SIMULATION OF A MINICOMPUTER IN A COMMUNICATIONS SYSTEM

Frederick V. Crowley

Bell Canada

ABSTRACT

An analysis of the performance of a minicomputer, which is used in a computer communications environment, is presented. The purpose of the paper is to show how some of the minicomputer's capabilities may be defined, and to establish a basis for a more detailed analysis. The principal aspect being considered is message throughput. Two models, oriented about the software, were developed: one using conventional mathematical queuing techniques and the other using a discrete simulation language (GPSS). The GPSS model was used to provide an approximate verification of the assumptions and of the results obtained mathematically.

1.0 INTRODUCTION

One of the problems of using a minicomputer in a communications environment is trying to determine the performance capabilities of the machine. The manufacturer has general performance figures for his equipment, but from the point of view of the customer, these figures may not be useful. The user is interested in the minicomputer's performance with respect to his application and not in generalities. For example, some typical questions might be:

What message throughput and delay will be encountered?

What percentage utilization of the various facilities will occur in the application?

Will there be sufficient time available for other secondary tasks such as gathering statistics?

Should priorities be established for various operations within the system, and what should these priorities be?

How much buffer space will be required?

These are some questions that a system designer may have when trying to use a minicomputer in a proposed communication system.

An obvious approach is to model the system, and then run a variety of simulations. The problem that arises however, is: how should the system be modelled.

This paper shows how a design problem was solved using relatively simple modelling techniques, with a view to minimizing the cost of obtaining reasonably accurate performance predictions.

2.0 THE PROBLEM

The problem was to predict the performance of a minicomputer, from a throughput versus delay point of view, when it was used in a communications system. Briefly, the computer had to process messages from an asynchronous mode and convert them into synchronous information and vice versa. A number of asynchronous and synchronous lines were to be employed.

The minicomputer itself makes use of a Communications Executive Software package supplied by the manufacturer. This basic package provides the customer with the capability of interleaving the execution of a number of user written programs, based on their established priorities. These system programs or tasks are used to control the flow of messages and to condition the messages from the asynchronous to synchronous environment. The system software is thus divided into two principal categories:

1 The executive software overhead.

2 Customer written programs or tasks.

These software tasks are considered to be non-suspendable operations. The

term non-suspendable is used in reference to operations that may only be preempted (on a priority-preempt-resume basis) by I/O interrupts. Operations may not be preempted by other operations. Each operation is assigned a priority. When a system event occurs, rescheduling takes place on a priority basis. A system event results when one of these three events occurs:

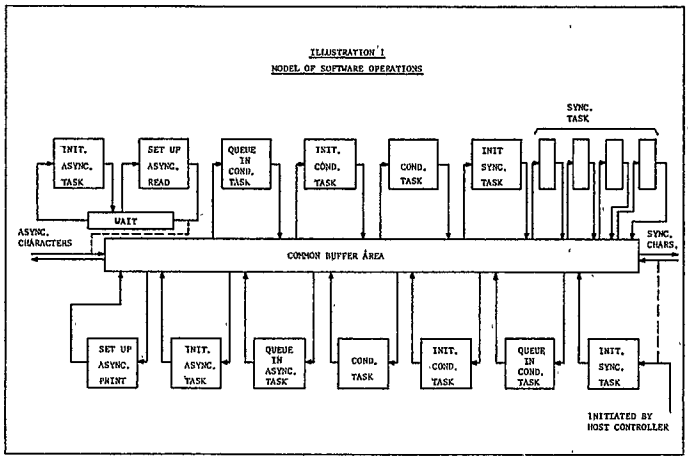
1 Completion of an I/O transfer.

2 Internal interrupt.

3 Completion or cancellation of a task.

The various software operations are shown in the block diagram of illustration 1. Referring to this illustration it can be seen that there are three principal tasks which perform two operations each. These tasks are: the asynchronous task, the conditioner task, and the synchronous task. They are used to convert asynchronous information into synchronous information, and to convert synchronous information into asynchronous information.

ILLUSTRATION 1
MODEL OF SOFTWARE OPERATIONS



An asynchronous line is initialized, characters are read into a common buffer area by the asynchronous task, and then queued for conditioning into synchronous information. A conditioner task is then initialized, the message processed, and placed back into a common buffer area. The synchronous task is then initialized, and the message transmitted on a synchronous line. The reverse procedure is followed by messages proceeding from the synchronous to asynchronous environment. Thus, these tasks assign buffers, perform the appropriate message formatting, and communicate with the synchronous and asynchronous lines. Another characteristic of the system software is that if no buffers are available, no messages are input to the system. In essence, a limiting action occurs which will restrict the message throughput if sufficient buffer space is not available.

Also, in addition to the customer written tasks, there is the software overhead. The manufacturer's supplied figures are useful in considering this aspect.

Table 1 shows how this system may be typified by counting the number of instructions and averaging the execution time and thus determining the average latency time of a single message.

TABLE 1

	Operation	No. of Instructions
1)	Initiate async task	100
2)	Set up async input (read)	250
3)	Q in conditioner task	120
4)	Initiate conditioner task	100
5)	Conditioner task operation	250
6)	Initiate sync task	100
7)	Setup time for sync	800
8)	Initiate sync task	100
9)	Q in conditioner	120
10)	Initiate conditioner	120
11)	Cond. op	250
12)	Q in async	120
13)	Initiate async	100
14)	Set up async (print)	200
	TOTAL	2730

Assume: average instruction = 2 us

therefore latency time = $2730 \times 2 \times 10^{-6}$ sec

$= 5.6 \times 10^{-3}$ sec

Therefore, given this application, the problem is to predict system performance for various design configurations.

Two approaches were considered feasible in this case:

1 A mathematical queuing analysis.

2 Computer simulation using GPSS (General Purpose Simulation System).

The principal reasons for being restricted to these two methods were cost, accuracy and computer availability.

3.0 DISCUSSION OF ASSUMPTIONS

The five basic areas where assumptions have to be made for modelling this system are:

1 Task service time.

2 Task initiation rate (implies a message arrival rate).

3 Message and character arrival rate on the synchronous and asynchronous lines.

4 Buffer space available.

5 Queuing discipline and scheduling algorithm.

The first assumption of task service time is relatively easy to consider by modelling the software. By counting the number of instructions and multiplying by an average instruction execution time, it is possible to say that the service time of each task is nearly constant (ie: the standard deviation of the service time is zero).

The second assumption to be made, concerning the task initiation rate, is the most difficult. Since only one operation can be executed at a time, and since one task must be completed before another can start, the simplest consideration is that of processing only one message. This processing of a single message causes a highly ordered procedure to take place. That is, the overall processing time for this one message is essentially constant. As the message rate increases, thus increasing the number of messages "in transit" in the system, the various operations start to be preempted for I/O interrupt servicing, and rescheduling occurs more frequently. Therefore, the initiation of some of the tasks become more disordered, with the degree of disorder being dependant upon the priority of the task. A completely disordered process for the task initiation rate may be characterized by a Poisson distribution(3). A digression here might be useful. The messages for the mini-computer originate on the asynchronous and synchronous lines, which are virtually independant of each other. Because of this independance, the message arrival rates to the system are approximately random and thus may be described by the Poisson distribution. This Poisson arrival pattern to the system coupled with the partially disordered rescheduling of tasks allows some faith to be placed in the assumption of describing the task initiation distribution of being exponential. Thus, by assuming a Poisson distribution, we will be considering a "worst case" situation. Obviously, this assumption will have to be examined more closely.

The assumption of available space is simplified by assuming that infinite buffer storage is available. If the system is designed such that the probability of there being insufficient buffers available is very small, this assumption is not unreasonable.

The queuing discipline that is used throughout this analysis is on a first-come, first-served basis (FIFO) with respect to any given task.

The scheduling algorithm is on a priority basis with non-preemptive priority for operations and preempt - resume priority for I/O interrupts.

For simplicity, no abnormal situations, such as console interrupts, machine malfunctions, etc. are considered.

4.0 THE MATHEMATICAL MODEL

Based on the previous assumptions, each of the software tasks may be characterized as an M/D/1 queuing(1) problem.

The basic problem in creating a mathematical model of this system is in handling I/O interrupts, which have preemption capabilities, and in handling the operations which have non-preemptive priority capabilities, all requiring use of the processor facility.

One approach to solve this problem is to initially consider each task in isolation. Statistically, a finite queue will develop for any given task, as long as the arrival rate is less than the service rate, and some non-constant arrival distribution is assumed. Treating each operation in isolation permits the "extra delay" due to preemption to be calculated for each operation. Adding the "extra delay" to the service time of each operation and then recalculating their second moments, removes the problem of I/O interrupts. The problem has now been reduced to analyzing the queuing problems of the various tasks on a non-preemptive priority basis. This is now a relatively easy problem to solve and permits such factors as facility utilizations, queuing delays, average queue lengths, and the total message delay to be determined.

5.0 THE GPSS MODEL

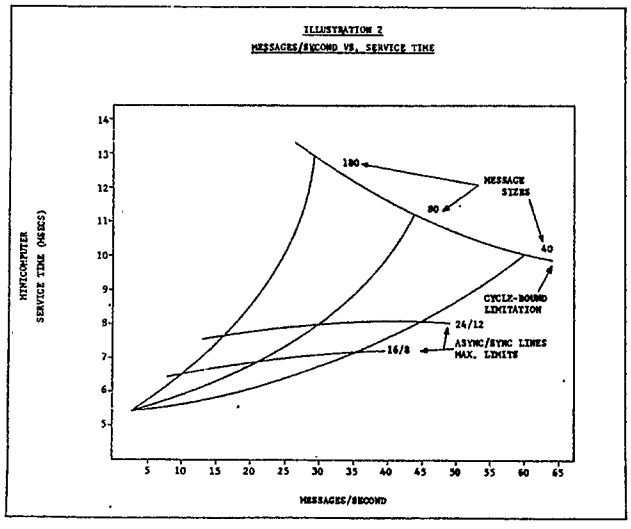
To try and eliminate some of the doubts that arise from the assumptions that were made when deriving the mathematical model, a GPSS model was created. The principal assumption that we wish to eliminate is the task initiation distribution. Referring back to illustration 1, the modularity of the software facilitates implementation in GPSS. Specifying a message arrival rate according to a known distribution

(eg: Poisson), and assigning a priority to each message depending upon which task it is entering, and by rescheduling tasks by use of a "PRIORITY BUFFER" block, a more accurate representation of the mini-computer may be obtained. The principal factor is that the nature of GPSS frees us from the constraint of having to specify a task initiation distribution.

The coding of GPSS is an encouraging factor to those who must make decisions on the system design, as it is reasonably easy to understand and relate to when compared to a mathematical analysis. The main disadvantage of using a simulation language in this application is that it is expensive. The operations that are being modelled have service times in the micro-second region, while I/O interrupts occur in the order of milliseconds. This results in a simulation where it takes approximately five seconds to simulate one second of operation. However, GPSS provides a good means of creating another model to compare to the mathematical analysis.

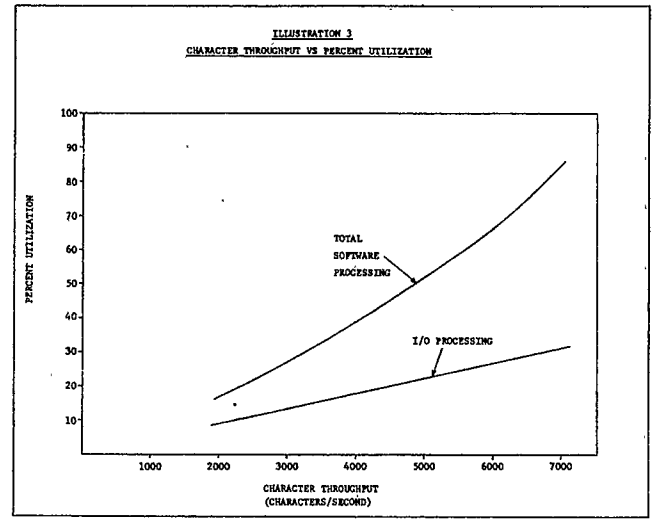
6.0 SIMULATION RESULTS

The principal characteristics of interest result from plotting throughput in messages/second, versus the overall mini-computer service time including delays and queuing. A family of curves may be drawn depending upon what message size is being considered. A typical series of curves is shown in illustration 2.



Two important restrictions are shown in this diagram. The first obvious restriction is governed by the capabilities of the mini-computer (ie: the computer becomes "cycle-bound"). This is an absolute upper maximum beyond which the mini-computer cannot operate and is determined by the point at which the processor is nearly 100% utilized. Of course, if the programming of the various tasks is made more efficient, then this curve will be shifted upwards. The second restriction is due to the physical limitations of the asynchronous and synchronous lines transmitting at fixed baud rates. In this example the line rates are quite slow (1200 to 2400 baud) and operate in a half-duplex mode. Thus, for the computer to become "cycle-bound," a relatively large number of lines are required. Therefore, the system throughput will be limited by the line capabilities before the mini-computer capabilities. An obvious conclusion would be that increasing the asynchronous and/or synchronous line speeds (if possible) will tend to increase the utilization of the machine.

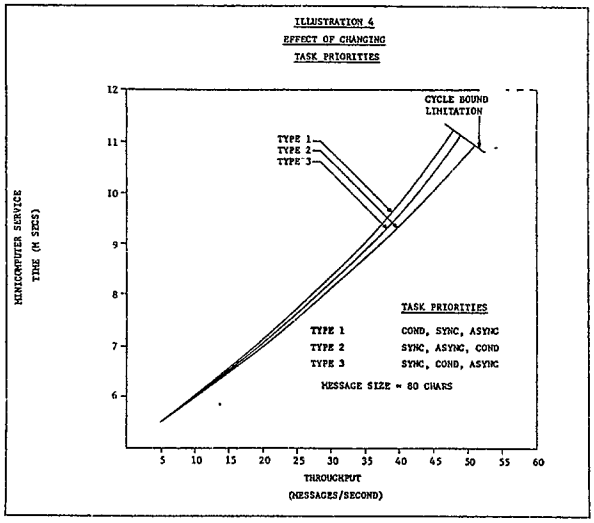
Another interesting characteristic that results from an analysis of this kind is the character throughput versus the percentage utilization of the processor (illustration 3). Two curves are of specific interest. One is the percentage



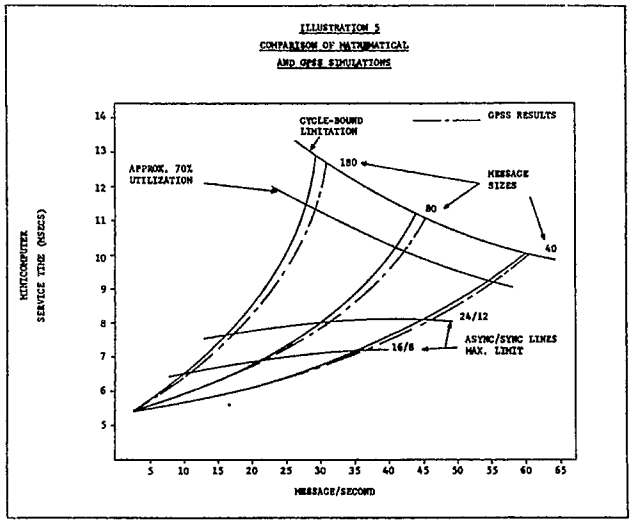
utilization of the processor for all operations and the other is the percentage utilization of the processor in dealing with I/O interrupts. These curves are of course dependant upon the system parameters (eg: message size and line rates). Note that at the "cycle-bound" point of the mini-computer, the processor utilization to handle the I/O interrupts is approximately 32%. This corresponds to a character throughput of about 7300 characters/second. If an operating region of approximately 3000 to 5000 characters/second is assumed, the percentage utilization of the processor is quite low (27% to 53%). Thus, assuming the computer does not become "core-bound," there is sufficient capabilities for other "housekeeping"chores such as statistics

gathering.

Illustration 4 shows the effect of varying the established priorities of the various operations. As can be seen there is very little difference in the overall service time for a message until the system starts to become more heavily loaded. Near the "cycle-bound" point of the curves, a small spread is noted between three typical priority schemes. The difference is almost negligible due to the fact that the service times are nearly equal. If there was a greater spread in service times, then the system delay would show a greater variation.



The GPSS results are shown with the mathematical results in illustration 5. The main item to note is that the GPSS results tend to give "better" results for system performance than the mathematical results. That is, the mathematical results tend to predict that the message throughput is slightly less than that predicted by the GPSS model. This difference is attributed to the task initiation distribution which we assumed to be Poisson.



7.0 DISCUSSION OF RESULTS

One of the more questionable assumptions that was originally made for the mathematical model was the assumption of an exponential distribution for the tasks' initiation rates. This was said to be a "worst case" assumption. One of the reasons for using a GPSS model was to eliminate this assumption. Referring again to illustration 5, note that three message sizes have been assumed. As can be expected, as the message throughput increases, there are increased delays in the system, and the mini-computer service time increases until the "cycle-bound" point is reached. Note that as the message throughput increases (by decreasing message size), the difference between the results of the two modelling techniques decreases. For example, with a processor utilization of approximately 70%, the percentage error for the 180 character message size is about 6%, for the 80 character message size, about 5%, and for the 40 character message size, about 2%. This tends to lend credence to the assumption that as the system becomes more heavily loaded, the task initiation distribution becomes more disordered and approaches an exponential distribution. For low processor utilizations, the queuing delays are obviously going to be relatively small, and the assumption of a Poisson distribution will not be significant for the conditions being considered.

Another aspect which must be taken into consideration is the actual physical limitations of the overall system in which the mini-computer is being used. One consideration is that it may not be physically possible to support sufficient synchronous or asynchronous lines to give a message throughput which approaches the "cycle-bound" restriction. Two curves showing the line restrictions are included in illustration 5. These two curves represent 24 and 12 asynchronous and synchronous lines respectively, and 16 asynchronous and 8 synchronous lines. Note that the percentage utilization at the points where they cross the performance curves is quite low. Thus the accuracy of the mathematical model is quite good for this operating region.

To summarize therefore, the greatest error in the mathematical model will result from the condition of a low message throughput with a relatively large message size (which implies a high processor utilization).

8.0 CONCLUSIONS

A mathematical analysis of the mini-computer permitted quick, easy to understand information to be gathered about the

predicted performance of the system. By making several general assumptions, to facilitate the calculations, and verifying them by using a discrete simulation language, it is possible to place a fairly high degree of confidence in the results.

Treating the various modular tasks in an isolated fashion to determine the "extra" service time associated with preemption, and then treating them as a "whole" to determine the various queuing delays, allows a complicated system service distribution to be modelled in a simple fashion.

The modularity of these techniques allows various system configurations to be modelled to determine the sensitivity of the system to the change. New tasks may be easily introduced into the model, or existing tasks removed. This is important in the system design phase, as many different ideas and configurations may be quickly tested before the actual physical system is built.

APPENDIX

Queuing Relationships used in the Analysis Of The System Performance

The basic theorem of a single-server queuing model is the Khinchine - Pollaczek equation(2).

$$N(w) = \frac{\rho^2}{2(1-\rho)} \left\{ 1 + \left[\frac{\sigma ts}{Ts} \right]^2 \right\} \quad (1)$$

where:

$N(w)$ = mean number of messages waiting for service.

ρ = facility utilization.

σts = standard deviation of the service time.

Ts = mean service time.

This relationship is useful in determining the queuing delays that result in computer systems and is applicable to any single-server system where an exponential arrival pattern and any distribution of service time is assumed. An important characteristic of this relationship is that it is valid for any dispatching discipline, provided that the selection of the next item to be serviced does not depend on the service time. In the case we are considering in this paper, the dispatching discipline is on a first-in, first-out (FIFO) basis with priority and for some operations, preemption.

The mean number of items waiting for service may be determined from:

$$N(w) = \lambda * T(w) \quad (2)$$

λ = average number of message arrivals/sec.

$T(w)$ = average time spent waiting for service.

The facility utilization (ρ) may be expressed by the following:

$$\rho = \lambda * Ts \quad (3)$$

Using these two relationships, equations 2 and 3, and substituting into the Khinchine - Pollaczek equation (1), the mean time ($T(w)$) that a message spends waiting for service is:

from (2) $$T(w) = \frac{N(w)}{\lambda} \quad (4)$$

using (3) $$= \frac{N(w)}{\rho} Ts$$

using (1) $$T(w) = \frac{\rho^2 Ts}{2(1-\rho)\rho} \left\{ 1 + \left[\frac{\sigma ts}{Ts} \right]^2 \right\}$$

$$= \frac{\rho Ts}{2(1-\rho)} \left\{ 1 + \left[\frac{\sigma ts}{Ts} \right]^2 \right\}$$

Now since we are considering constant service times in this analysis, equation 4 reduces to:

$$T(w) = \frac{\rho Ts}{2(1-\rho)} \quad (5)$$

as $\sigma ts = 0$

PRIORITY QUEUING

In the analysis there are two types of dispatching disciplines:

1 Non-preemptive, priority queuing.

2 Preemptive priority queuing.

Assume that a task has a priority i and this operation has priority over an i + 1 priority operation for N operations. If we assume that each task initiation is independant of the other (random - Poisson), with rates $\lambda_1, \lambda_2, \ldots \lambda_\eta$ then the total mean arrival rate is random and may be represented by:

$$\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_\eta \quad (6)$$

Now, if the various priority classes require different service times according to their priority, then the dispatching

discipline is no longer independant of the service time and thus the Khinchine - Pollaczek equation cannot be used. In this case the mean waiting time relationship may be represented by(3):

$$T(w_i) = \frac{\lambda b_2}{2\left[1-(\rho_1 + \rho_2 + \ldots \rho i-1)\right] * \left[1-(\rho_1 + \rho_2 + \ldots \rho i)\right]} \quad (7)$$

where: $\rho i = \lambda i \; b_1 i \qquad b_1 i = {}_0\int^{\infty} t dQ(t)$

$b_1 i = {}_0\int^{\infty} t dQ(t)$

$b_2 = \frac{\lambda_1}{\lambda} b_{21} + \frac{\lambda_2}{\lambda} b_{22} + \ldots \frac{\lambda N}{\lambda} b_2 N$

$b_2 = {}_0\int^{\infty} t^2 \; dQ(t)$

$Q(t)$ = service time distribution of each operation.

Since we are considering constant service times

$$Q(t) = \begin{cases} 0 \text{ when } 0<t<T_s \\ 1 \text{ when } t>T_s \end{cases}$$

Therefore $b_N = {}_0\int^{\infty} t^N dQ(t)$

$= {}_0\int^{\infty} t^N \; \delta \; (t - Ts) \; dt = Ts^N$

$b_2 = Ts^2$ and $b_{2i} = T^2_{si}$

PREEMPTIVE-RESUME PRIORITY QUEUING

This queuing situation must take into account the initial wait for service, as well as the subsequent waits for higher priority interrupts. Thus the mean waiting time for the ith customer is given by (3):

$$T(\bar{W}i) = \frac{1}{\left(1-\sum_{k=1}^{i-1} \rho k\right)} \left[\rho i \sum_{K=1}^{i-1} \rho k + \frac{\lambda i \sum_{K=1}^{i} \lambda_K \; b_{2K}}{2\left[1-\sum_{K=1}^{i} \rho k\right]} \right] \quad (8)$$

where $P_i = \lambda_i \; Ts_i$

b_{2k} = second moment of service time for the kth operation.

λ_i = mean arrival rate for the ith operation.

Now considering situations in which there are only two priority classes (as there would be when considering operations and I/O interrupts in isolation), then equation 8 reduces to:

$$T(w_i) = \frac{1}{1-\rho_1} \left[\rho_2 \rho_1 + \frac{\lambda_2 (\lambda_1 b_{21} + \lambda_2 b_{22})}{2(1-(\rho_1+\rho_2))} \right] \quad (9)$$

BIBLIOGRAPHY

1. Chu, W.W. and Konheim, A.G., On the Analysis and Modelling of a Class of Computer Communication Systems, IEEE Trans. on comm., vol. com-20, No. 3, June 1972.

2. Martin, J. Systems Analysis for Data Transmission, Prentice-Hall Inc., Englewood Cliffs, N.J., 1972.

3. IBM Publication GF20-0007-1, Analysis of Some Queuing Models in Real-Time Systems.