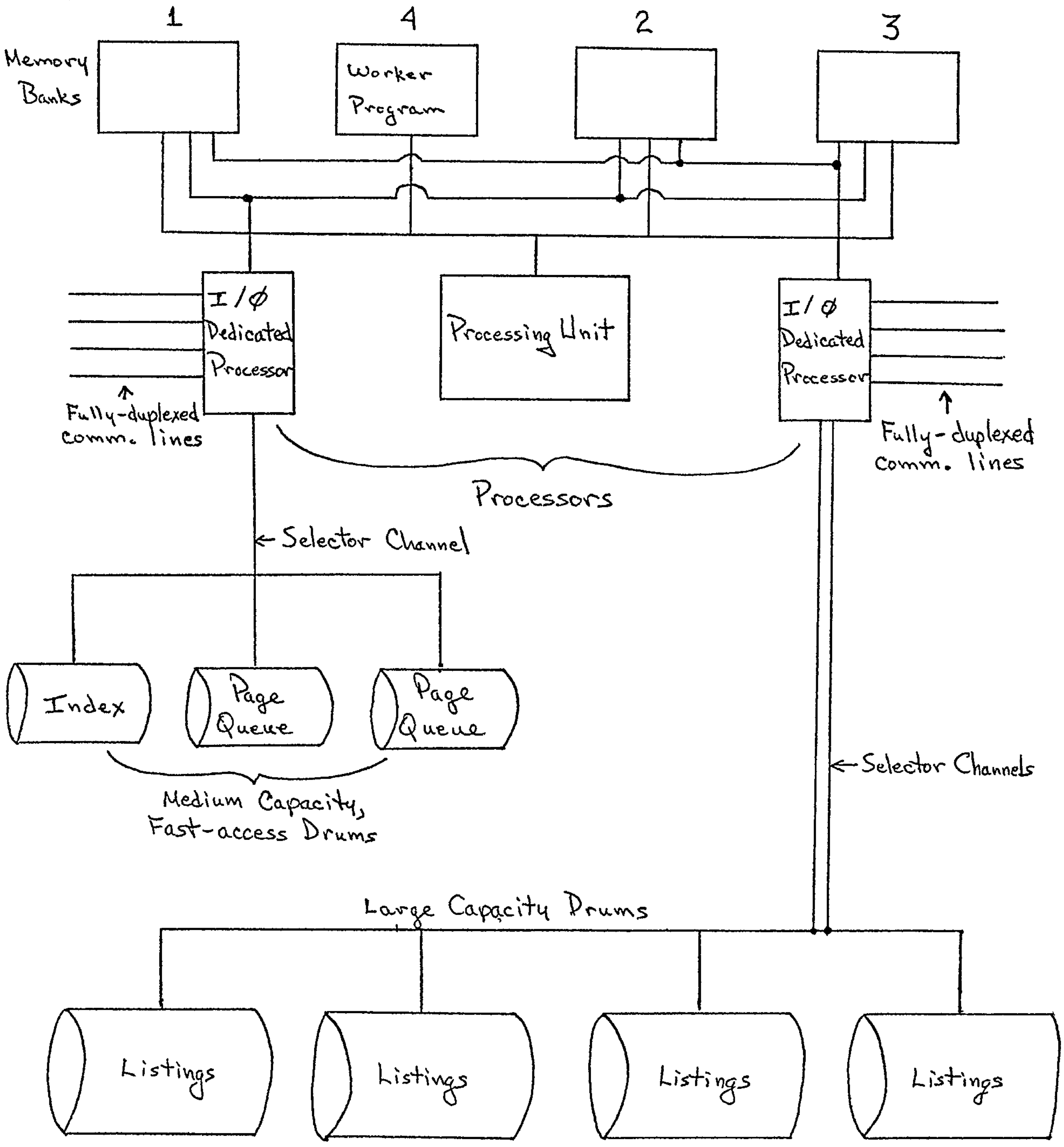


Fig. 2

e. Data channel groups; i.e., sets of channels connected to particular groups of file devices

f. DAB's

In addition, queues are located in front of all facilities and storages in order to provide needed statistical information.

5. GPSS/360 REPRESENTATION OF SYSTEM FUNCTIONS

This section will explain, primarily from a programming standpoint, the manner in which certain facets of the DAS have been modeled. The treatment is not intended to be a step-by-step description of program logic, but rather to isolate some of the less obvious methods used to represent various system functions and attempt to clarify them.

5.1 SELECTION OF MEMORY BANK

In a hardware configuration involving more than one separately addressable memory bank, it is necessary to decide just what the function of each bank will be. Each time an initial request is received, the general DAS model assigns to it one of the banks available for data storage. This bank is then used for all data associated with that particular request until the customer has been satisfied.

The development of an algorithm to choose a bank to assign to a request is complicated by the fact that each bank is represented by four consecutively numbered storages. Thus the use of the GPSS "SELECT" block is effectively ruled out. The method finally implemented involves a matrix half word SAVEVALUE (MH1) with a row for each of the thirty allowable banks and a single column. As many elements of this array must be INITIALized as there are memory banks available for data. The initialized values should be 60, 64, ... (60+(N-1)*4), where N is the number of data banks to be used for the particular configuration being simulated. (The value of N is supplied by the user as the initial value of a half word SAVEVALUE.) Note that this method requires that all banks used for data must be numbered sequentially smaller than those used only for worker programs.

Determination of the bank to be assigned begins by choosing at random a number between 1 and 30. This number is converted to a number between 1 and the number of banks available for messages. The latter value is used as a row subscript in MH1. If there is sufficient space available in the bank whose number was found in the indicated row of MH1, the number of the bank (i.e., the number of the first of the four storages which represented it) is ASSIGNed to a parameter of the transaction representing the initial request. If there was not enough available space in the bank just considered, the next bank in sequence is checked by the same process. Note that the "Next" bank may be either the one whose number is 4 greater than the previous bank, or else will be bank number 60 if the previous bank was the last one in the sequence of permissible banks. If all of the banks are checked in this way and none of them contain enough free space, the bank originally selected will be assigned.

5.2 COMMUNICATION LINES

In the general DAS model, 1 or 2 communication lines for each of the 1 to 8 operator bureaus (DAB's) may be simulated. However, all bureaus must have the same number of lines. In addition, the lines may be either fully-duplexed (able to transmit simultaneously two messages in opposite directions) or half-duplexed (only one message at a time). The user supplies the number of lines per bureau and whether or not the lines are fully-duplexed using half word SAVEVALUES. He also specifies which lines are attached to which I/O processors or multiplexors. Fully-duplexed lines are represented by one facility for each direction with a queue at each end on the line, while half-duplexed lines are represented by two queues and only a single facility.

The method of selecting the route over which a given message will travel between the operator and an I/O processor is as follows:

The queue for the inbound (from the operator) communication line is chosen first, based on the DAB number selected randomly by a GPSS variable. If only one line per bureau has been specified, the queue number is just the DAB number ($247 \leq$ DAB nr. $\leq 247 +$ number of bureaus-1 ≤ 254) minus 36. If there are two lines per bureau, the process is somewhat more involved. In this case, one queue number is computed exactly as in the one-line case, and another is computed by subtracting 28 instead of 36 from the bureau number. The current lengths of these two queues are compared, and the shorter one will be selected. The number of the line to be used is determined directly from the queue number. Note that if there are two lines per bureau, this method will result in the number of one line being 8 greater than the number of the other, instead of just one greater as might be expected. Once the line has been selected, the I/O processor or multiplexor to which that line is connected is determined directly from the line number.

For outgoing messages, the queue is determined simply by adding 16 to the inbound queue number. If the lines are half-duplexed, the line number is assigned as in the inbound case. If fully duplexed has been specified, the line number is 20 greater than the number of the input line.

5.3 MULTIPLE PROCESSORS

In any DAS configuration involving more than one processor, some scheme must exist for describing precisely just what functions will be performed by each unit. Before implementing the general model, three such schemes were considered:

(1) For each GPSS transaction (which represents an initial request), select one processor by some rule immediately after the transaction is generated and use it for all subsequent processing associated with that transaction.

(2) Dedicate certain processors to particular tasks, which they would perform on all transactions. For example, in a three-processor system, one processor might serve as an I/O controller; a second might be responsible for storage allocation and certain operating system functions, while the third might handle searching, sorting, etc.

(3) Each time the need for some task to be performed on a transaction arises, regardless of what the task may be, select one of the processors by some rule and use it to perform the task. The rule used should be one which considers all processors as potentially able to perform any task.

The method finally decided upon was a combination of 2 and 3. The use of method 2 per se was ruled out after considering the difficulty of specifying a processor for each of the many tasks that must be performed on any transaction. Method 3 per se causes difficulty in the case of I/O-dedicated processors. Such processors may be accepting information from several communication lines and/or file channels simultaneously; i.e., their GPSS counterparts may need to be occupied by several transactions at once. This precludes the representation of this type of processor as a GPSS facility, since only one transaction can occupy a facility at a given time. Consequently, the use of GPSS storages is suggested in this case.

By combining methods 2 and 3, the advantages of both can be realized. Processors dedicated to I/O are represented as storages, and are treated quite differently from control/arithmetic units. The user may place an upper limit on the capacity of these storages which corresponds to the maximum rate of data flow which the processors can handle in thousands of bytes per second. All other processors are represented as facilities which are preempted at various priority levels, as will be explained later.

The particular I/O processor to be used at a given time is determined solely by the hardware configuration; i.e., once a particular communication line or file device has been selected, that processor to which the line or device is physically connected will be the one used. Selection of a control/arithmetic processor is accomplished by Method 3, by using the GPSS SELECT block to choose the processor with the least number of tasks waiting for its service.

5.4 PRIORITIES AND SUBFACILITIES

By using the GPSS PREEMPT block with the PRiority option, it is possible to, in effect, make the GPSS system serve as an interrupt handler. The current model allows up to five different priority levels at which any control/arithmetic processor may be PREEMPTed. The facilities which are PREEMPTed are those which are selected as described in the preceding section. In addition, each major facility has associated with it five "subfacilities" which are used to keep track of the usage of each processor at each of the five possible priority levels. Immediately after a major facility representing a processor is PREEMPTed, the appropriate subfacility is SEIZEd, with the number of the latter facility being determined from the priority of the entering transaction and the number of the major facility, which is found in parameter 23 of the transaction.

Priorities are assigned to transactions on the basis of the next task to be performed on them. The assignment is made with the GPSS PRIORITY block, using the BUFFER option. A complete description of the function of this option is beyond the scope of this paper. Basically, however, its function is to insure that any other transactions already waiting to PREEMPT a given processor at a priority greater than or equal to the priority which the PRIORITY block is assigning will be processed before the current transaction. This should be recognized as a standard technique for handling interrupts.

5.5 I/O AND CYCLE STEALING

This is a rather complex topic, particularly when several processing units are involved. In order to get the model into operation, several simplifying assumptions had to be made. Just how accurate the representation thus obtained actually is has not been determined exactly at the present time, particularly in the case of several processors, but it is thought to be a reasonably good approximation.

Basically, the problem revolves around the fact that, at a given time, data can be flowing between a given memory bank and several I/O sources. For example, various data associated with a particular bank may be coming in over one communication line, out over another, in from one of the listing files, and out onto the page queue simultaneously. If one or more control/arithmetic processors happen to be operating on data in that bank at the same time, the associated processing times must be increased to compensate for processor cycles demanded by the I/O operations. Clearly, the more I/O activity occurring at a certain time, the more cycles that must be stolen.

The actual computation in the DAS model of the number of cycles to be stolen is based on probabilistic concepts. The user supplies the percentage of processor cycles that involve memory accesses in the bank containing the message being processed, the number of cycles required for the transmission of a file word and a communication character, and the number of memory kilocycles per second. The model keeps track of the flow in and out of each data bank, maintaining this quantity in the first of the four GPSS "storages" associated with each bank. Each time a transaction calls for data transmission, the model computes the number of cycles to be stolen by means of a probabilistic equation involving the above quantities and the number of characters to be transmitted. The number thus determined is maintained in the fourth of the four storages representing the bank.

At this point, an important simplifying assumption is made. Instead of actually representing the stealing of a single cycle at a time, which would create major programming difficulties, it has been assumed that no drastic effect on the overall results of the simulation would be noted if the required cycles were stolen in blocks instead.

To accomplish this, a test is made on the contents of the fourth storage representing the bank in question. Whenever the contents exceed the number of cycles in one simulation clock unit (.1 msec), a transfer is made to a routine which interrupts simultaneously all processors currently demanding cycles in the bank for an amount of time computed from the contents of the storage and the number of memory kilocycles per second. The matter of determining which processors are active in which banks is taken care of by the routine SAVB, which is called every time processing of any kind is requested. This routine continually updates a 9 x 5 matrix SAVEVALUE (one row for each possible processor and one column for each priority level) which maintains a record of the banks in which processors are active.

6. CONCLUSION

The general model of the DAS which has been described above enables the user to simulate a wide variety of hardware configurations with very little, if any, modification to the body of the GPSS program. The user can describe virtually all pertinent system characteristics by means of input parameters. This model was able to provide much information on the various designs under consideration in a relatively short time and with a minimum of additional programming effort. Pertinent statistical data obtained included the utilization of processors, memory banks, file devices, channels, and communication lines; average and maximum I/O cycle rates, average and maximum contents of file and communications buffers, average and maximum number of transactions in the system, queueing data, and distributions of system response times. The flexibility of the model allowed six different runs involving three major configurations in ten working days.

Mr. Hall is currently a Member of the Technical Staff in the Programming Research and Development Department at Bell Telephone Laboratories, Incorporated, in Holmdel, N.J. The majority of his work has been in the areas of machine aids to design and the simulation of communications systems. He holds a B.A. in mathematics from Lehigh University and an M.S. in computer science from Purdue University.