This paper discusses the designing of complex teleprocessing systems using a discrete simulation modeling tool, the Systems Network Analysis Program/Simulated Host Overview Technique (internally and informally called SNAP/SHOT). This modeling tool aids in designing computer communications systems composed of local and remote terminals, teleprocessing lines, host processors that control the teleprocessing lines, and interconnected communications systems. The model is capable of analyzing both tree- and mesh-structured networks.

Performance analysis of complex communications systems

by H. M. Stewart

The large teleprocessing systems being designed today frequently involve thousands of terminals, many different applications and software subsystems, and more than one host processing system. The performance of such a system can affect the entire operation of a business or other organization. Therefore the initial design and any subsequent changes must be done with assurance that the system will meet the designer's expectations.

Determining the performance characteristics of a new teleprocessing system by trying it out usually is too costly, in terms of both time and money, to be practical. Reconfiguring a system to measure the performance of possible design alternatives is generally impractical, since telephone lines may have to be rerouted and hardware or software changed in many locations over a large area. Such changes cannot be made quickly or inexpensively.

To give some reasonable assurance that a system, once installed, will give the performance desired, it is necessary to be able to predict the response times and loads imposed on all system components, given certain traffic characteristics. It is also useful to be able to determine which components of the system will become limiting factors as the workload changes, so that management can arrange for adequate resources as needed. Insight about which components are critical to system performance is valuable in anticipating problems and alleviating bottlenecks before the performance becomes unacceptable.

There are several techniques for doing system design analysis of installed systems. It is possible to use a teleprocessing driver to increase the load on a system. However, in using a driver it is necessary to have the hardware available. It is possible to monitor the installed system and measure the central processing unit (CPU) with software monitors, but little or no information is provided on the performance of the network. In addition, both of these techniques require that the programs and hardware be available and running at the time of the analysis.

To answer questions about response time, system load, and performance in general, a modeling tool called the Systems Network Analysis Program/Simulated Host Overview Technique (internally and informally referred to as SNAP/SHOT) was developed at the IBM Marketing Support Center in Raleigh, North Carolina.[1] The tool is a discrete simulation model used by IBM systems engineers, along with the customer personnel involved, to model computer communications systems that include local and remote terminals, teleprocessing lines, remote programmable clusters such as the IBM 8100 Information System[2] and 3790 Communication System,[3] and host processors for network control. It is used for predictive analysis of complete teleprocessing systems. Results of the model are used by IBM marketing teams and customer management to anticipate the effects of the growth of current systems and the installation of new systems. (The output may not be retained by non-IBM personnel).

The goal in developing the simulator was not to produce just a modeling language, but to provide the user with a program he could use by simply describing his system in terms of hardware, software, and system traffic load without being concerned with the internal workings of IBM software and hardware. Sufficient material is available for one who wishes to write a simulation model to describe the internals of computer systems.[4,5] The users for whom the new technique was developed, however, are data processing professionals whose interest is not in writing a model, but in designing a system consisting of standard available components without having to develop the tool to do the analysis.

This discussion of performance analysis of communications systems begins with a brief overview of the modeling tool. The overview is not intended to provide a complete tutorial on the language used, but to orient the reader as to the level of information required to provide input to the model. The overview is followed by a typical example of the model's use in evaluating design alternatives for a corporation's teleprocessing system. The example is based on an installed system. Possible designs are evaluated in light of the anticipated growth of the corporation, and a final design is selected, based on the results of the model.

Overview

The modeling program consists of a preprocessor, a simulation model, and a postprocessor. The preprocessor accepts macro input statements that describe the system to be modeled, and it generates the input required by the model. Examples of these statements are given in Figures 1 through 6. The preprocessor also sets up the interface between the model and the postprocessor, so that the postprocessor can control the model's execution and print the intermediate and final simulation results.

The simulation model has two major components, a network part and a host part. The network part is concerned with representing the teleprocessing lines, the communications controller, the terminals, and the flow of messages through those components. The host part of the model is concerned with predicting CPU utilization and host response times based on number of instructions executed, interrupts, software subsystem scheduling algorithms, priorities, and CPU types. Multiple host processors can be modeled in one simulation run. The network part and the host part of the model can be used independently of each other if the concern is strictly network or host related, or they can be used simultaneously to model a complete system.

The model provides information about the performance of different hardware and software alternatives to handle a particular anticipated workload. As systems grow in numbers of terminals and message volumes, and as new applications are added and processing requirements increase, the model output will show which component needs attention in order to handle the additional workload. This data, in conjunction with cost and benefit dollar figures, allows design alternatives to be compared at each stage of the growth process.

Message definition

As with all communications systems, the amount of message traffic to be handled by a teleprocessing network determines the load on all components of the network. Therefore every simulation with the modeling program requires that messages be described. Messages sent from a terminal to a host computer are described by MSGIN (message-in) macros. Similarly, messages or responses sent to terminals from application programs in a host processor are described by MSGOUT (message-out) macros. When any of a set of different messages is sent from a terminal, the set is defined by a MSGINMIX (message-in-mix) macro followed by each of the MSGIN's in the set. Each message definition in the MSGINMIX must specify a percentage that indicates the probability of occurrence of that particular message—that is, its frequency relative to the other MSGIN's in the mix.

Figure 1 Example input messages

```
MIX01   MSGINMIX
        MSGIN     LENGTH=50,PERCENT=25,CPUDELAY=.5,MSGOUT=MENU,...
        MSGIN     LENGTH=40,PERCENT=50,CPUDELAY=1.25,MSGOUT=SCRN1,,
        MSGIN     LENGTH=60,PERCENT=25,CPUDELAY=2.5,MSGOUT=FINAL,..
        ENDMIX

MENU    MSGOUT    LENGTH=250,DEST=ORIGIN
SCRN1   MSGOUT    LENGTH=725,DEST=ORIGIN
FINAL   MSGOUT    LENGTH=250,DEST=ORIGIN
```

Figure 1 illustrates an example in which three input messages have been defined within the MSGINMIX labeled MIX01. The definitions specify the message length in characters (LENGTH), the probability of occurrence (PERCENT), and the time in seconds that each message spends in the host system once it arrives there (CPUDELAY). The appropriate output message (MSGOUT) will be generated and transmitted back to the originating terminal (DEST=ORIGIN) after the specified time has elapsed. The example assumes an inquiry/response situation, in that one message goes from a terminal to the host, and one response goes back to the terminal.

Facilities are available to specify to the preprocessor many different message flow variations. Not all message traffic can be characterized as inquiry/response messages. For example, in a data entry environment, most traffic is input to the host, and no response is necessary to the input messages. And in a remote job entry (RJE) environment, batches of input may be sent to the host, and unrelated batches of output may be sent from the host to an RJE terminal. Even in an inquiry/response situation, it is not unusual for the output to consist of data sent to a nearby printer, as well as a response to the originating terminal. All such combinations of message traffic can be represented in the model.

The network description

The performance of a network is influenced by the network topology, the line discipline that controls the flow of data through the network, and the message rate, as well as by the message characteristics described above. The simulator can model all terminals and clusters in an SNA (Systems Network Architecture) environment,[6] as well as most IBM binary synchronous and start-stop protocol devices.

Figure 2 Preprocessor input required to define a multipoint line

```
WESTPA LINE    TYPE=3276,CHARSEC=600,MILES=500
       CLUSTER MSGRATE=150,MSGINMIX=MIX01,SESSIONS=4,...
       CLUSTER MSGRATE=250,MSGINMIX=MIX02,SESSIONS=6,...
       CLUSTER MSGRATE=330,MSGINMIX=MIX01,SESSIONS=3,...
```

In Figure 2, a multipoint line identified by the label WESTPA is defined with a LINE macro, and the primary terminal type on the line is an IBM 3276 Control Unit Display Station.[7] The three lines that follow give the preprocessor input required to define the network.

The protocol for this line is determined by the type of terminal. In this case it is an SDLC (synchronous data link control) line.[8] The line speed is specified by coding the data transmission speed in characters per second (CHARSEC), or by coding the baud rate of the line. The 600 characters per second specified is equivalent to a 4800-baud line, since there are 8 bits per character in the SDLC line discipline. All necessary data link control characters will be added to the message length automatically by the model as a function of the line discipline and terminal type. The length of the communications line WESTPA is shown as 500 miles. This distance is used to compute the time required for data and control information to travel from the cluster to the host and back again.

Each 3276 Control Unit connected to the line being defined must be represented by a CLUSTER macro immediately following the LINE macro that defines the line. Figure 2 shows three of them attached to the line WESTPA, each with its individual characteristics coded. The MSGRATE parameter indicates the rate, in messages per hour, at which messages should be generated for the cluster. MSGINMIX indicates the set of messages that the terminals attached to this cluster are eligible to send, as described in the preceding section. Finally, the number of devices attached to the controller is indicated by the SESSIONS parameter. The devices can be either display stations or printers. When this model is run, the following activities take place:

- The simulator generates messages for each cluster at the hourly rate requested. This rate is used as the mean of an exponential interarrival function. (Users may define their own message arrival functions if an exponential interarrival time distribution does not represent the behavior of the cluster or device in question.)
- Whenever it is time to generate a message, the simulator selects the appropriate MSGINMIX and randomly chooses one of the messages in the set. The distribution of messages generated is determined by the value of the PERCENT operands. (See Figure 1.)

- The generated message is transmitted to the host, following 3276 SNA protocol. The message is delayed in the network at the appropriate points; for example, the message waits until the cluster is next polled, then the text is transmitted down the line, with any necessary control information.
- The CPUDELAY specified in the MSGIN is invoked, representing the time necessary to process the transaction in the host. If the host is being modeled in detail, application program activity is simulated at this point, instead of just a time delay.
- When the CPUDELAY time has expired, the output message identified by the MSGOUT operand of the MSGIN is created and returned to the originating terminal. Once again, appropriate delays in the network are reflected by the model.

The preprocessor accepts many network input specifications that are directly comparable to parameters the user would code to generate the network control program (NCP), which runs in the 3705 Communications Controller.[9] Some of these parameters significantly affect the response time and throughput of the network. The model provides an easy way to determine the best combination of parameter values to handle the particular types of traffic in the network.

The information required for the lines, or trunks, that connect host processors is similar to the information provided for lines between terminals and a host processor.

The host model

The host model consists of tables of instructions used by specific IBM software functions, a priority interrupt mechanism, a priority dispatcher, and the CPU or CPU's defined by the user in the preprocessor input. All System/370 CPU's can be modeled. The characteristics of the processor to be modeled are defined with a CPU macro.

To use the host model, it is necessary to describe the processing requirements imposed on the host system by each type of message that enters the host. Thus each MSGIN macro for a detailed host model has a parameter that specifies the application processing to be done when the message arrives at the host. This parameter (APPLPROG) identifies a macro that describes the application work to be done. One APPLPROG macro is required for each application program to be modeled.

Figure 3 shows what the input would look like for a processor running an on-line application with three types of messages to be processed. Since the processing requirements for each message are different, each message points to a unique APPLPROG macro

Figure 3 Input for an on-line application with three types of messages

```
DETR     CPU      TYPE=158,DETAIL=YES,OS=OSVS1,TP=VTAM
*
INVOICE  APPLPROG DBDC=NONE,APPLEXEC=35000,BDAMRDIO=3
LINEITM  APPLPROG DBDC=NONE,APPLEXEC=55000,BDAMRDIO=2
SUMMARY  APPLPROG DBDC=NONE,APPLEXEC=97000,BDAMRDIO=17
*
MIX01  MSGINMIX
       MSGIN    LENGTH=50,PERCENT=25,APPLPROG=INVOICE,MSGOUT=GETITM
       MSGIN    LENGTH=40,PERCENT=50,APPLPROG=LINEITM,MSGOUT=PRTTOT
       MSGIN    LENGTH=60,PERCENT=25,APPLPROG=SUMMARY,MSGOUT=NEXTIN
       ENDMIX
*
GETITM MSGOUT  LENGTH=250,DEST=ORIGIN
PRTTOT MSGOUT  LENGTH=653,DEST=ORIGIN
NEXTIN MSGOUT  LENGTH=900,DEST=ORIGIN
```

which defines the work to be done. This example assumes a System/370 Model 158 running under OS/VS1 and the Virtual Telecommunications Access Method (VTAM).[10]

The specification DETAIL=YES indicates that the host model is to be activated for this particular CPU. For each of the three application programs defined with APPLPROG macros, a certain number of APPLEXEC instructions are to be executed, and a certain number of Basic Direct Access Method (BDAM) file reads (BDAMRDIO) are to be done.[11] The notation DBDC=NONE specifies that these programs are not running under control of an IBM-supplied software subsystem.

The model's execution of the BDAMRDIO macro will include the operating system's instructions to initiate a disk I/O operation and check for errors on completion of the operation. Program execution is suspended while the disk I/O operation is in progress. There are similar operands for other access methods used in on-line applications, including the Indexed Sequential Access Method (ISAM), the Virtual Sequential Access Method (VSAM), and Data Language/I (DL/I).[11,12]

Software and hardware monitoring statistics are the most useful for defining the application program's execution requirements. Software monitoring facilities are available for programs that run under the control of IBM data communications program products.[13] For applications that have not yet been implemented, enough of the system design must be completed so that the number of instructions and the number of disk I/O operations can be estimated.

Figure 4 Examples of an application program labeled INVOICE

```
Example A -- INVOICE APPLPROG APPLTIME=1.5


Example B -- INVOICE APPLPROG APPLEXEC=125000,


Example C -- INVOICE APPLPROG APPLEXEC=90000,
                              BDAMRDIO=2,
                              BDAMWRIO=1
```

Model output

The output report consists of statistics obtained during the simulated time. As messages are being simulated, the program keeps track of the time when each message enters and leaves each part of the modeled system. At the end of the life of the message, therefore, the total time, network time, 3705 time, and CPU time can be calculated. In all, nine response times are calculated for each message. Also, as messages wait in queues, the model keeps track of average time in the queue, average length of the queue, and maximum queue length. With such a large number of statistics available, identifying the source of a performance bottleneck is simplified.

The basic output report consists of the following information, as appropriate:

- System summary of the nine response times by message type.
- CPU and 3705 message counts.
- Detailed host CPU statistics by application program, by priority level, and by function.
- Detailed statistics by application program and software subsystem for wait time, CPU utilization, and queues.
- Detailed I/O statistics including channel utilization, simulated I/O response time, and contention between disk files.
- 3705 message counts, buffer utilization, and CPU utilization.
- Statistics for 3705-to-3705 trunks.
- Detailed network information for each line and terminal, including response time, line utilization, wait for poll, and wait for line, as well as counts of messages by message type.

Levels of detail

The model is designed so that the user can define the system to be simulated at various levels of detail. Five examples, presented in Figures 4 and 5, describe an application program labeled INVOICE. Each succeeding example requires more time for data collection, input definition, and model running time than the preceding one.

Figure 5 Example application program showing high-level logic flow

```
Example D -- INVOICE   APPLPROG APPLOGIC=INVLOGIC
             INVLOGIC  APPLOGIC
             LOOP      EXECUTE   30000
                       BDAMRDIO  1
             RP1       REPEAT    LOOP,1
                       EXECUTE   30000
                       BDAMWRIO  1
                       ENDLOGIC
```

Figure 6 Example application program with file names and units defined to reflect actual behavior of channels and devices

```
Example E -- INVOICE   APPLPROG APPLOGIC=INVLOGIC
             INVLOGIC  APPLOGIC
             LOOP      EXECUTE   30000
                       READ      ORDER
             RP1       REPEAT    LOOP,1
                       EXECUTE   30000
                       WRITE     ORDER
                       ENDLOGIC
             ORDER     FILE      UNIT=SYSDS,TYPE=BDAM
             SYSDS     UNIT      CHANNEL=01,TYPE=3350
```

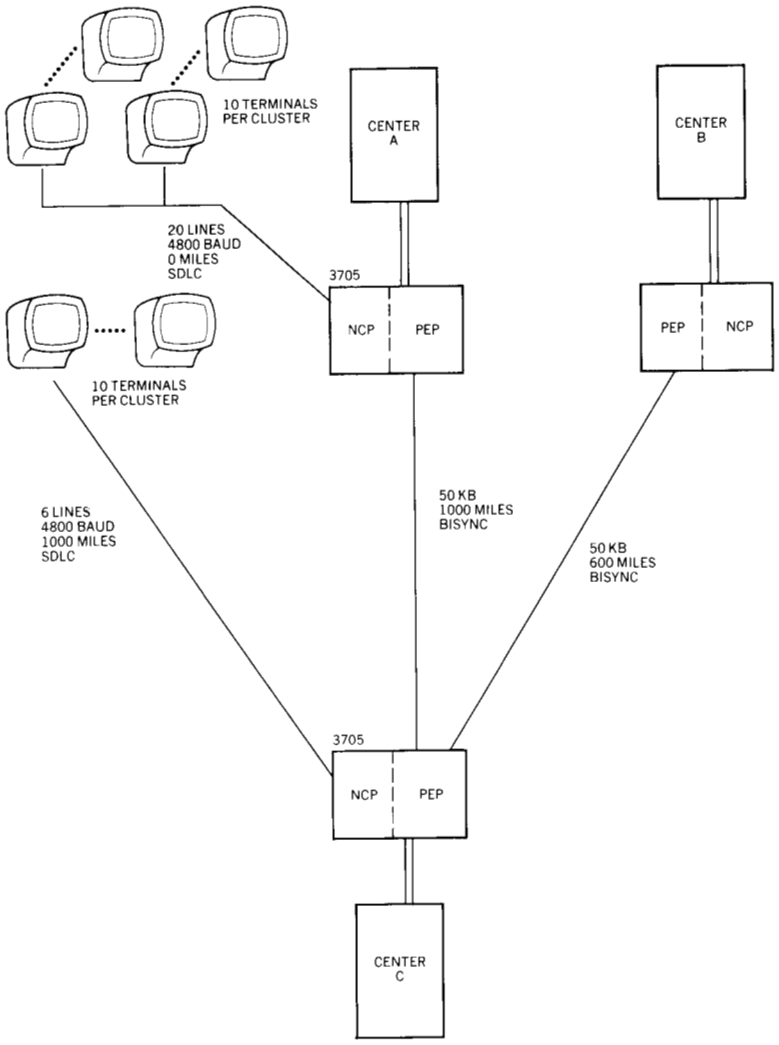
The more detailed the input, the more information can be gained from the result—but note that the additional information is not always necessary or justified. In Figure 4, Example A indicates a delay time of 1.5 seconds and would be appropriate if the user did not require detailed simulation of the host CPU.

Example B indicates that the application program consists of 125 000 executable instructions and that three direct-access-storage-device I/O's (EXCP0=3) are required to process a message.

Example C indicates that the program consists of 90 000 executable instructions, two BDAM reads (BDAMRDIO), and one BDAM write (BDAMWRIO). The BDAM parameters have an associated I/O delay time and associated instructions, the number of which depends on the type of system control program.

In Figure 5, Example D reflects the high-level logic flow of the INVOICE application program. The APPLOGIC entry in the APPLPROG macro points to a series of preprocessor input statements that define a sequence of operations to be performed. The series is terminated when an ENDLOGIC statement is encountered. The EXECUTE statement specifies the number of instructions to be executed. BDAMRDIO and BDAMWRIO are as described above.

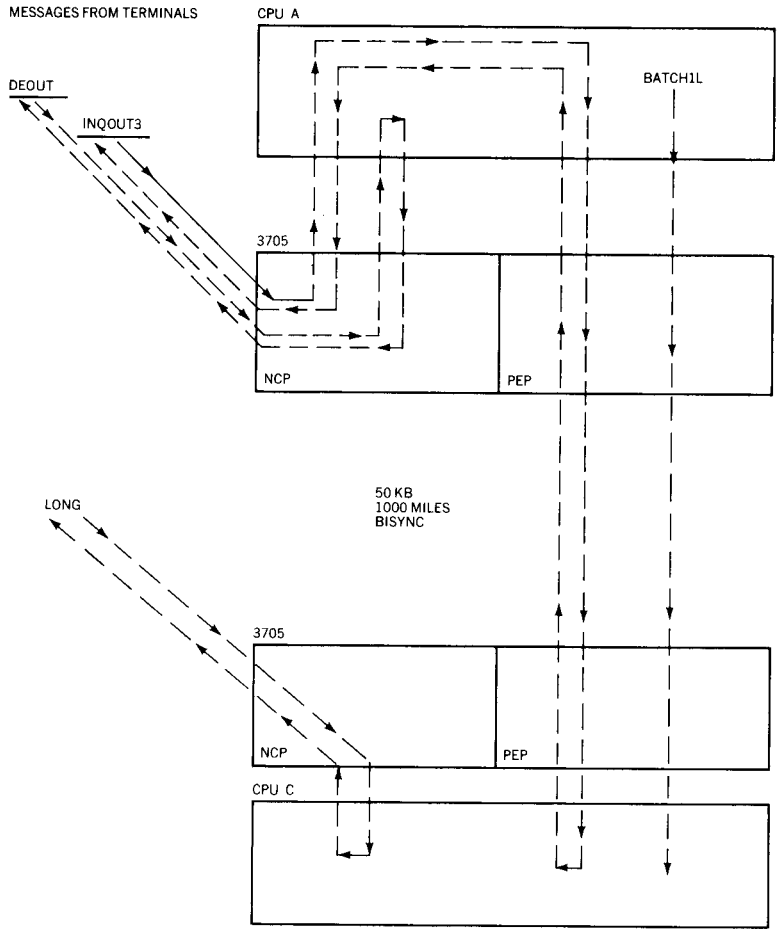
Figure 7 Current configuration



REPEAT is a logic-flow control statement which allows part of the APPLOGIC to be re-executed. In this case, REPEAT causes the instructions between LOOP and RP1 to be executed one more time, for a total of two executions of the BDAMRDIO statement.

Example E, in Figure 6, is similar to Example D, except that file names and units are defined to reflect the actual behavior of channels and devices. The READ and WRITE statements define the type of operation to be performed, as well as the name of the file to or from which data is to be transferred. The FILE macro defines the organization of the data (TYPE=BDAM) and, with the UNIT statement, the physical characteristics of the hardware in which the data resides. In this case, the UNIT is an IBM 3350 direct access

Figure 8 Message flow in current configuration



storage device attached to the CPU via channel 1 (CHANNEL=01). The representation of I/O devices does not explicitly include control-unit or head-of-string simulation.

All the above examples show only a small portion of the input macro language. Additional information is available in the documents cited in Reference 14.

The design process

An example of how the model is used, based on an installed data processing system, illustrates one of the many design questions that must be answered in a complete teleprocessing system analysis. The installation has three data centers, here called A, B, and

C, as illustrated in Figure 7. Each center has a local network, as shown in Figure 8, for processing data maintained only at that center. In our example, the related messages are called DEOUT.

There is a need for some of the terminals attached to centers A and B to access data at the headquarters center, C. The related messages are called INQOUT3 in our example. In addition, there is one application that is available only at C, so there are terminals near A and B that have dedicated teleprocessing lines attached to center C, 1000 miles away. Messages relating to that application are called LONG. Currently there are 50 000-bit-per-second lines between A and C, and between B and C, which are used for transferring data in batch mode. The data is transferred in batches of 80 000 characters, with approximately 15 seconds elapsing between successive batches. In our example, messages of this type are called BATCH1L. These same lines are available for transmitting INQOUT3 messages between batches.

All the terminals currently installed are IBM 3274 Display Units[7] which operate with VTAM through an IBM 3705 Communications Controller with the network control program (NCP). The lines between the centers operate with a bisynchronous line protocol, with a BTAM program controlling the lines, through the same 3705 controller with the Partitioned Emulation Program.[9]

The objectives of the proposed installation were to provide the following improvements over the then current configuration:

- Decrease network costs.
- Provide improved response time for INQOUT3 messages.
- Maintain the same or better BATCH1L throughput.
- Allow for traffic increase with minimal performance degradation.

To correctly design a system to meet those objectives, it was first necessary to identify the factors that most affected system performance. The amount of data processed on the host processors in the three centers would not change as a result of network reconfiguration, since the data traffic would remain the same. Therefore it was not necessary to gather data to characterize CPU utilization or the I/O activity generated by each type of message on the host systems. Since the analysis concerned the communications system, however, it was necessary to collect as much data as possible on traffic volumes, message lengths, response times, and terminal configurations. Because the traffic at centers A and B was almost identical, it was decided to examine only centers A and C.

Once the data was obtained, it was possible to create a model of the installed system and calibrate it with the system's known per-

Figure 9 Results of initial model runs

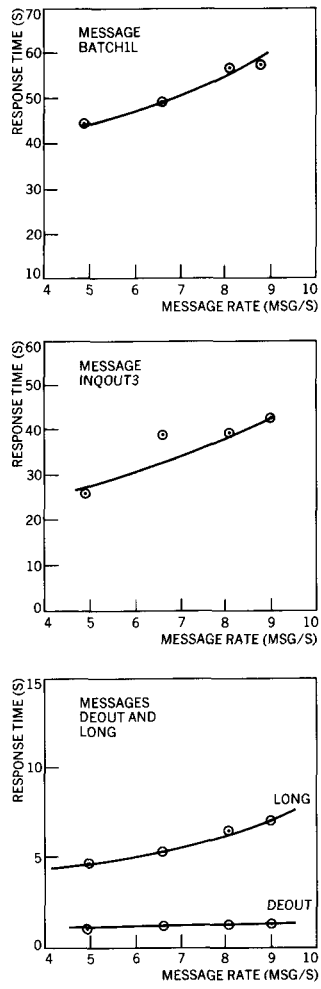
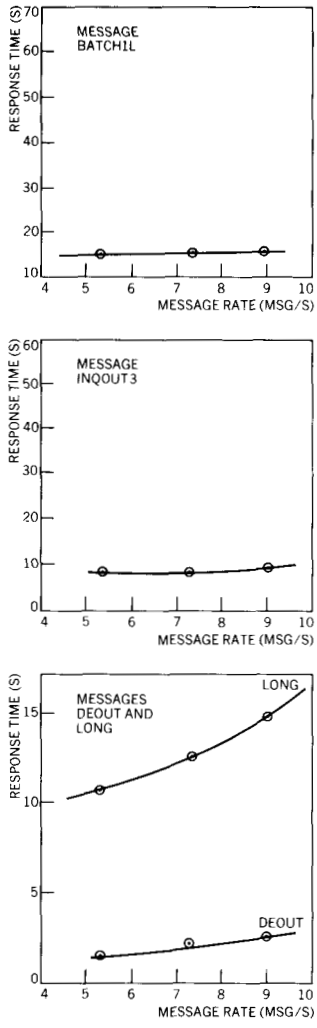


Figure 10 Results of first SNA model



formance. It is imperative that the model results be calibrated and compared with the data in the measured system. Measurements of such parameters as traffic rates and response times must be compared with the model's output, and any differences must be resolved to ensure that the model is an accurate representation of the real system. The system can be modeled in sufficient detail that any discrepancies between the measured data and the model can be quickly identified and the cause of the differences reconciled.

The system was modeled over a range of traffic rates, so that the performance of the model could be compared with the performance of the real system for different times of day, when traffic characteristics are different. Results of the model are depicted in Figure 9, in which response times are plotted against the aggregate message rate. The model compared very well with data obtained from the installed system. The response time for BATCH1L is defined as the time required to transmit the whole message in one direction, plus the time required to send a response back to the sending center. For the other messages, the response time is defined as the time between depressing the *enter* key at the terminal and the arrival at the terminal of the last character of the response.

These results showed that as traffic increased, the response times of both BATCH1L and INQOUT3 increased dramatically. The DEOUT traffic, which did not use the center-to-center lines, showed no appreciable increase in response time. Line-utilization and wait-for-line-queue statistics from the model showed that the lines between the centers were being used at maximum capacity, so the traffic on those lines was delayed because of the queuing effect. Thus the limiting factor was seen to be the center-to-center line, so design efforts were made to improve that portion of the system.

first SNA model

Based on the first set of model runs and the stated performance objectives, several design changes were considered. First, the lines between the centers were attached to the NCP and run as full-duplex SNA links, allowing data to be transmitted in both directions simultaneously. This change, which reduced line utilization, was accomplished in the model merely by changing the line type on the TRUNK macro. If the modeled performance proved to be substantially better, the BTAM application could be rewritten to use VTAM.

It was decided, too, that terminals with dedicated lines to the headquarters center C could, by taking advantage of the line sharing allowed by VTAM, also use the SNA trunk, thereby saving line costs. Implementing this change in the model required a change in two input statements to reflect the new type of protocol used on

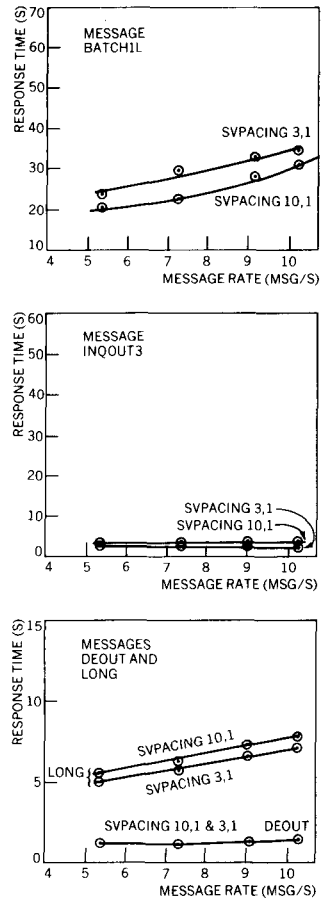
the center-to-center lines, and it also required moving four input statements so that the lines attached to center C could be moved to the closest 3705. The results of this change are reflected in Figure 10.

As we reviewed these results, we discovered several surprises. The response times for BATCH1L and INQOUT3 improved, as expected, because of the full-duplex SNA protocol, but the response times for LONG and DEOUT became longer. Model output showed that the NCP had run out of buffers, a condition called "slowdown." When slowdown occurs, the NCP stops polling lines until enough data has been transferred out of the 3705 to free the required number of buffers. The NCP enters slowdown mode when a user-specified minimum percentage of buffers remains. The NCP will stay in slowdown mode until a larger user-specified percentage of buffers is available. This NCP slowdown caused the longer response times for the LONG and DEOUT messages.

Slowdown can be alleviated either by adding more storage to the 3705 or by limiting the amount of data entered into the 3705 by the batch transfer application. We chose the latter solution for economic reasons. Data entry can be limited in an SNA environment by pacing, a method of controlling the rate at which data is transferred between components of an SNA system.[15] For example, the specification SVPACING=(3,1) causes the sending program to send only three frames of data, then wait for a response from the receiving program before sending another three frames. The receiving program must signal the sending program that it is ready to receive another batch of data. The second number in the (3,1) definition indicates that the receiving program will give such a signal when it has received the first frame.

By implementing SVPACING, the 3705 will receive only three frames of the 80 000-character message at a time, so the buffer requirements will be reduced. The performance implications of SVPACING are that batch data will not be transferred as fast because of the responses required, but the response time for INQOUT3 messages is better because they have to wait for only three batch frames to be transmitted, not for the whole 80 000-character message.

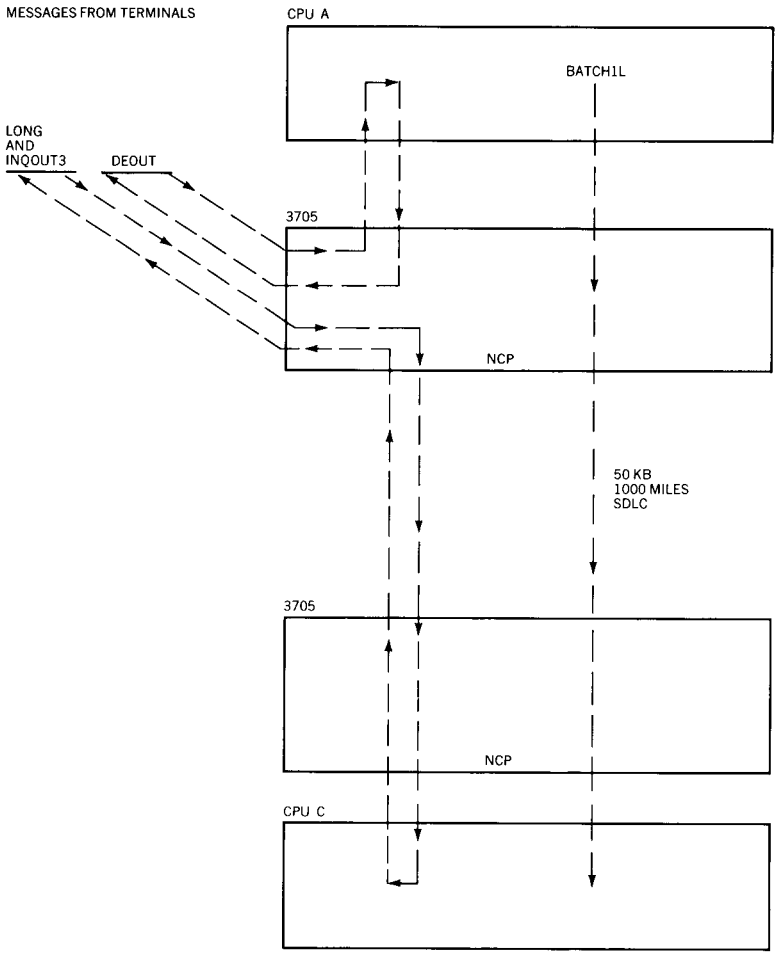
Figure 11 Results of final SNA model



final SNA design

SVPACING was implemented, and the results met all the performance objectives. An additional run was made with SVPACING=(10,1) to determine the effect on interactive work if batch transmission was optimized. This change improved batch performance considerably compared with SVPACING=(3,1), but, as expected, response times were slower for the LONG and INQOUT3 inquiry messages, which also used the trunk. The decision as to which value to use depends on installation priorities. The results are shown in Figures 11 and 12, and the associated configuration is illustrated in Figure 13.

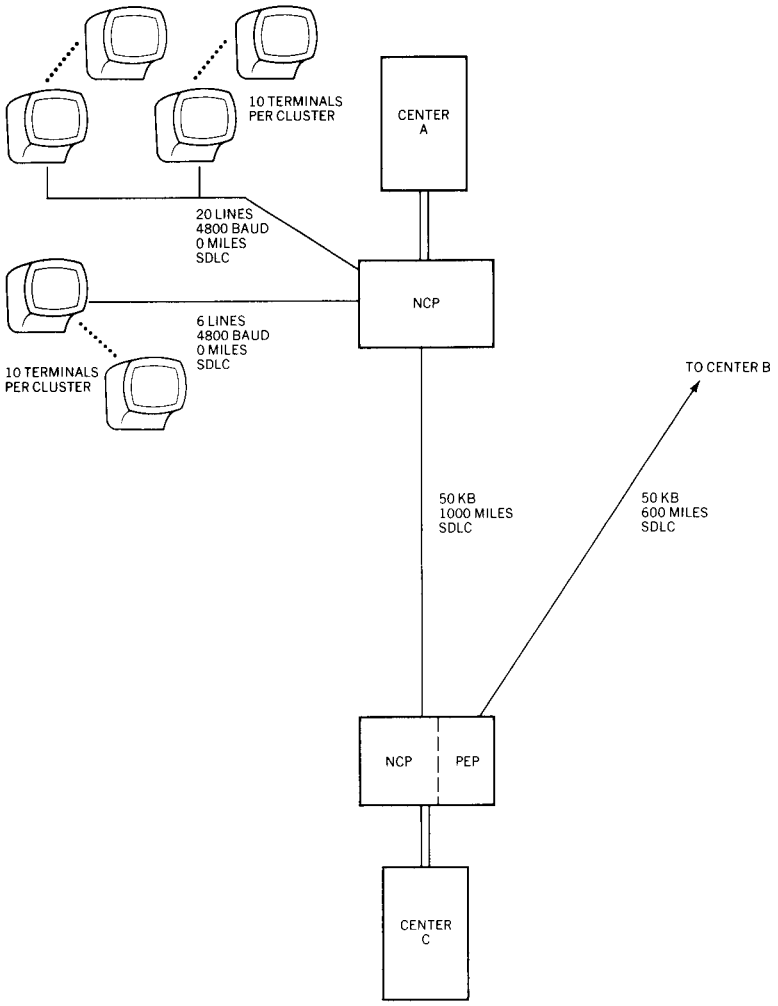
Figure 12 Message flow in proposed configuration



As can be seen in Figure 11, the BATCH1L response time increased, but it is still well below that in the initial case (Figure 10). The response time for LONG is somewhat greater because the message must share the line with the batch messages, but the difference is not so great that it is of concern. All other response times are the same as or better than in the initial case. Also, line costs are reduced by eliminating the special lines to center C for one application. The system performs well in terms of response time, even in the ten-message-per-second range, which is well beyond the installation's expected traffic growth.

Several other possible designs were considered. One was simply to install a remote concentrator for terminals that were far away from center C. This solution would have cost more, however,

Figure 13 Proposed configuration



both because of the cost of the concentrator and because it would not have made use of the currently installed 50 000-bit-per-second line.

Another solution would have been to implement a distributed processing system, in which data would be distributed in such a manner that it would be closer to the end user. This alternative was not explored because it would have meant redesigning the system more than desired. Distributed systems can be simulated because distributed nodes can be modeled much as a host is, or simply as nondetailed nodes, much like a cluster.

Conclusions

Whether a system is more complex than the one illustrated, or simpler, the Systems Network Analysis Program/Simulated Host Overview Technique makes it possible to evaluate many design alternatives with minimal effort. After the initial gathering of data, only three days were required to execute the design analysis discussed in this paper, including many variations not described. The time required for gathering data depends on whether automated data collection techniques are used. Many IBM software subsystems have associated monitoring facilities. If monitors are not available, data must be collected manually, or else a monitor must be written. Many installations already have methods of tracking system traffic and utilization and can use already available data.

In making the design decisions, all the results were easily analyzed and correct conclusions readily drawn. With the simulation model it is possible to analyze design possibilities and learn how the system responds to changes without affecting the normal operation of an installation during the learning period. In addition, upon completion of several design alternatives, the user can develop a good understanding of why the system performs the way it does, and he can have confidence that it will perform up to expectations.

CITED REFERENCES AND NOTES

1. The model was developed by the Communications Systems Analysis Department of the IBM Data Processing Division's Raleigh Marketing Support Center—Communications.
2. *An Introduction to the IBM 8100 Information System*, IBM Systems Library, order number GA27-2875, available through IBM branch offices.
3. *An Introduction to the IBM 3790 Communication System*, IBM Systems Library, order number GA27-2807, available through IBM branch offices.
4. G. Gordon, *System Simulation*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1969).
5. J. R. Emshoff and R. L. Sisson, *Design and Use of Computer Simulation Models*, MacMillan Publishing Company, New York, New York (1970).
6. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15**, No. 1, 4–23 (1976).
7. *An Introduction to the IBM 3270 Information Display System*, IBM Systems Library, order number GA27-2739, available through IBM branch offices.
8. R. A. Donnan and J. R. Kersey, "Synchronous data link control: A perspective," *IBM Systems Journal* **13**, No. 2, 140–162 (1974).
9. *Introduction to the IBM 3704 and 3705 Communications Controllers*, IBM Systems Library, order number GA27-3051, available through IBM branch offices.
10. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture perspective," *IBM Systems Journal* **15**, No. 1, 53–80 (1976).
11. *Introduction to Data Management*, IBM Systems Library, order number SG20-8096, available through IBM branch offices.
12. *IMS/VS General Information Manual*, IBM Systems Library, order number GH20-1260, available through IBM branch offices.

13. A list of available IBM monitoring tools, as well as other IBM program products, can be found in *Keyword Index and Program Information*, IBM Systems Library, order number GB21-9949, available through IBM branch offices.
14. The following documents, available from the Raleigh Marketing Support Center through IBM branch offices, provide detailed information on the model: (A) *Introductory Flyer* is an introduction to what the model is and how it can be used. (B) *Input Reference Manual* describes the model's input macros and coding techniques. It also lists the latest enhancements. (C) *User's Guide* describes the data that must be collected for the model, discusses IBM's support for the program, provides examples of its use, and includes several data collection worksheets. (D) *CICS User's Guide*, similar to C, above, provides specific information for modeling a CICS system. (E) *IMS User's Guide* is similar to C but tailored for the IMS user. (F) *TSO User's Guide* is similar to C but tailored for the TSO user. (G) *Output Reference Manual* describes the model's output, or report data. (H) *Samples Manual* contains input and output reports for eight sample cases, with comments on each. (I) *Error Messages Manual* lists all error messages that might occur during the modeling process. Descriptive text is provided to aid the user in correcting problems.
15. *Systems Network Architecture General Information*, IBM Systems Library, order number GA27-3102, available through IBM branch offices. Pacing is discussed on page 5-2.

The author is located at the IBM Marketing Support Center—Communications, Dept 997, Bldg 622-2, P.O. Box 12195, Research Triangle Park, NC 27709.

Reprint Order No. G321-5100.