SIMULATION OF CENTRALIZED COMPUTER COMMUNICATIONS SYSTEMS

W. Chou, H. Frank and R. Van Slyke
Network Analysis Corporation
Beechwood, Old Tappan Road
Glen Cove, New York 11542

ABSTRACT

In this paper we describe the simulation approach for a general centralized computer communication system with emphasis on efficiency and versatility. The simulation program consists of three modules representing different levels in a hierarchy. The lowest level models the polled multidrop line connecting remote terminals to the concentrator. The second level models the trunk lines and the concentrator which interfaces the multidrop lines of lower speed to one or more higher speed trunk lines connected to a central computer. The highest level models the central computer which communicates with remote terminals via the trunks.

A hybrid simulation approach is used to ease program development and to shorten computer running time. When feasible, empirical distributions, analytical formulae or analytical models are used to eliminate simulation steps and simplify simulation procedures. The techniques developed are illustrated by application to the NASDAQ System.

1. INTRODUCTION

Determining the performance of a large centralized computer communication system is an extraordinarily difficult task. Events occur over a wide range of time intervals. Computer operations take place in microseconds, modem and channel operations take place in milliseconds, human interactions are on the order of seconds or minutes, and mean times between failures of equipment are usually weeks, months, or even years. Consequently, brute force simulations of such systems usually lead to large, unwieldy and unverifiable computer programs.

A centralized computer communication system is an interconnection of computers, communication devices, and terminals designed for the collection and distribution of data over a broad geographical area. Low, medium and high speed communication lines usually tie these devices to the Central Processing System. Terminals typically either contain or are connected to control units located in subscriber offices. The control units are connected on multidropped regional lines to regional data concentrators located in areas selected to minimize overall communication costs. The concentrators in turn are connected to the Central Processing System by trunk lines. A simple diagram of such an arrangement is shown in Figure 1. A complete message transaction from arrival at a terminal to receipt of the response at the terminal follows the sequence of events given in Table 1.

TABLE 1 MESSAGE FLOW IN SYSTEM

1) Message transmission from terminal to control unit.
2) Inbound transmission from control unit to concentrator on regional line.
3) Storage in input buffer of concentrator.
4) Concentrator processing.
5) Inbound transmission on trunk line from concentrator to central processor.
6) Storage in the central processor buffer staging area waiting to be processed.
7) Activation of CP processing to completion of processing.
8) Storage in buffer area waiting for outbound transmission to originating concentrator.
9) Outbound transmission on trunk line from central processor to concentrator.
10) Storage in reply buffer at concentrator.
11) Concentrator processing.
12) Outbound transmission on regional line from concentrator to control unit.
13) Message transmission from control unit to terminal.

In this paper we describe the simulation of a general centralized computer communication system with emphasis on efficiency and versatility. The simulation program consists of three modules. Each module represents a different level in a hierarchy. The lowest level models the polled multidrop line connecting remote terminals to the concentrator. The second level models the trunk lines and the concentrator which interfaces the multidrop lines of lower speed to one or more higher speed trunk lines connected to a central computer and the highest level models the central computer which communicates with remote terminals via the trunks.

A hybrid simulation approach is used to ease program development and to shorten computer running time. When feasible, empirical distributions, analytical formulae or analytical models are used to eliminate simulation steps and simplify simulation procedures. The approach is illustrated by application to the NASDAQ System[1,2].

The objective of the simulation system is to determine the capacity of the system under

differing input conditions. Within the maximum absolute limitations on the number of transactions the system can handle, usable system capacity is a function of response time. Response time is in turn a function of various system parameters such as regional line speed, trunk line speed, trunk line utilization, message processing time at the central processor, central processor occupancy, the number of control units on a regional line, the number of terminals on the regional line, and so on.

2. REGIONAL LINE POLLING, SELECTION AND FLOW CONTROL

An integral part of the performance analysis model is a regional line simulation system to test a multidropped line connecting terminals to a concentrator. The program is flexible and versatile within the limit of available information. The main body of the program includes four major tasks on a multidropped line: terminal polling, terminal selection, inbound message request and update transmission, and outbound message reply transmission. Terminal polling, terminal selection and message transmission are all simulated on an event-by-event basis according to procedures described below.

Polling Procedures

When no outbound message (reply message from the concentrator to a terminal) is ready for delivery, the concentrator sends polling sequences to terminals to ask if they have any messages to send. Two polling procedures are widely used from concentrator to terminals on regional lines. In both cases, only one character is used for poll identification.

String Polling

The string polling control sequence has the following format:

E	P	P		P		P	P
N	I	I	...	I	...	I	O
Q	D_1	D_2		D_1		D_n	T

ENQ is an ASCII character indicating the beginning of the polling sequence; EOT is an ASCII character indicating the end of the polling sequence; PID_i is the poll identification for the i^{th} terminal of the multidrop line. String polling requires response only if the terminal has data to transmit. Upon recognition of its polling code, the terminal will respond by raising carrier and generating a continuous binary one on the inbound transmission line (from the terminal to the concentrator) followed by the data. The concentrator will respond to the continuous binary one condition by clamping the outbound transmission line to a binary zero condition before the eighth bit of poll identification character immediately following the poll identification character of the terminal that responded. Consequently, the terminal polled will not recognize the character. Since all this must be done before the eighth bit, the total time required for: (1) propagating the last bit of the poll identification character of the responding terminal, (2) the terminal to recognize the character and raise the carrier, (3) the binary one signal to be propagated to the concentrator, and (4) for the concentrator to detect the carrier and clamp the outbound transmission line to zero condition, should be less than the transmission time of 7 bits.

Discrete (burst) Polling

A discrete polling control sequence consists of two characters and has the following format:

P	E
I	N
D_i	Q

If the terminal has no data to send, it raises the carrier and transmits an EOT character on the inbound transmission line. If it has data to send, it raises the carrier and transmits the data.

The advantage of the string polling is that the terminal waiting time can be significantly reduced. However, it has the disadvantage that the concentrator or CPU cannot know whether the terminal is still in operation if it does not respond. For this reason both string polling and discrete polling are often alternated within the same system. Discrete polling is periodically used to see if any terminal is not functioning. String polling is the predominant polling scheme because of its low overhead.

Selection Procedures

When an outbound message is ready for delivery, the concentrator sends a selection sequence to the proper terminal before it delivers the message. Both "fast" select and "acknowledged" select are considered in the program. They both meet ANSI standards.

Acknowledged Select

The selection sequence consists of three characters: SOH, PID and ENQ. When the concentrator receives from the terminal an affirmative reply (ACK) to its selection call, a message is sent from the concentrator to the terminal. When the concentrator receives a negative reply (NAK) or no reply, the concentrator will either retransmit the selection sequence or stop the call.

Fast Select

A fast select sequence consists of two characters: SOH and PID. It is attached to the outbound message as part of the prefix. Immediately after having transmitted the select sequence, the concentrator sends out the outbound message. In the acknowledged

selection procedure, a selection sequence requests consent from the terminal. In the fast selection procedure, a selection sequence notifies the terminal that a message is to be delivered immediately following the selection sequence and no reply to the selection call is expected.

The advantage of the fast select is its lower overhead in sending a message. By using it, overall response time can be improved. However, if a positive acknowledgment to the outbound message is not received by the concentrator, the concentrator does not know whether it is caused by the line error or terminal malfunction. To avoid this disadvantage, acknowledged select is used when a positive acknowledgment is not received by the concentrator by using fast select.

Concentrator Flow Control Procedures

The concentrator is a message oriented device, typically minicomputer based and core limited. The scheme with which input and reply buffers are accessed imposes limitations on the rate at which messages may flow. In typical systems, buffers are either dedicated to each regional line or are dynamically allocated from pools to regional lines according to terminal activity. If fixed buffers are assigned to regional lines, a regional line flow control scheme is necessary. With dynamically allocated buffers, a flow control scheme based on the total number of outstanding messages is more desirable.

An extreme case of regional flow control is the following: when a message is transmitted from a control unit to its concentrator, polling on the control unit's regional line is suspended until the message is processed by the concentrator and transmitted to the central processor over a trunk line. Polling on that line is then resumed until another message is found. The waiting message is then transmitted to the concentrator input buffer where it must wait until the response to the first message is received back at the concentrator or a time-out interval at the concentrator is completed. Thus, control units may have at most one outstanding message and a regional line may have at most one outstanding message. This latter restriction is necessitated by the static query buffer allocation at the concentrator. The restriction is imposed to prevent a message from returning from the central processor to the concentrator before a buffer is available. Thus, the central processor may assume that a buffer is available at the concentrator for every message and a positive acknowledgment scheme from concentrator to central processor is not essential.

If a dynamic buffer pool is available, then polling on all regional lines is suspended when all buffers are in use. Another extreme occurs when no positive flow control is used. In this case, messages which arrive when the buffer pool is full are not transmitted to the central processor and eventually a "time out" at the terminal must alert the user. Within the current model, we use a flow control procedure which does not allow more messages into the system than can be handled by a dynamically allocated return (i.e., outbound from central processor) buffer pool. Hence, messages are prevented from entering the system when potential overload conditions occur.

3. ANALYTIC MODELS FOR CONCENTRATOR /TRUNK LINE CONFIGURATION

Since it takes time to process or transmit a message, other messages may be waiting for their turn to be processed in the CPU or to be transmitted on a communication line. Queues are thereby formed and buffers are occupied. Many factors dictate the buffer occupancy distribution and the waiting time distribution of message traffic. Both simulation and analysis are possible to predict performance of the concentrator/trunk line segment. However, the simulation approach is costly and analytic approaches can give acceptable results if the problem is properly modeled.

Single Server Analysis[3]

A "single server" can only process or transmit one message at one time and the next message cannot be processed or transmitted until this one has finished processing or transmission. The formulae for waiting time and for the number of messages in waiting depend on message traffic distribution, service time distribution and the dispatching discipline. Service time is the time to process or transmit a message. The dispatching discipline is the rule to determine which message should be served first. For example, in the NASDAQ System, all quote request messages have equal priority and since the volume of updates and other messages are a very small percentage of the quote requests, the dispatching discipline is essentially a first-come-first-serve policy.

It is almost impossible to predict future traffic distributions in communication systems. However, in most systems messages arrive randomly and independently, and traffic patterns are chosen to be "Poisson distributed" to obtain tractable distributions. Furthermore, it has been shown in many queueing analyses that if there are several inputs to the server, then the arrival distribution to the server (which is the sum of the individual inputs) can be approximated as a Poisson distribution regardless of the distribution types of the individual inputs.

With traffic arrivals in a Poisson pattern and arbitrary service time distribution, the average waiting time for the server (processor or trunk) and the average number of messages in the queue are given below.

Average waiting time of a single server

$$AVWTS = \frac{P}{2(1-P)} \cdot AVS \cdot (1+A^2) \qquad (3.1)$$

Average number of messages in waiting for a single server

$$AVNS = \frac{P^2}{2(1-P)} \cdot (1+A^2) \qquad (3.2)$$

where

P = facility utilization factor (for the processor, it is the percentage of the time that the processor is processing; for the trunk, it is the percentage of the time the trunk is transmitting).

AVS = average service time for one message (for the processor, it is the average time to process one message; for the trunk, it is the average time to transmit one message).

A = coefficient of variation for the service time distribution
= standard deviation/mean.

In the above equations "A" equals unity if the service time is exponentially distributed and equals zero if the service time is constant. For other practical situations the coefficient of variation is between one and zero. (Equation (3.1) is known as the Pollaczek-Khintchine equation.)

There are no manageable equations for the probability distribution functions for the waiting time and the number of messages in waiting, except if the service time is exponentially distributed. However, the waiting time from an exponentially distributed service time is usually greater than the one from a non-exponentially distributed service time, if the average service time is the same. Therefore, using equations based on exponentially distributed service time gives more conservative results. Experience and simulations indicate that these results, though conservative, are close to true answers in the communication environment.

With message arrival in a Poisson pattern and service time exponentially distributed, the probability functions of waiting time and number of messages in waiting are as follows:

$$\text{Prob(waiting time} > t) = Pe^{-(1-P)t/AVS} \qquad (3.3)$$

$$\text{Prob(exactly N messages in waiting)} = (1-P)P^N \qquad (3.4)$$

$$\text{Prob(more than N messages in waiting)} = P^{N+1} \qquad (3.5)$$

Multiserver Analysis

In a multiserver environment two or more servers may serve a same function simultaneously if there is more than one message demanding service. For example, if there are redundant lines between concentrator and CPU, there are two servers for transmitting messages from the concentrator to the central processor. Thus, when a message is ready to be transmitted, it can go to either of the two trunks not transmitting another message. The following are the formulae for the average waiting time and the average number of messages in waiting.

Average waiting time for multiserver

$$AVWTM = (B/M)AVWTS \qquad (3.6)$$

Average number of messages in waiting for multiserver

$$AVNM = B \cdot AVNS \qquad (3.7)$$

where AVWTS is average waiting time for a single server; AVNS is the average number of messages in waiting for single server; M is the number of servers; and

$$B = (MP)^M / (P(M!)(\sum_{N=0}^{M}((MP)^N/N!) - P\sum_{N=0}^{M-1}((MP)^N/N!)))$$

Furthermore, it is assumed in equations (3.6) and (3.7) that all M servers have equal capacity and are equally loaded.

One important similarity between the behavior of single server and multiserver queues is not demonstrated by the above equations. The averages given in equations (3.6) and(3.7) are obtained by averaging both those messages which have actually waited and those messages which have been instantly served without waiting at all. If the average waiting time is obtained only from those messages which have actually waited, the average waiting times will be AVS/(1-P) and AVS/M(1-P) for single server and multiserver, respectively. Similarly, the average number of messages in waiting will be $P^2/(1-P)$ for both cases. The above discussion indicates that, so far as the waiting time and number of messages in waiting for those messages actually waiting are concerned, the multiserver may be viewed as a single server with a capacity M times as much as each of the original servers. The average waiting time and the number of messages in waiting will then be (AVS/M)/(1-P) and $P^2/(1-P)$, the same as in the multiserver case. In this way, the average waiting time is equivalent to AVS/M.

To be conservative, we consider only those messages which must wait for service. Therefore, the multiserver can be treated as a single server under the conditions stated above. It should be emphasized that all the discussions have been focused on messages in waiting, not on messages in service. Thus, the actual average service time for messages in service is AVS, rather than AVS/M, and the maximum number of buffers required for messages in service is M, rather than one.

Trunk Line Queueing Analysis

For generality, we consider the case where there are two trunk lines connected between the central processors and each concentrator facility. For messages in waiting, the trunks are equivalent to a single line with twice the single line speed. For messages in service, the line speed is the same as the single line's

speed. The precise performance of the system is then dependent on line speeds, message length distributions, etc. To provide an illustrative case, we consider the NASDAQ system. Here, trunk lines are either 50,000 bps or 7,200 bps. The distribution of message lengths for the NASDAQ system is known fairly precisely.

Approximately 95% of the reply messages have a length of 120 characters and 5% have a length of 36 characters. Thus,

The average reply length (mean) = 132 characters (1056 bits)

The coefficient of variation, A = 0.266

The average service time = .0645 sec. on a 7.2 Kbps line (used in determining waiting time) or = .01056 sec. on a 50 Kbps line

The average transmission time = .129 sec. on a 7.2 Kbps line or = .02112 sec. on a 50 Kbps line

Equation (3.1) is used for determining the average waiting time of a reply. Similar approaches can determine the input message waiting and transmission time. However, input messages are generally short and queueing delays arise primarily from return message queueing. For the outbound trunk, P is 0.75; for the inbound trunk, P is 0.13.

To calculate the number of messages in waiting, Equation (3.5) is used. In the program a random number between zero and one, representing a probability, is generated. If its value is greater than or equal to P^N but less than P^{N+1}, N messages are in waiting. The time a message must wait for its turn to be transmitted is then equal to the average service time multiplied by the number of messages in waiting.

4. CENTRAL PROCESSOR UNIT

Analysis of central processor performance is among the most difficult of all analysis problems. Analysis procedures are critically dependent of the function of the system and the processor configuration. However, a major factor in the analysis approach utilized is that while CPU operations take place in microseconds and processor drum and disk operations in milliseconds, message transmissions typically require hundreds of milliseconds and required response times are measured in seconds. Consequently, detailed microsecond simulations of the processing system are pointless. What is needed is semiquantitative predictions of throughput and saturation regions. Useful predictions techniques are also system dependent. Hence, we again use the NASDAQ system as a representative example.

Central Processor Facility Description

The overall Central Processor System is shown in Figure 2. Communications between central processors and concentrators is controlled by the Communications Terminal Module Controller (CTMC). There are two CTMCs. Either CTMC can handle all necessary traffic but ordinarily both are in operation.

One CTMC can be connected to up to 16 Communication Terminal Modules (CTM). 12 CTMs are connected to the modems which are connected to the communication lines. The CTMs, which must be matched to the communications lines, recognize end of text characters. The CTMs cause interrupts one character after such characters and generate monitor interrupts if the buffer being read from is empty or the buffer being read into is full. External interrupts occur on receive, one character after end of text is detected. Both vertical and horizontal parities are generated and checked by the software.

There is no storage in CTM except for a one or two character buffer. The CTM processes each message on a character by character basis and operates in a full duplex mode. SYNC characters are originated by the software. Horizontal parity and end of text characters are generated by the software and placed in core along with the outgoing message.

All messages are queued for processing or for transmission in core. Each 7,200 bit/sec. line has two dedicated buffers for input and 50,000 bit/sec. line has three dedicated buffers. When the end of text character of a message is recognized, the message is moved from its input buffer to a buffer in a shared pool. This pool, called a staging area, contains approximately 70 buffers, each long enough to store the entire input message and its response. After a message arrives in the staging area, all operations are performed without further transfer.

Output queues for each communication line are formed within the staging area. Output characters are transmitted to the CTMs by the CTMC such that the lowest number CTM (i.e., the ones connected to the 50 kilobit/second line) have the highest priority. If long queues for one or more concentrators grow, the concentrators may either individually or jointly be slowed down by stopping further transmission to the central processor for a short period of time.

Each CTMC is connected to each central processor through a switch to its own I/O channel. Each I/O channel has a transmission rate of 2.65 microseconds per character for communications.

The on-line system keeps track of inputs, accepts subscriber requests, updates and references the data pool, generates Level 1 output and keeps an activity record. In handling a quotation request or quotation update, the following primary units are involved: CPU, core, and drums. The number of references to core and drum vary depending on whether the

message is a quotation request or update and whether the security involved has been defined by the system to be "active" or "inactive." The designation of securities as "active" or "inactive" is made daily. After closing, securities are ranked according to the number of requests for quotations made during that day. The most active securities of the day are then designated as the "active" files for the following day's processing. Computation times vary between active and inactive files. For complete processing, one drum access is needed for an active security quotation request and two for an inactive one. For an active quotation request, the single access is to a high speed drum while for an inactive request one access is to a high speed drum while the other access is either to a high speed or a low speed drum. For an active quotation update, three accesses, all to high speed drums, are required while for an inactive security, an update requires one access to a high speed drum and three to either a high speed or low speed drums. Up to 7 messages can be processed together. This number is chosen to maintain effective drum utilization.

The drum rotation times are 8.5 milliseconds and 34.1 milliseconds. Only two drums on the same subsystem can be accessed at the same time. Files are distributed in duplicate over all drums. All drums are normally in operation. On an update call the primary file is updated and then copied to the backup file. The directory for active stocks are core stored and for inactive stocks stored on the high speed drums.

Computation times also vary depending on whether the message is an update or a quotation request, and if it is a quotation request, whether the request is for the first frame or for a substantial frame. The average computation times for a quotation request is approximately 4 milliseconds for the lst frame and 8 milliseconds for each subsequent frame. An average update request requires about 8 milliseconds of computation time. Table 2 summarizes average service times for various message classifications.

TABLE 2
MESSAGE SERVICE TIMES IN CENTRAL PROCESSOR

Type	Average Processing Times	Average Drum Time	Total Average Service Time
Active Update Request	8 ms	8.4 ms	16.4 ms
Active Quotation Request (lst frame)	4	4.2	8.2
Active Quotation Request(2nd frame, etc.)	8	4.2	12.2
Inactive Update Request	8	34.1	42.1
Inactive Quotation Request (lst frame)	4	17.1	21.3
Inactive Quotation Request(2nd frame, etc.)	8	17.1	25.3

Buffer Utilization

The buffer pool in the central processor is shared by several queues: the (four) concentrator facilities and the CPU. If each of the queues is in a single server environment with Poisson arrival and exponential service, and if the facility utilization factor is the same for each of the servers, then

$$\text{Prob(more than m messages in waiting for service)} = \sum_{i=m+1}^{m+5} \binom{m+5}{i} (1-p)^{m+5-i} p^i \qquad (4.1)$$

Equation (4.1) may be interpreted as the probability of buffer overflow if there are m buffers in the buffer pool. This formula can be used to determine the number of buffers required so that the probability of overflow is small.

The number "m" in Equation (4.1) includes both messages waiting for service and messages in the process of being served. This formula gives a conservative result for the messages in waiting, but not for the ones in service. As stated, Equation (4.1) is for single servers. Yet, each of the five queues has more than one server queue for the messages in service. For a multiserver queue, the buffers required for messages in service. For a multiserver queue, the buffers required for messages in service may be as many as the number of servers. Therefore, given a value of probability, the total number of messages in the system may be larger than the one obtained from Equation (4.1), but with a difference of no more than the total number of servers minus the number of different queues. The buffer pool in the central processors is shared by five different queues and fifteen servers (eight trunks and seven active modes in the central processors). To be conservative, Equation (4.1) should be interpreted as the overflow probability when there are m+10 buffers. Therefore, Equation (4.1) should be rewritten as

$$\text{Prob(not enough buffers when there are m+10 buffers in the pool)} = \sum_{i=m+1}^{m+5} \binom{m+5}{i} (1-p)^{m+5-i} p^i \qquad (4.2)$$

If the number of queues is not five, (4.2) can be modified as

$$\text{Prob(not enough buffers when there are m+10 buffers in pool, and there are M different queues.)} = \sum_{i=m+1}^{m+M} \binom{m+M}{i} (1-p)^{m+M-i} p^i \qquad (4.3)$$

Processor Utilization and Queueing

System capacity is severely limited by the average processing time required for each transaction. Processing time includes both application time and executive overhead. This processing time varies with time of day, percentage of processor occupancy and type of messages.

It must be emphasized that unless an extensive and expensive analysis is performed, either by simulation or by measuring the processor occupancy with simulated messages of various rates, it is impossible to predict accurately the remaining usable capacity of the processor. However, measurements of processor occupancy under the sustem's normal operating condition can substantially aid in analysis. With these data simplified mathematical models can be derived to approximately predict the processor's behavior.

Figure 3 shows the processor occupancy behavior during the busiest time period. To be conservative these data are used. The straight line shown is the least square regression line fit for these data. From this line, an equation can be obtained as

Transaction rate = 79 times processor occupancy - 20

When both processors are 100% occupied, the above equation gives the maximum transaction rate of about 138 transactions per second. The actual maximum may be somewhat higher or lower than this number. In addition, inefficiencies in the two processor configurations lead to no more than about 150% occupancy for message processing. In other words, the average transaction rate should be no more than approximately 100 transactions per second.

5. MODULE INTEGRATION AND AN ILLUSTRATIVE EXAMPLE

The use of the model involves the integration of the three modules in the following way:

Polling, line selection and flow control are simulated on an event-by-event, message-by-message basis for a "test" regional line. The behavior of the other regional lines connected to the concentrator is approximated and an analytic queueing model to represent the trunk line behavior is applied. When a message in an input buffer of the test regional line is ready for transmission to the central processor, the waiting time for the access to the inbound trunk line depends on the traffic load of other regional lines connected to the same concentrator. To determine message waiting time, the number of regional lines having an unempty input buffer must be determined. This number is calculated as follows: Given that the average transaction rate on the trunk(s) connecting the concentrator to the CPU is known, the total number of waiting inputs can be determined by generating a random number. (The formula used to obtain this number is detailed in Section 3.) The time waiting for the access to the trunk is then equal to the number so obtained multiplied by the time required to transmit one average inbound message from the concentrator to the CPU. Knowing the utilization on the outbound trunk, the waiting time for access to it can also be determined by generating a random number (the formula used is in Section 3) the average delays for inbound and outbound messages on the trunks are therefore computed. To complete the calculation, the time a message spends in the CPU must be estimated. This is done using the queueing approach described in Section 3 in conjunction with the empirical processor occupancy curves for the CPUs. With this simple analysis, for a given number of messages requiring service at the CPU, processor utilization is easily determined. Equation 3.1 is then used to estimate the average waiting time. This estimate is incorporated in the trunk/concantrator analysis to predict a return time for an outbound message to reach the concentrator. These analytic predictions are then used in the regional line simulation to estimate response times and system throughput.

To illustrate the possible uses of the simulation system, we again use the NASDAQ system. Figures 4 and 5 show respectively, the effect on terminal response time when traffic on a test regional line is varied with all other traffic held constant and the terminal response time when traffic on the test line is held constant but traffic outside the line is varied. Figures 6 and 7 show two other parametric studies--regional line configuration variation and increase in trunk line speed.

To produce each such curve, approximately 30 seconds of CDC 6600 time was required(about $10). Note that if a complete simulation of the entire system had been attempted, the generation of even a few of these curves would have been prohibitively expensive. On the other hand, a completely analytic approach to the problem would have led to substantial inaccuracies and inflexibilities. Consequently, the hybrid approach is extremely well suited for practical and economical studies of existing and proposed centralized computer communication systems.

References

1. H.Frank, I.T.Frisch, R.Van Slyke, "Testing the NASDAQ System-Traffic and Response Time," Proceedings of the Symposium on Computer Communication Networks and Teletraffic, Polytechnic Press, N.Y. pp.577-586, 1972.
2. H.Frank, I.T.Frisch, R.Van Slyke, "Testing the NASDAQ Systems-Reliability and Availability," (in preparation).
3. L. Kleinrock, Queueing Systems: Theory and Applications, Wiley, N.Y., 1973.

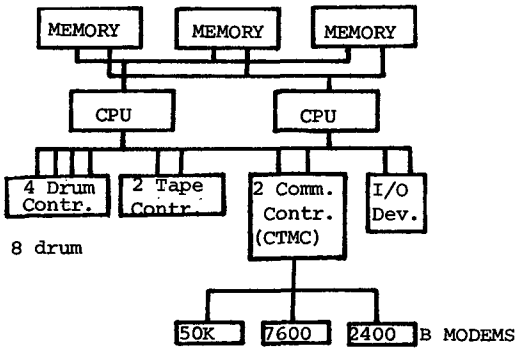


Fig. 2 A simplified block diagram of the NASDAQ system Central Processing System.

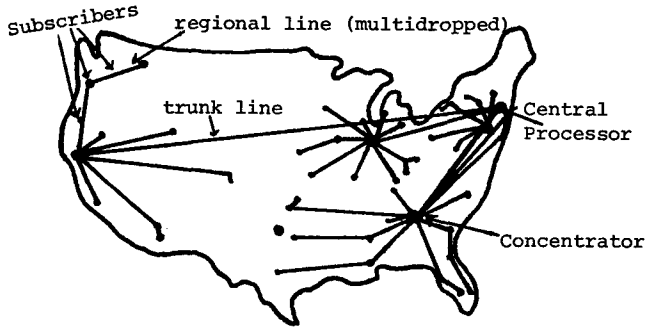
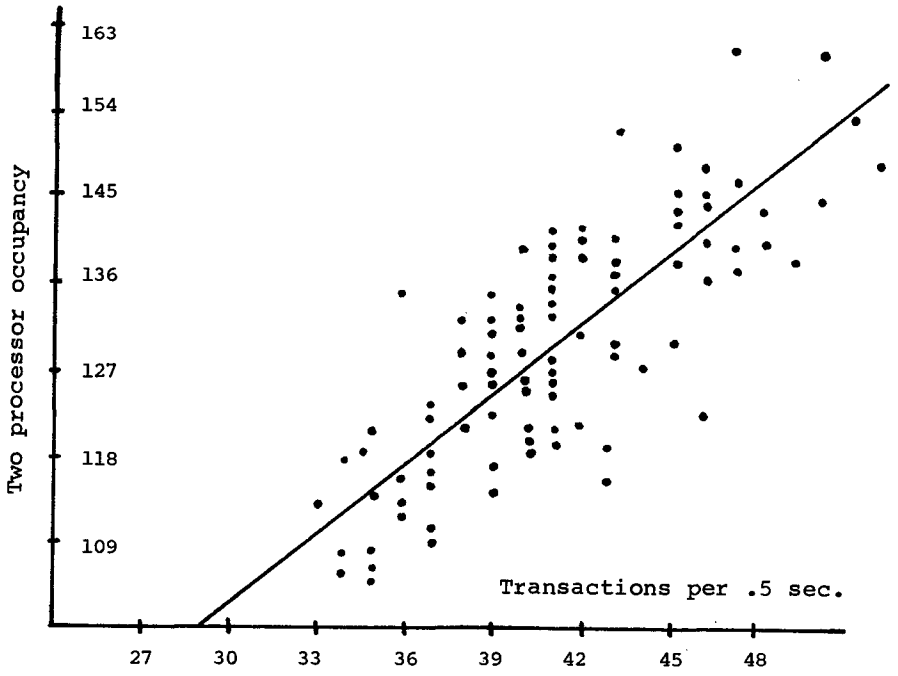


Fig. 1 A Simplified layout of a centralized Computer Communication System (e.g. The NASDAQ System)



Two processor occupancy rate during busy period

Figure 3

Effect on terminal response time of changes in regional line traffic rate

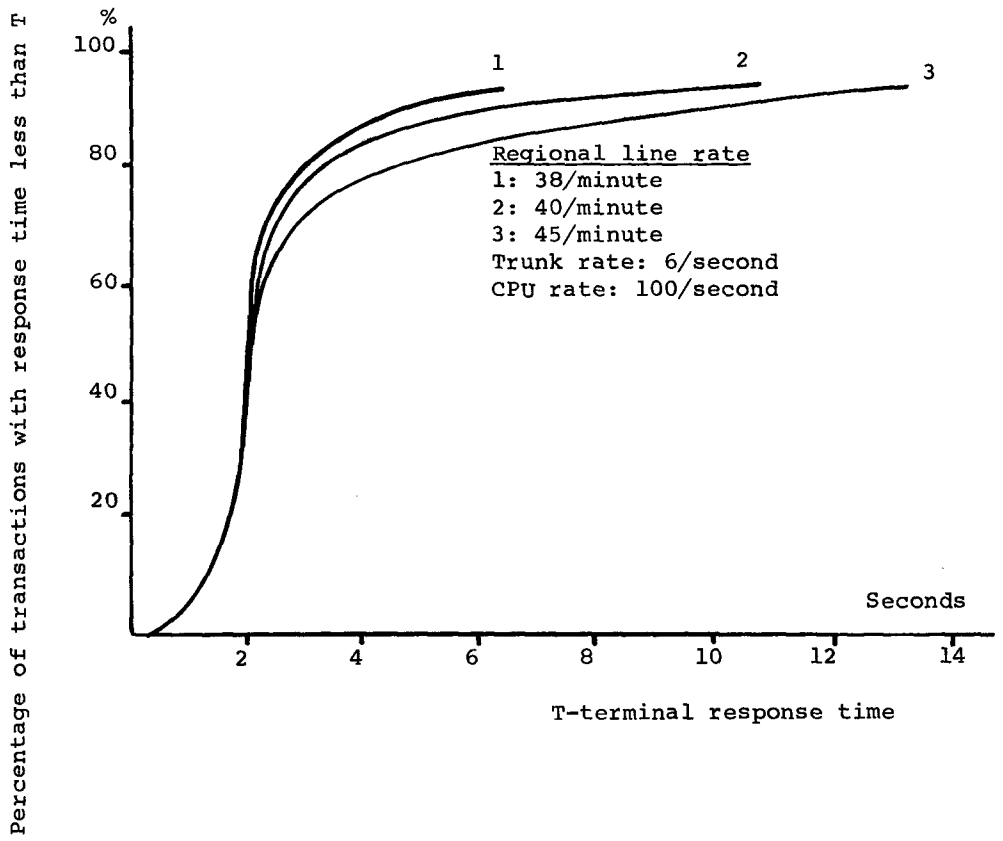


Figure 4

Effect on terminal response time of changes in traffic outside of test regional line

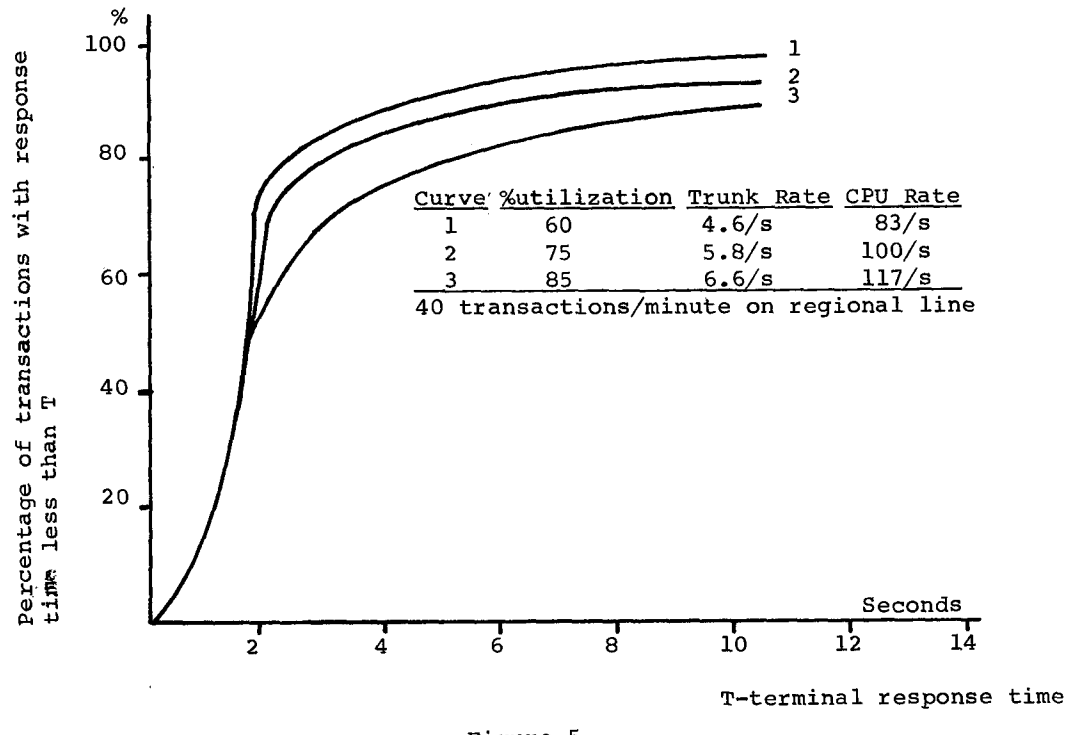


Figure 5

Effect of two different test regional line configurations on terminal response time

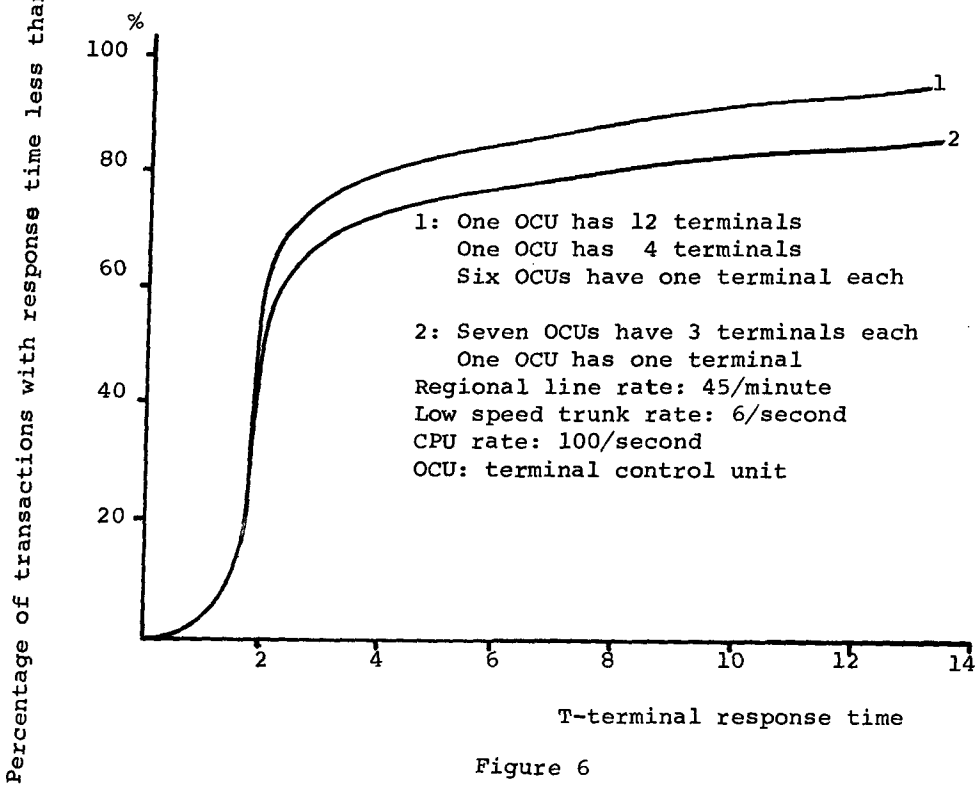


Figure 6

Effect on terminal response time of 50 KBPS trunk line speed

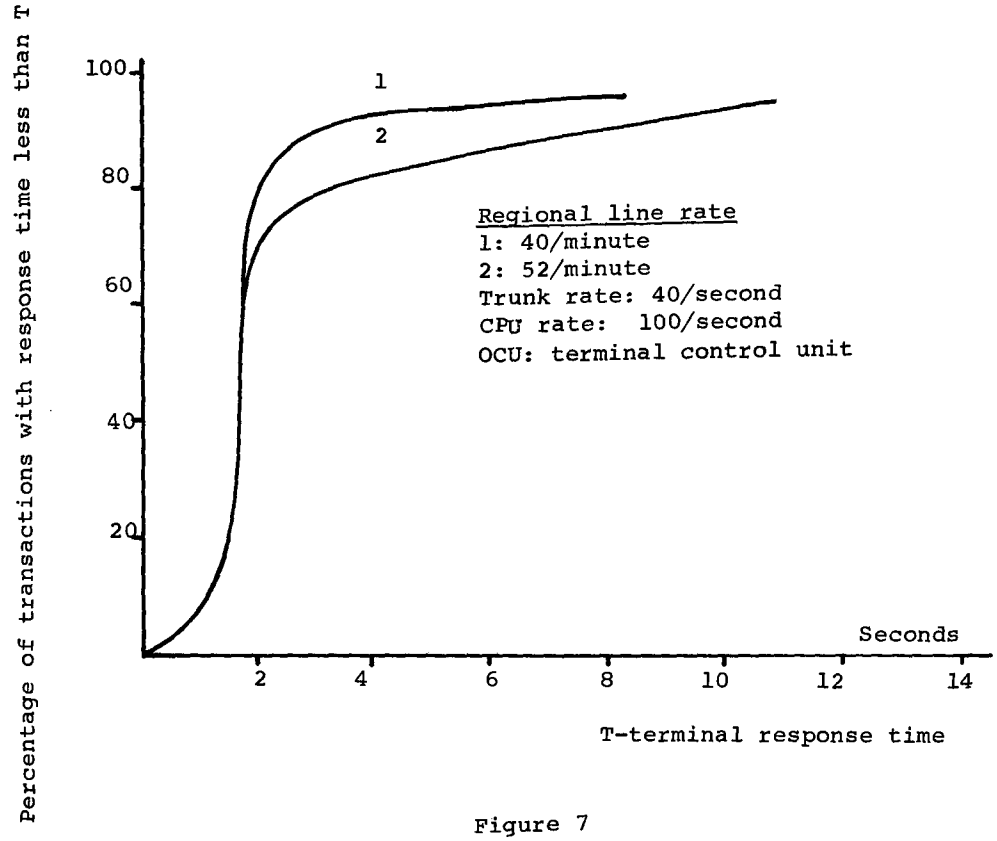


Figure 7