A Queueing Model of Multiprogrammed Computer Systems Under Full Load Conditions

ALEXANDRE BRANDWAJN

École Nationale Supérieure des Télécommunications, Paris, France

ABSTRACT. This paper presents a queueing model of a multiprogrammed computer system with virtual memory Two system organizations are considered (i) all the processes present in the system share primary storage, (ii) processes which have generated a file request (slow I/O) lose their memory space until the I/O is completed Our model assumes balanced memory allocation among processes, and accounts for the memory sharing effect through the use of lifetime functions The model explicitly takes into account the fact that, if a written-onto page is to be replaced at the moment of a page fault, it first has to be saved in the secondary memory. An approximate closed form solution is obtained by using an equivalence and decomposition approach A procedure for evaluating the accuracy of the approximation is presented The numerical examples illustrate the influence of the system and program behavior parameters taken into account in our model

KEY WORDS AND PHRASES multiprogramming, queueing theory, equivalence of queues, decomposition methods, computer modeling, virtual memory, throughput

CR CATEGORIES 4 32, 4 35, 5 5

Introduction

Relatively general results on queueing networks [13, 12, 3] exist, and they have been successfully applied to the study of the performance of computer systems [16, 15, 9]. The queueing networks for which an exact analytical solution is known have in common the fact that the service rate for a service station may depend only on local congestion, i.e. on the number of customers at the service station. Their analytical solutions share the product form; i.e. they have the form of a product in which each factor corresponds to the state of one service station.

These results, however, do not allow taking into account some important dependencies encountered in computer systems, and this motivates the search for new, possibly approximate, solutions of complex queueing networks [2, 10, 11, 6]. Our approach in this paper will be of the latter type. We consider a model of a batch virtual memory system in which the random variable representing uninterrupted computation time at the CPU and the routing probabilities for a process depend on program behavior characteristics such as input-output rate, total compute time, or page fault rate. We also take into account the fact that, at the moment of a page fault, it can happen that there is no overwritable place free in real memory to contain the page to be brought in, and so the contents of some memory page must first be saved in the secondary storage. We use our model to examine the effect of memory sharing between processes on system throughput. Therefore we assume (according to experimental evidence [4]) that the page fault rate of a process depends on the amount of memory allocated to the process, i.e. in the case of balanced memory allocation we consider in this paper, on the degree of multiprogramming. The latter is defined to be the number of processes sharing real memory.

Author's address École Nationale Supérieure des Télécommunications, 46, Rue Barrault, 75634 Paris Cedex 13, France

Throughout this paper, the random variables representing the different service times considered are all assumed to be independent, exponentially distributed, and stationary. System overhead is not taken into consideration in our model. (The effect of overhead in a similar system is studied in [7].)

We study two different system organizations: In the first one (henceforth called *fixed multiprogramming*), all the processes present in the system share real core; in the second (*floating multiprogramming*), processes which have issued a file request lose their memory space until the requested I/O is performed. This type of organization has been adopted, for instance, in the ESOPE system [5].

Our model leads to a queueing network which is not a particular case of Jackson's networks [13], and this is why we apply an equivalence and decomposition approach [6]. We show that our network is equivalent, in a given sense, to a simple system whose formal analytical solution is well known. This solution contains an unknown parameter, and we compute an approximate explicit expression for it by analyzing a subnetwork of our queueing model. Having thus obtained an approximate solution for our model, we evaluate its accuracy.

1. *The Model*

The multiprogrammed, paged, virtual memory computer system we deal with in this paper consists of a CPU, a secondary memory device (SM), and a filing disk (FD) (see Figure 1(a)). A queue of requests is associated with each device; the order in which these requests are serviced is not taken into consideration.

Denote by N the total number of processes (users) executing in the system and by n_0, n_1, and n_2 the current numbers of processes queued and in service at the CPU, the SM, and the FD, respectively. At any time we have

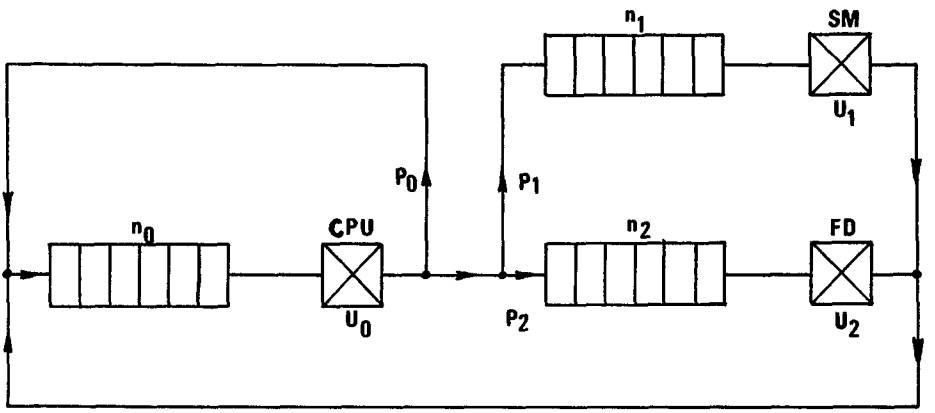

FIG 1(a)

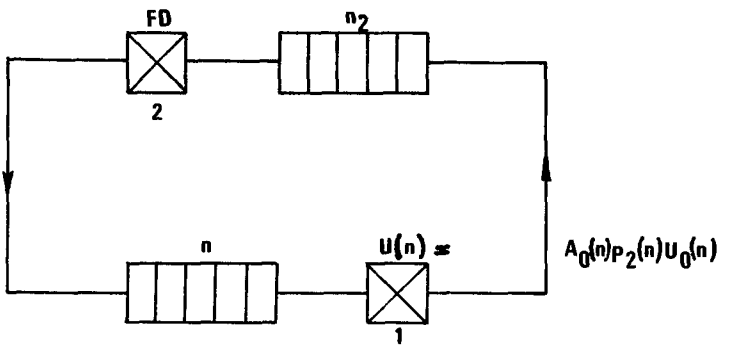

FIG 1(b)

$$N = n_0 + n_1 + n_2 = n + n_2, \tag{1}$$

n being the total current number of processes at the CPU and the SM, i.e. $n = n_0 + n_1$.

All the processes are assumed to be statistically identical and independent. In the model their behavior is characterized by a compute time followed by a page fault, in which case the process enters the SM queue; an I/O (file request), in which case the process joins the FD queue; or a program termination, in which case the process leaves the system.

It is assumed that the system operates under sufficient load conditions, and thus the feedback loop around the CPU and its queue represents processes which depart from the system but are immediately replaced by a new process, maintaining a *constant* number of users. Processes which terminate their service at the SM or the FD return into the CPU queue (see Figure 1(a)).

Let c and r be the mean total compute time for a process and the mean compute time between two successive file requests, respectively. Experimental evidence [4] indicates that the mean compute time between page faults q for a program executing in memory space m (called the lifetime function) can be approximated by a function of the form

$$q = \gamma m^k. \tag{2}$$

γ depends on the processing speed and on program characteristics while k depends on program locality as well as on the memory management strategy. According to Belady and Kuehner [4], k is in the range $1.5 \leq k \leq 2.5$.

Note that the degree of multiprogramming is equal to N (the total number of processes) in the case of "fixed" multiprogramming and to n (the current number of processes at the CPU and the SM) in the case of "floating" multiprogramming. We assume that total primary memory available is of size M and that it is equally shared among the processes (*balanced allocation*), i.e. $m = M/N$ in the first case, and $m = M/n$ in the second case. Denote by $q(n)$ the mean execution time between two successive page faults for a process when there are n processes at the CPU and the SM. We have

$$q(n) = \gamma(M/n)^k \tag{3}$$

with "floating multiprogramming" organization, and

$$q(n) = \gamma(M/N)^k$$

with "fixed multiprogramming." In the latter case, the mean compute time between page faults is actually independent of n, but we write uniformly $q(n)$ for both system organizations. Let $\nu_0 = 1/c$, $\nu_1(n) = 1/q(n)$, $n = 1, \ldots, N$, and $\nu_2 = 1/r$.

We assume that the random variable representing uninterrupted computation time at the CPU is exponentially distributed and that the service rate of the CPU is

$$u_0(n) = \nu_0 + \nu_1(n) + \nu_2$$

when a total of n processes are at the CPU and the SM.

In order to take into account the fact that not all memory pages are directly overwritable, we assume that at the moment of a page fault a preliminary transfer of a memory page into the SM is or is not required with probability β or $\alpha = 1 - \beta$, respectively.

It seems reasonable to think that probabilities α and β depend not only on program characteristics and system memory management strategy, but also on the degree of multiprogramming: Most replacement algorithms choose first a page which is directly overwritable in order to avoid increasing the SM service time, and when the number of processes sharing real core is small there should be a greater chance that a program termination will occur before all such pages have been used than with a high degree of multiprogramming. Therefore we write $\alpha(n)$ and $\beta(n)$.

We also assume that the service times at the SM and the FD are independently and exponentially distributed with expected value $1/u_2$ for the FD. We shall consider that

there are two types of SM service: types 1 and 2 corresponding to the case where there is enough space in main memory to contain the page to be brought in and the case where a page must first be saved, respectively. The mean service time of the SM is $1/\mu_1$ in the first case and $1/\mu_2$ (we assume $1/\mu_2 = 2/\mu_1$) in the second case.

We use the throughput of the system, defined as the average number of programs processed by the system per unit time in the long run, as a measure of system performance. Denote by A the *CPU utilization*, i.e. the probability of a nonempty CPU at steady state. System throughout θ is easily shown to be $\theta = A/c$, where c is the mean total compute time of a process. Thus it is sufficient to obtain the CPU utilization.

The behavior of the system at any time t is completely characterized by the joint probability distribution $p(n_0, n_1, n_2, i, t)$ of the number of processes in the three queues (n_0, n_1, n_2) and the type of SM service in progress ($i = 1, 2$) at time t; note that for $n_1 = 0$ the variable i is meaningless. As we are not going to solve the system equations directly, we shall not write them down at this stage. The state vector (n_0, n_1, n_2, i) will be used, however, later on.

Let us note that our model is not a particular case of Gordon and Newell's networks [12] (at least because of the presence of two types of service at the SM), and its solution cannot be obtained by direct application of their result.

In Section 2 we obtain an approximate solution for our model.

2. *Equivalence and Decomposition*

Let $p(n)$, $n = 0, \ldots, N$, be the stationary probability distribution of the total number of processes at the CPU and the SM ($n = n_0 + n_1$), and let $A_0(n)$ be the stationary conditional probability that the CPU is busy given n, i.e.

$$A_0(n) = \text{Prob}\{\text{CPU busy} | n \text{ processes at the CPU and the SM}\}$$

The CPU utilization A may be expressed as

$$A = \sum_{n=1}^{N} p(n)A_0(n). \tag{4}$$

Hence we see that, in order to compute our performance measure for the system, it suffices to obtain $p(n)$ and $A_0(n)$. We do so by applying an equivalence and decomposition approach.

Let us start by defining what we mean by equivalence.

Definition 2.1. Two queueing systems are *equivalent from the point of view* of a given state description (a vector of variables) if the probability distributions of the chosen state vector are identical in both systems.

We now define a particular simple queueing network.

Definition 2.2. System 1 is a cyclic queueing network consisting of two exponential servers labeled 1 and 2 (see Figure 1(b). n and n_2 denote the current numbers of customers, all statistically identical and independent, at servers 1 and 2, respectively; we have $n + n_2 = N$, the constant total number of customers circulating in the system. The service rates are $u(n) = A_0(n)v_2$, $n = 1, \ldots, N$ for server 1 and u_2, $n_2 = 1, \ldots, N$ for server 2.

We have

THEOREM 2.1. *The queueing model described in Section* 1 *is equivalent from the point of view of the variable n, at the stationary state, to System* 1.

The proof of this theorem is similar to equivalence proofs of [6] and will be omitted.

Using Theorem 2.1 we easily obtain an expression for the stationary probability of $n = n_0 + n_1$:

$$p(n) = (1/H)\left(u_2^n \Big/ \prod_{j=1}^{n} u(j)\right), \quad n = 0, 1, \ldots, N, \tag{5}$$

where H is a normalization constant. Equation (5) may be rewritten as

$$p(n) = (1/H)\left((u_2 r)^n \Big/ \prod_{j=1}^{n} A_0(j)\right), \quad n = 0, 1, \ldots, N; \quad H = \sum_{n=0}^{N}\left((u_2 r)^n \Big/ \prod_{j=1}^{n} A_0(j)\right), \tag{6}$$

a well-known and computationally efficient solution form. Equation (6), however, contains an unknown parameter (which we need also if we want to apply (4)): the conditional probability $A_0(n)$. Therefore we seek a means for computing, at least approximately, $A_0(n)$.

Suppose that the rates of transitions due to paging ($v_1(n)$, μ_1, μ_2) are much greater than the rates of transitions due to file requests (v_2, u_2). Then it is intuitively clear that the subnetwork composed of the CPU and the SM reaches, on the average, its steady state relatively rapidly between two successive changes in the total number of users in it n. Hence $A_0(n)$ is not much different from the CPU utilization in a closed cyclic two-server network consisting of the CPU and the SM, obtained by setting $v_2 = 0$ and $u_2 = 0$, with a total of n processes (this system will be referred to as *System* 2). Denoting by A_0^n the CPU utilization in System 2 with n users, we have

$$A_0(n) \approx A_0^n. \tag{7}$$

More generally, the joint probability distribution $p(n_0, n_1, i)$ ($i = 1, 2$ indicates the type of SM service in progress) in System 2 is approximately equal to the conditional probability of having (n_0, n_1, i) given n in the model of Section 1, $p(n_0, n_1, i \mid n)$:

$$p(n_0, n_1, i \mid n) \approx p(n_0, n_1, i).$$

Now it is obvious that the rates of transitions due to file requests may not be much smaller than those due to paging, so (7) may seem inapplicable. Nevertheless, we apply it, and in Section 4 we analyze the accuracy of the approximation resulting from this decomposition. As we shall see, the accuracy turns out to be excellent, and its analysis provides more insight into the decomposability of our model.

Section 3 is devoted to the computation of A_0^n, i.e. to the study of System 2, which is interesting per se because it takes into account the paging behavior of virtual memory.

3. *Two-Server System*

3.1 System Equations. Consider System 2 with a total of n processes executing in it. $p(n_0, n_1, i)$ is the stationary probability that there are n_0 and $n_1 = n - n_0$ processes at the CPU and the SM, respectively, and that the current service type of the SM is i; let

$$f_{n_0}^n = p(n_0, n_1, 1), \quad n_0 = 0, \ldots, n-1;$$

$$g_{n_0}^n = p(n_0, n_1, 2), \quad n_0 = 0, \ldots, n-1;$$

$$f_n^n = \alpha(n) p(n_0 = n, n_1 = 0);$$

$$g_n^n = \beta(n) p(n_0 = n, n_1 = 0).$$

Using the fact that under equilibrium, for any n_0, the rate of arrivals of processes to the CPU equals the rate of departures of processes from the CPU, and that the ratio of type 1 over type 2 SM service requests generated is $\alpha(n)/\beta(n)$, we obtain the following set of equations:

$$\begin{aligned} -\mu_1 f_1^0 + v_1(1) f_1^1 &= 0; \\ -\mu_2 g_1^0 + v_1(1) g_1^1 &= 0; \\ \beta(1) f_1^1 &= \alpha(1) g_1^1 \end{aligned} \tag{8}$$

for $n = 1$, and

$$-\mu_1 f_n^0 + v_1(n) f_n^1 = 0; \tag{9}$$

$$-\mu_2 g_n^0 + \nu_1(n) g_n^1 = 0; \tag{10}$$

$$-\mu_1 f_{n_0}^n - \mu_2 g_{n_0}^n + \nu_1(n)[f_{n_0}^{n_0+1} + g_{n_0}^{n_0+1}] = 0, \quad n_0 = 1, \ldots, n-1; \tag{11}$$

$$\beta(n)[-(\mu_1 + \nu_1(n)) f_{n_0}^n + \nu_1(n) f_{n_0}^{n_0+1}] = \alpha(n)[-(\mu_2 + \nu_1(n)) g_{n_0}^n + \nu_1(n) g_{n_0}^{n_0+1}],$$

$$n_0 = 1, \ldots, n-1; \tag{12}$$

$$\beta(n) f_n^n = \alpha(n) g_n^n, \tag{13}$$

for $n = 2, 3, \ldots$.

Note that this set of equations can also be easily obtained from the system balance equations and that condition (13) is necessary because of the notation f_n^n and g_n^n.

The solution of eqs. (8) is straightforward, so we shall now turn to the solution of system equations for $n \geq 2$. The latter may be viewed as simultaneous homogeneous linear difference equations with two unknown functions $f_{n_0}^n$ and $g_{n_0}^n$, where (11) and (12) are the general equations and (9), (10), and (13) the boundary equations.

Denote by E the operator of displacement, i.e. $Ef(x) = f(x+1)$. Equations (11) and (12) can be rewritten in a symbolic form:

$$\Phi(n) f_{n_0}^n + \Psi(n) g_{n_0}^n = 0, \quad \Phi_1(n) f_{n_0}^n + \Psi_1(n) g_{n_0}^n = 0, \tag{14}$$

where

$$\Phi(n) = -\mu_1 + \nu_1(n)E, \quad \Psi(n) = -\mu_2 + \nu_1(n)E;$$

$$\Phi_1(n) = \beta(n)[-(\mu_1 + \nu_1(n)) + \nu_1(n)E], \quad \Psi_1(n) = \alpha(n)[\mu_2 + \nu_1(n) - \nu_1(n)E].$$

Using a difference equations technique (see [14, p. 601], we eliminate one of the unknown functions (e.g. $f_{n_0}^n$) from (14), thus obtaining

$$[\Phi_1(n)\Psi(n) - \Phi(n)\Psi_1(n)] g_{n_0}^n = 0,$$

i.e.

$$[\nu_1(n)]^2 g_{n_0}^{n_0+2} + \nu_1(n)[\mu_1 + \mu_2 + \nu_1(n)] g_{n_0}^{n_0+1} + [\mu_1\mu_2 + \nu_1(n)(\alpha(n)\mu_1 + \beta(n)\mu_2)] g_{n_0}^{n_0} = 0. \tag{15}$$

Equation (15) is a simple linear homogeneous equation with constant coefficients. Its solution can be written (see [14]) as

$$g_{n_0}^n = C_1(n)[r_1(n)]^{n_0} + C_2(n)[r_2(n)]^{n_0}, \quad n_0 = 1, \ldots, n, \tag{16}$$

where

$$r_{1,2}(n) = \{\mu_1 + \mu_2 + \nu_1(n) \pm \sqrt{[[\nu_1(n)]^2 + (\mu_2 - \mu_1)(2\nu_1(n)(\alpha(n) - \beta(n)) + \mu_2 - \mu_1)]}\}/2\nu_1(n)$$

are the two roots of the characteristic equation

$$[\nu_1(n)]^2 r^2 + \nu_1(n)[\mu_1 + \mu_2 + \nu_1(n)] r + \mu_1\mu_2 + \nu_1(n)[\alpha(n)\mu_1 + \beta(n)\mu_2] = 0,$$

and $C_1(n)$ and $C_2(n)$ are "arbitrary constants" (to be determined from boundary and probability conditions).

It follows from (14) that

$$f_{n_0}^n = \xi_1(n) C_1(n)[r_1(n)]^{n_0} + \xi_2(n) C_2(n)[r_2(n)]^{n_0}, \quad n_0 = 1, \ldots, n, \tag{17}$$

where

$$\xi_j(n) = -\Psi(n)_{E=r_j(n)}/\Phi(n)_{E=r_j(n)} = -[\nu_1(n) r_j(n) - \mu_2]/[\nu_1(n) r_j(n) - \mu_1], \quad j = 1, 2. \tag{18}$$

$C_1(n)$ and $C_2(n)$ are then easily determined by using the boundary equations (9), (10),

and (13) and the condition that the obtained solution must be a probability distribution, i.e.

$$\sum_{n_0=1}^{n} [f_{n0}^{n} + g_{n0}^{n}] = 1.$$

We have

$$C_1(n) = 1/[A_1(n) + A_2(n)B(n)], \quad C_2(n) = B(n)C_1(n), \tag{19}$$

where

$$A_j(n) = (1 + \xi_j(n))\{r_j(n) - [r_j(n)]^{n+1}\}/(1 - r_j(n)) + \nu_1(n)r_j(n)[\xi_j(n)/\mu_1 + 1/\mu_2], \quad j = 1, 2,$$

$$B(n) = [r_1(n)]^n[\beta(n)\xi_1(n) - \alpha(n)]/[r_2(n)]^n[\alpha(n) - \beta(n)\xi_2(n)].$$

Thus, finally, f_{n0}^{n} and g_{n0}^{n} ($n \geq 2$) are given, for $n_0 = 1, \ldots, n$, by (17) and (16) with (18) and (19); for $n_0 = 0$ we have

$$f_n^0 = \nu_1(n)f_n^1/\mu_1, \quad g_n^0 = \nu_1(n)g_n^1/\mu_2.$$

A_0^n, the CPU utilization in System 2, is obtained as

$$A_0^n = 1 - [f_n^0 + g_n^0],$$

and there is no computational difficulty in evaluating this expression. For $n = 1$, we have

$$f_1^0 = \nu_1(1)/(\mu_1 G_1), \quad g_1^0 = \nu_1(1)\beta(1)/(\alpha(1)\mu_2 G_1);$$

$$f_1^1 = 1/G_1, \quad g_1^1 = \beta(1)/(\alpha(1)G_1);$$

$$A_0^1 = 1 - (f_1^0 + g_1^0);$$

$$G_1 = 1 + \nu_1(1)/\mu_1 + \beta(1)[1 + \nu_1(1)/\mu_2]/\alpha(1).$$

In Section 3.2 we present some numerical results illustrating the behavior of System 2.

3.2 Numerical Results with the Two-server Model. Numerical results obtained from the two-server model with lifetime function (3) are reported in Figure 2, which shows the effect of the degree of multiprogramming on CPU utilization (and hence on throughput). The probability that at the moment of a page fault there is no overwritable page free, $\beta(n)$, is assumed in this example to vary linearly with n: $\beta(n) = a + bn$; a and b are set to $a = -0.5/9$ and $b = 0.5/9$ so that $\beta(n)$ rises from zero for one process in memory to 0.5 when the degree of multiprogramming equals 10. This is an arbitrary assumption used in the numerical example to show what happens if $\beta(n)$ varies in this way; note that no assumption on the form of $\beta(n)$ has been made in the solution of System 2.

The curves labeled 1, 2, 3, and 4 correspond to the following sets of model parameters: 1: $M = 128$ pages, $1/\mu_1 = 5$ msec, $k = 1.5$; 2: $M = 256$ pages, $1/\mu_1 = 5$ msec, $k = 1.5$; 3: $M = 128$ pages, $1/\mu_1 = 5$ msec, $k = 2.0$; 4: $M = 128$ pages, $1/\mu_1 = 10$ msec, $k = 2.0$. γ in (3) is set to 0.01.

We see the important effect of the primary memory size and of the mean service time of the paging SM device ($t_{SM} = 1/\mu_1$) on system performance. We also see that an improvement in program behavior (increasing the locality exponent k from 1.5 to 2.0) can produce a considerable improvement of the throughput as well as an increase in the optimal (i.e. which maximizes system throughput) degree of multiprogramming.

Section 3.3 is devoted to the problem of determining analytically the optimum degree of multiprogramming in the case where the existence of pages not directly overwritable is neglected.

3.3 Optimal Degree of Multiprogramming in a Restricted Case. We now consider System 2 assuming that $\beta(n) = 0$ for all n, and we examine the problem of

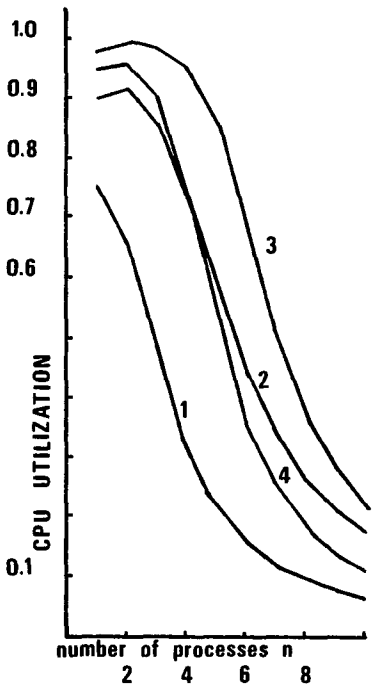

FIG 2

determining analytically an expression for the degree of multiprogramming N_0 which maximizes system throughput. This seems to be a difficult problem in a general case, and even for the restricted case we are considering, we are only able to obtain an approximate value.

The assumption $\beta(n) = 0$ reduces System 2 to a simple finite source M/M/1 model. CPU utilization is obtained (see [9]) as $A_0^n = 1/(1 + y)$, where $y = z^n(1 - z)/(1 - z^n)$ and $z = n^k/(\gamma u_1 M^k)$.

M is the primary memory size, k and γ are the parameters of the Belady lifetime function ((3)), and $1/u_1$ is used to denote the now unique mean service time of the SM.

For convenience let $d = 1/(\gamma u_1 M^k)$. Clearly maximizing A_0^n is equivalent to minimizing y. Consider the case $1/u_1 \ll \gamma M^k/n^k$, where the mean compute time between two successive page faults of a process is much larger than the mean service time of the SM device We then have $z \ll 1$, and therefore $y \approx y_1 = z^n$. Considering n as a continuous variable and taking $dy_1/dn = 0$, we see that y_1 is minimized by

$$N_1 = (1/d)^{1/k}/e, \tag{20}$$

where e is the basis of the natural logarithms.

In various numerical examples (Figures 3-6), we see that N_1 is a good estimate of N_0, the "optimum" degree of multiprogramming. Note that even when (20) is not well satisfied, N_1 still seems close to N_0, probably because $(1 - z)$ is relatively "flat" as compared to $z^n/(1 - z^n)$.

Let us note that the applicability of Formula (20) can be extended to include the case where the probability $\beta(n)$ has some constant value, say β_0, for all n, if, instead of considering explicitly two types of service at the SM, we assume an exponentially distributed SM service time of mean

$$1/\mu_1 = (1 - \beta_0)/\mu_1 + \beta_0/\mu_2.$$

It is worth mentioning that this approach yields, as regards system throughput,

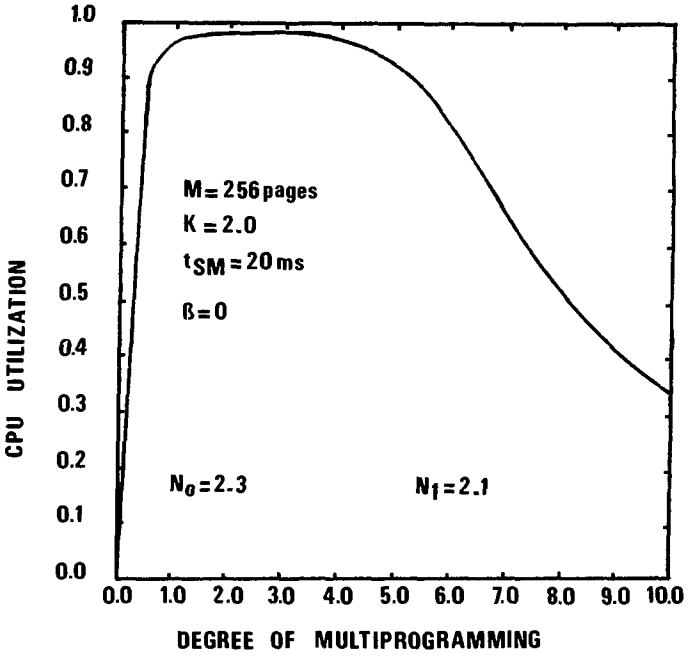

FIG 3

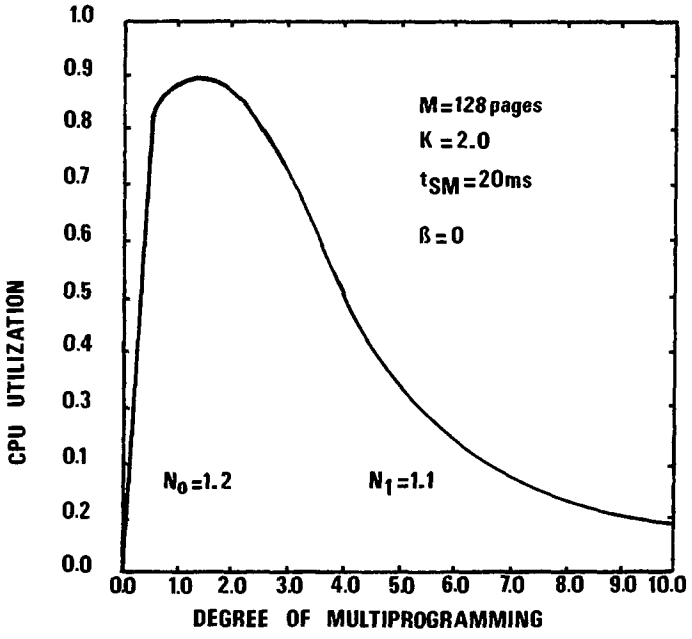

FIG 4

numerical results very close (the CPU utilization seems slightly greater near the optimum degree of multiprogramming and practically identical for other values of the multiprogramming degree) to those obtained with the explicit consideration of two types of exponential SM service times. This may be regarded as a sign of a relative "robustness" of our model vis-à-vis distributional assumptions.

Having obtained and studied in this section the CPU utilization in System 2, A_0^n, we

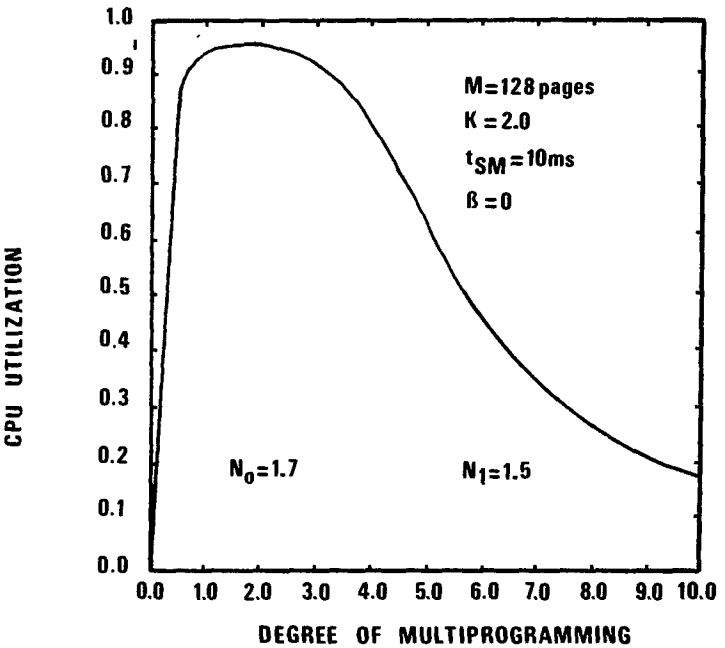

FIG 5

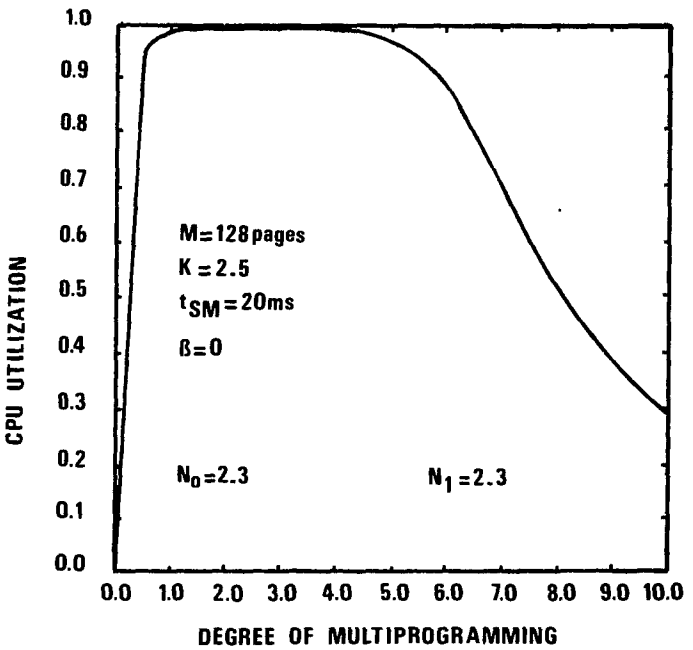

FIG 6

address the problem of the accuracy of the approximation which is introduced if we use A_0^n for $A_0(n)$ in (6). This will be done in Section 4.

4. *Accuracy of the Approximation*

First of all, note that the only approximation in the method proposed in Section 2 for computing $p(n)$ stems from the decomposition, i.e. from (7). Therefore our goal will be

primarily to determine how different is A_0^n from the conditional probability $A_0(n)$.

Recall that the behavior of the model of Section 1 is completely characterized by the joint probability distribution $p(n_0, n_1, n_2, i)$, where i indicates the type of the SM service in progress. Let (remember (1))

$$f_n(n_0) = \text{Prob}\{(n_0, n_1, 1)|n\}, \quad n_0 = 0, \ldots, n-1; \tag{21}$$
$$g_n(n_0) = \text{Prob}\{(n_0, n_1, 2)|n\}, \quad n_0 = 0, \ldots, n-1; \tag{22}$$
$$f_n(n) = \alpha(n)\ \text{Prob}\{n_0 = n|n\}; \tag{23}$$
$$g_n(n) = \beta(n)\ \text{Prob}\{n_0 = n|n\}. \tag{24}$$

We have

$$A_0(n) = 1 - [f_n(0) + g_n(0)].$$

Using the fact that

$$p(n_0, n_1, n_2, i) = p(n)\ \text{Prob}\{(n_0, n_1, i)|n\},$$

the formal solution for $p(n)$ (eq. (6)), and (21)–(24) in the balance equations for our model, we easily obtain a set of equations for the conditional probabilities $f_n(n_0)$, $g_n(n_0)$, and $A_0(n)$. These equations are given in Appendix 1.

When applying the decomposition, we use probabilities $f_{n^0}^{n_0}$ and $g_{n^0}^{n_0}$ in System 2 for $f_n(n_0)$ and $g_n(n_0)$, i.e. we use the probabilities of having (n_0, i) in System 2 with a total of n processes in lieu of the conditional probabilities of having (n_0, i) given n in our model. Denote by $\epsilon_n(n_0)$ and $\eta_n(n_0)$ the corresponding errors. We have

$$f_n(n_0) = f_{n^0}^{n_0} + \epsilon_n(n_0), \quad g_n(n_0) = g_{n^0}^{n_0} + \eta_n(n_0), \quad n_0 = 0, \ldots, n; \tag{25}$$
$$A_0(n) = A_0^n - \epsilon_n(0) - \eta_n(0). \tag{26}$$

Since both $f_n(n_0)$, $g_n(n_0)$ and $f_{n^0}^{n_0}$, $g_{n^0}^{n_0}$ are probability distributions, i.e. normalized with respect to unity, we also have

$$\sum_{n_0=0}^{n} [\epsilon_n(n_0) + \eta_n(n_0)] = 0, \quad \text{for } n = 1, \ldots, N. \tag{27}$$

Substituting (25) and (26) into the equations of Appendix 1 and neglecting higher order error terms (i.e. products of two or more errors), we obtain the following set of equations:

$$A_0^{n+1}[-(\mu_1 + u_2)\epsilon_n(0) + \nu_1(n)\epsilon_n(1)] + u_2\epsilon_{n+1}(1) + u_2 f_n^0[\epsilon_{n+1}(0) + \eta_{n+1}(0)] = Q_{1n}^0, \tag{28}$$
$$A_0^{n+1}[-(\mu_2 + u_2)\eta_n(0) + \nu_1(n)\eta_n(1)] + u_2\eta_{n+1}(1) + u_2 g_n^0[\epsilon_{n+1}(0) + \eta_{n+1}(0)] = Q_{2n}^0 \tag{29}$$

for $n = 1, \ldots, N-1$, where

$$Q_{1n}^0 = u_2(A_0^{n+1} f_n^0 - f_{n+1}^1), \quad Q_{2n}^0 = u_2(A_0^{n+1} g_n^0 - g_{n+1}^1), \tag{30}$$

$$\begin{aligned} A_0^{n+1}\{&-[\nu_1(n) + \mu_1 + \nu_2 + u_2]\epsilon_n(n_0) + \alpha(n)\mu_1\epsilon_n(n_0 - 1) + \alpha(n)\mu_2\eta_n(n_0 - 1) \\ &+ \nu_1(n)\epsilon_n(n_0 + 1) + \nu_2 A_0^n\,\epsilon_{n-1}(n_0 - 1) - \nu_2 f_{n-1}^{n_0-1}[\epsilon_n(0) + \eta_n(0)]\} \\ &+ [\epsilon_{n+1}(0) + \eta_{n+1}(0)][(u_2 + \nu_2)f_{n^0}^{n_0} - \nu_2 A_0^n f_{n-1}^{n_0-1}] + u_2\epsilon_{n+1}(n_0 + 1) = Q_{1n}^{n_0} \end{aligned} \tag{31}$$

for $n = 2, \ldots, N-1$, $n_0 = 1, \ldots, n-1$, and an analogous equation for $\eta_n(n_0)$, where

$$\begin{aligned} Q_{1n}^{n_0} &= u_2(A_0^{n+1} f_{n^0}^{n_0} - f_{n+1}^{n_0+1}) + \nu_2 A_0^{n+1}(f_{n^0}^{n_0} - A_0^{n+1} f_{n-1}^{n_0-1}), \\ Q_{2n}^{n_0} &= u_2(A_0^{n+1} g_{n^0}^{n_0} - g_{n+1}^{n_0+1}) + \nu_2 A_0^{n+1}(g_{n^0}^{n_0} - A_0^{n+1} g_{n-1}^{n_0-1}), \end{aligned} \tag{32}$$

$$-\mu_1\epsilon_N(0) + \nu_1(N)\epsilon_N(1) = 0, \tag{33}$$
$$-\mu_2\eta_N(0) + \nu_1(N)\eta_N(1) = 0, \tag{34}$$

$$\begin{aligned} &-[\nu_1(N) + \mu_1 + \nu_2]\epsilon_N(n_0) + \alpha(N)\mu_1\epsilon_N(n_0 - 1) + \alpha(N)\mu_2\eta_N(n_0 - 1) \\ &\quad + \nu_1(N)\epsilon_N(n_0 + 1) + \nu_2 A_0^N\epsilon_{N-1}(n_0 - 1) - \nu_2 f_{N-1}^{n_0-1}[\epsilon_N(0) + \eta_N(0)] = Q_{1N}^{n_0}, \end{aligned} \tag{35}$$

$$\begin{aligned} &-[\nu_1(N) + \mu_2 + \nu_2]\eta_N(n_0) + \beta(N)\mu_1\epsilon_N(n_0 - 1) + \beta(N)\mu_2\eta_N(n_0 - 1), \\ &\quad + \nu_1(N)\eta_N(n_0 + 1) + \nu_2 A_0^N\eta_{N-1}(n_0 - 1) - \nu_2 g_{N-1}^{n_0-1}[\epsilon_N(0) + \eta_N(0)] = Q_{2N}^{n_0} \end{aligned} \tag{36}$$

for $n = N$ and $n_0 = 1, \ldots, N-1$, where

$$Q_{1N}^{n_0} = v_2[f_N^{n_0} - A_0^N f_{N-1}^{n_0-1}], \quad Q_{2N}^{n_0} = v_2[g_N^{n_0} - A_0^N g_{N-1}^{n_0-1}]. \tag{37}$$

In order to evaluate the errors $\epsilon_n(n_0)$ and $\eta_n(n_0)$, we have to solve this system of equations subject to condition (27) and

$$\beta(n)\epsilon_n(n) = \alpha(n)\eta_n(n), \tag{38}$$

which follows from (23) and (24). Before indicating how this can be done in an efficient way, we formulate a few remarks.

The coefficients of the system of equations are all known since they are either the parameters of our model or state probabilities of System 2. Equations (28), (29), and (31) and its analogue are obtained by neglecting higher order terms introduced by expressions of the form

$$[A_0^n - \epsilon_n(0) - \eta_n(0)][f_{n^0}^{n_0} + \epsilon_n(n_0)].$$

Hence, in order to validate the solution, it suffices to verify a posteriori that it satisfies the condition

$$A_0^n \gg |\epsilon_n(0) + \eta_n(0)|. \tag{39}$$

Owing to condition (27), the unique solution of our system[1] would be $\epsilon_n(n_0) = \eta_n(n_0) = 0$ for all n and n_0 if the $Q_{1n}^{n_0}$ and $Q_{2n}^{n_0}$ were all zeros. Thus the $Q_{1n}^{n_0}$, $Q_{2n}^{n_0}$ reflect the error introduced by the decomposition; the smaller these terms, the smaller the error. An inspection of (30), (32), and (37) reveals the very interesting form of the $Q_{in}^{n_0}$, $i = 1, 2$: a sum of products of two terms, one corresponding to the rates of transitions which change n (v_2, u_2), and the second depending only on the internal properties of System 2. It then becomes obvious that there are two entirely distinct reasons for which our system may be decomposable. The first is when v_2 and u_2 are small—this corresponds to the case of intuitive decomposability presented in Section 1. The second is when

$$|f_n^{n_0} - A_0^n f_{n-1}^{n_0-1}| \quad \text{and} \quad |g_n^{n_0} - A_0^n g_{n-1}^{n_0-1}| \tag{40}$$

are small. Note that (40) reflect the departure of our model from a queueing network with a product form solution [9, 13, 12], since these terms would be zero if our model had such a solution. This would be the case, for instance, if $\mu_1 = \mu_2$ with "fixed multiprogramming" organization for any nontrivial values of v_2 and u_2 (see Appendix 2). Incidentally it is this second reason of decomposability that explains zero error results when the equivalence and decomposition method is applied to central server networks [9] and, also, for our model when $N = 1$.

Let us now tackle the problem of finding an efficient solution procedure for the system of eqs. (28)–(36), together with (27) and (38). This system can be written in a matrix form as

$$TE = Q, \tag{41}$$

where E is the vector of the errors, Q is the vector of the $Q_{1n}^{n_0}$ and $Q_{2n}^{n_0}$, and T is the matrix of coefficients. Note that neglecting in eqs. (28)–(36) the terms which involve $n + 1$, we obtain N sets of simple recurrence relations which can easily be solved separately for each $n = 1, \ldots, N$ (owing to the independent normalization conditions (27)) if we start from $n = 1$. This suggests the following iterative procedure. Denote by T' the matrix of coefficients obtained by neglecting the terms in $n + 1$. In order to compute E_0, a first approximation to E, we solve at the first iteration

$$T'E_0 = Q \tag{42}$$

Then consecutive residual terms for E are computed by solving, at iteration k,

$$T'E_k = (T' - T)E_{k-1} \quad \text{for } k = 1, 2, \ldots . \tag{43}$$

[1] The equations are of full rank

Note that (42) and (43) are sets of simple linear recurrence relations solved separately for each $n = 1, \ldots, N$. Assuming this procedure converges, we have

$$E = \sum_{k=0}^{\infty} E_k.$$

This last statement is easy to prove. Indeed let

$$\hat{E} = \sum_{k=0}^{\infty} E_k. \tag{44}$$

Summing (43) over $k = 1, \ldots,$ we obtain

$$T' \sum_{k=1}^{\infty} E_k = (T' - T) \sum_{k=0}^{\infty} E_k,$$

i.e. using (44), $T'E_0 = T\hat{E}$, which, given (42), yields

$$T\hat{E} = Q. \tag{45}$$

Since the linear system (41) has exactly one solution (conditions (27) and (38) are taken into account in T, T', and Q), it follows from (45) that $\hat{E} = E$.

Let us now consider briefly the convergence of our iterative procedure. The solution of (43) at iteration k is essentially equivalent to the solution of

$$S_k = T'^{-1}(T' - T)(S_{k-1} + E_0), \tag{46}$$

where $S_k = \sum_{i=1}^{k} E_i$. It is well known from the theory of iterative methods in numerical analysis that (46) converges if and only if

$$\|S_{k+1} - S_k\| \leq L \|S_k - S_{k-1}\|, \quad \text{with } L < 1.$$

In our case,

$$\|S_{k+1} - S_k\| = \|T'^{-1}(T' - T)(S_k - S_{k-1})\| \leq \|T'^{-1}(T' - T)\| \|S_k - S_{k-1}\|;$$

so

$$\|T^{-1}(T' - T)\| = m < 1$$

is a sufficient condition for the convergence of our procedure. Quite intuitively then, our procedure will converge if the coefficients of the terms involving $n + 1$ (directly proportional to v_1, u_2) are not too large as compared to the coefficients of T taken into account in T'. An upper bound for L may be obtained by considering the Jacobi matrix for the function $T'^{-1}(T' - T)$. Hence sufficient (but not necessary) convergence conditions may be derived for our iterative procedure.

The interested reader will find in [8] more details on this procedure and on its convergence. Let us say here only that, for the values of system parameters explored in this paper, the method converges very rapidly (one or two iterations in most instances, and no more than six iterations in any case for an accuracy greater than 10^{-3}). In Appendix 3 we give the recurrence relations corresponding to (42) and (43).

Note that we have been able to evaluate not only the difference between $A_0(n)$ and A_0^n, which was our goal, but also more generally the error introduced when we use the probabilities $f_{n_0}^{n_0}$, $g_{n_0}^{n_0}$ in lieu of the conditional probabilities $f_n(n_0)$, $g_n(n_0)$. Note also that our approach yields the errors with their signs (and not just a bound for the absolute value), thus indicating whether the corresponding probability is under- or overestimated.

In Section 5 we present the numerical results obtained from our model of Section 1.

5. *Numerical Results of the Model*

Before discussing the numerical results of the complete model described in Section 1, we recall the approach used to solve it. First we have obtained, using the equivalence

Theorem 2.1, an expression for the probability of having a total of n processes at the CPU and the SM. Then, using the decomposition, we have obtained A_0^n as an approximation for $A_0(n)$, which was the unknown parameter of (6), and which we needed to compute the CPU utilization A by (4). Note that the equivalence and decomposition approach in fact yields more. Knowing $f_{n_0}^n$ and $g_{n_0}^n$ from the analysis of System 2, we know an approximate solution for the detailed state of our model:

$$p(n_0, n_1, n_2, 1) \approx p(n) f_{n_0}^n, \quad p(n_0, n_1, n_2, 2) \approx p(n) g_{n_0}^n,$$

so performance measures other than A (like mean queue lengths) can also be computed.

Figures 7–13 show the numerical results obtained from our model: The CPU utilization A is plotted versus the total number of processes N. Throughout the examples, γ of (2) is kept constant at 0.01. As in Figure 2, in Figures 7–12 $\beta(n)$ is assumed to vary linearly with the degrees of multiprogramming, from 0 for one process in memory up to 0.5 for ten processes sharing primary memory.

The influence of the primary memory size M is illustrated in Figures 7 and 8, in which a set of system parameters with $M = 128$ pages and $M = 256$ pages, respectively, is used. We note a marked increase of the CPU utilization and of the optimal total number of processes (which is also the degree of multiprogramming in the case of "fixed multiprogramming") for both system organizations considered. In Figures 8–10 we examine the effect of varying the parameter k of the lifetime function (2) (corresponding, to some extent, to program locality). As for System 2, this appears to be an important parameter: An improvement in program locality (increasing k from 1.5 to 2.5) can result in a considerable increase of the optimal degree of multiprogramming and of the system throughout. The fact that the mean service time of the paging device ($t_{SM} = 1/\mu_1$) can also importantly affect system performance is illustrated in Figures 11 and 9 ($t_{SM} = 10$

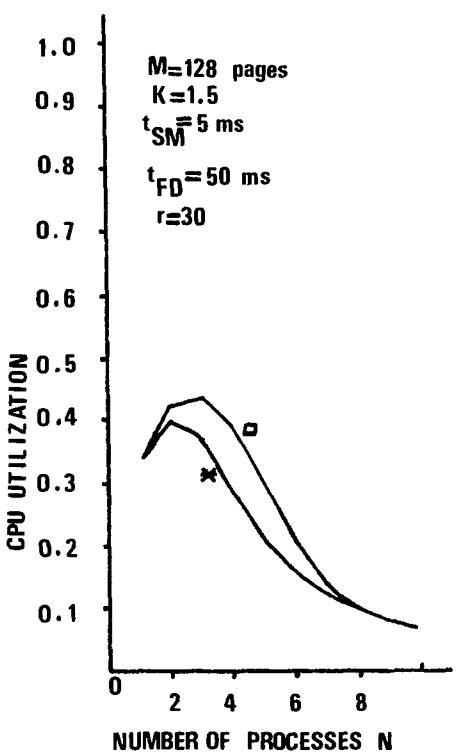

FIG 7

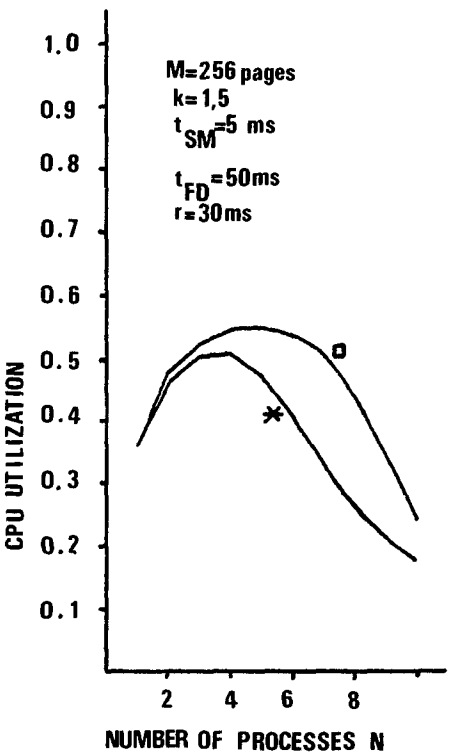

FIG 8

□ = floating multiprogramming, * = fixed multiprogramming

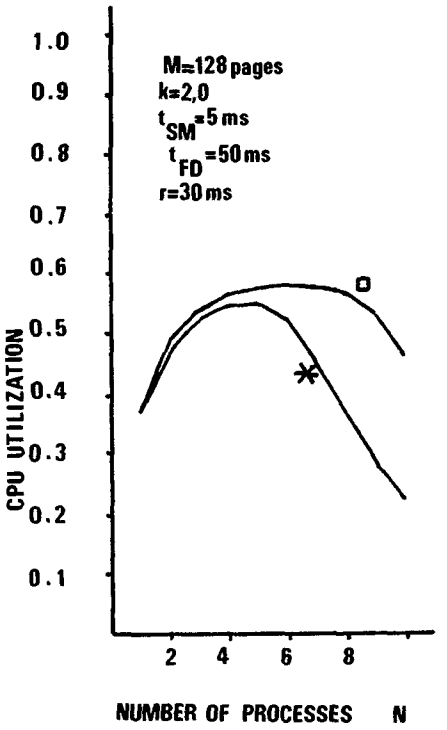

FIG 9

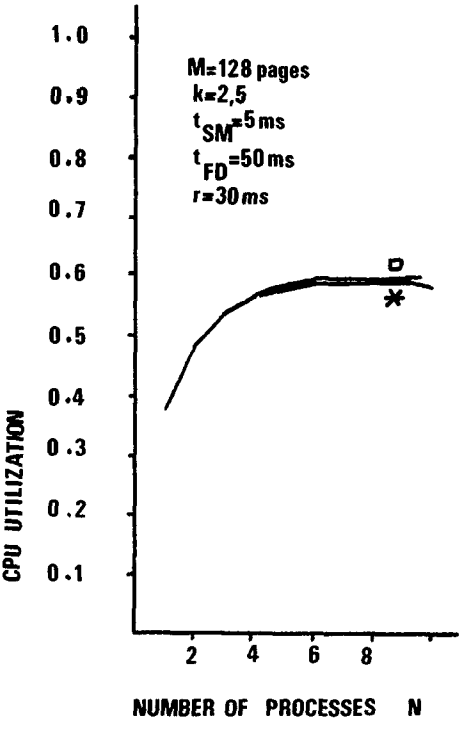

FIG. 10

□ = floating multiprogramming, * = fixed multiprogramming

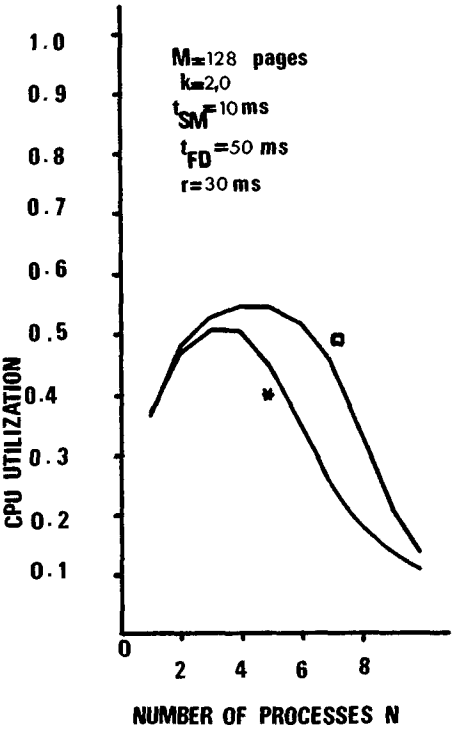

FIG. 11

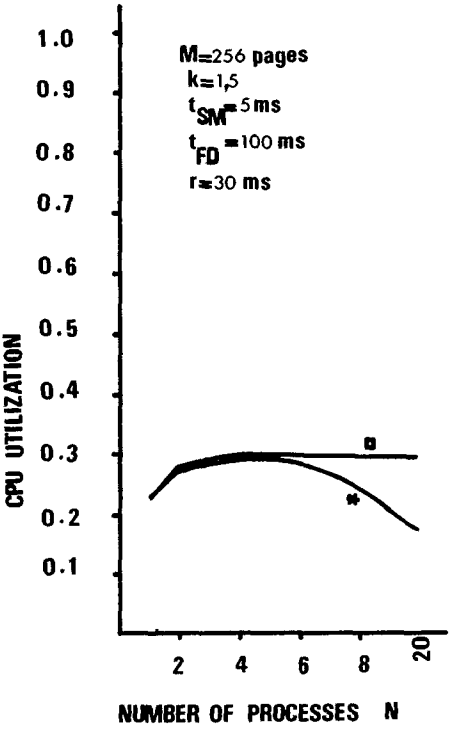

FIG. 12

□ = floating multiprogramming, * = fixed multiprogramming

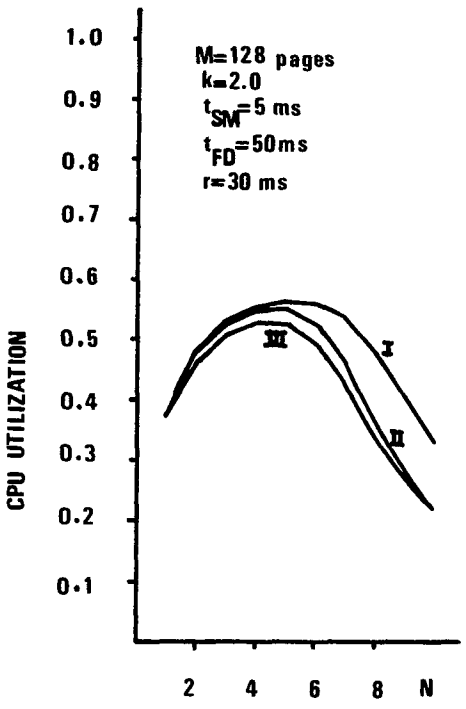

FIG. 13

msec and t_{SM} = 5 msec, respectively). Figures 8 and 12 show the influence of the mean service time of the filing disk ($t_{FD} = 1/u_2$). For the values of model parameters used in Figure 12, the system is clearly "input/output bound."

We observe, as in [7], that increasing the primary memory size or the program locality or decreasing the mean service time of the paging device reduces the sensitivity of system throughput to the number of users.

Finally, in Figure 13 we study the influence of the probability $\beta(n)$ in the case of "fixed multiprogramming" system organization. Curves labeled I, II, and III correspond to a set of system parameters with I: $\beta = 0$, $N = 1, \ldots, 10$; II: $\beta = 0.5N/9 - 0.5/9$; III: $\beta = 0.5$, $N = 1, \ldots, 10$.

We observe that the presence of pages which are not directly overwritable and, more generally, the form of the function $\beta(n)$ may have a considerable effect on system throughput and on the optimal degree of multiprogramming. Therefore it is interesting to obtain measurement results showing the form of $\beta(n)$.

If we compare the figures obtained for the two system organizations considered, we note that "floating multiprogramming" results in significantly higher thrashing threshold and also in higher CPU utilization than "fixed multiprogramming." The results obtained, however, should be regarded as an optimistic estimate of the performance of the "floating multiprogramming" system organization, since our model does not account for the mechanism by which a process that has completed a file operation acquires memory pages.

Using the iterative procedure developed in Section 4, we have evaluated the accuracy of our results. Condition (39) turns out to be well satisfied; in most instances the $|\epsilon_n(0) + \eta_n(0)|$ are much smaller than 10 percent of the corresponding A_0^n. As expected, the accuracy is slightly better in the case of "fixed multiprogramming" (the only departure from a product-form solution is then due to two types of service at the SM). The relative error on A (the CPU utilization) is negligible, less than 1 percent in all cases. In Appendix 4 we give the evaluated errors in one of the worst instances.

6. *Conclusion*

We have presented a queueing network model of a virtual memory multiprogramming system.

We have been able to obtain an approximate, computationally efficient, closed form solution by using equivalence and decomposition methods. We have also been able to evaluate the accuracy of the approximation and to determine that there are two totally distinct reasons for which the decomposition can be used in the queueing network representing our model. The first can be invoked if the rates of transitions between the subnetwork obtained by decomposition and the remainder of the system are small (this corresponds to the often used intuitive argument of the subnetwork having the time to reach its steady state between two interactions with the remainder of the system). The second depends only on internal properties of the subnetwork and reflects, to some extent, the departure of our model from a queueing system with a product form solution.

The approach used to determine the accuracy can be applied to other models solved by the equivalence and decomposition method.

Models of time-sharing systems, for which our network can represent the processing part, may be analyzed in a similar way.

Appendix 1

The equations for the conditional probabilities $f_n(n_0)$, $g_n(n_0)$ are as follows:

$$A_0(n+1)[-(\mu_1+\mu_2)f_n(0)+\nu_1(n)f_n(1)]+u_2f_{n+1}(1)=0,$$
$$A_0(n+1)[-(\mu_2+\mu_2)g_n(0)+\nu_1(n)g_n(1)]+u_2g_{n+1}(1)=0$$
$$\text{for } n=1,\dots,N-1;$$

$$A_0(n+1)\{-[\nu_1(n)+\mu_1+\nu_2+u_2]f_n(n_0)+\alpha(n)\mu_1f_n(n_0-1)+\alpha(n)\mu_2g_n(n_0-1)+\nu_1(n)f_n(n_0+1)+\nu_2A_0(n)f_{n-1}(n_0-1)\}+u_2f_{n+1}(n_0+1)=0,$$

$$A_0(n+1)\{-[\nu_1(n)+\mu_2+\nu_2+u_2]g_n(n_0)+\beta(n)\mu_1f_n(n_0-1)+\beta(n)\mu_2g_n(n_0-1)+\nu_1(n)g_n(n_0+1)+\nu_2A_0(n)g_{n-1}(n_0-1)\}+u_2g_{n+1}(n_0+1)=0,$$
$$\text{for } n_0=1,\dots,n-1,\ n=2,\dots,N-1;$$

$$-\mu_1f_N(0)+\nu_1(N)f_N(1)=0,\quad -\mu_2g_N(0)+\nu_1(N)g_N(1)=0;$$

$$-[\nu_1(N)+\nu_2+\mu_1]f_N(n_0)+\alpha(N)\mu_1f_N(n_0-1)+\alpha(N)\mu_2g_N(n_0-1)+\nu_1(N)f_N(n_0+1)+\nu_2A_0(N)f_{N-1}(n_0-1)=0,$$

$$-[\nu_1(N)+\nu_2+\mu_2]g_N(n_0)+\beta(N)\mu_1f_N(n_0-1)+\beta(N)\mu_2g_N(n_0-1)+\nu_1(N)g_N(n_0+1)+\nu_2A_0(N)g_{N-1}(n_o-1)=0$$
$$\text{for } n_0=1,\dots,N-1;$$

$$\beta(n)f_n(n)=\alpha(n)g_n(n)\quad \text{for } n=1,\dots,N;$$

$$\sum_{n_0=0}^{n}[f_n(n_0)+g_n(n_0)]=1\quad \text{for } n=1,\dots,N.$$

Appendix 2

We now give formulas (40) in the case of "fixed multiprogramming" with $\mu_1=\mu_2=\mu$.

Our model becomes a central server network with two peripheral servers, SM and FD. We have

$$f_{n^0}^{n}=\alpha\rho^{n_0}/G(n)\quad\text{and}\quad g_{n^0}^{n}=\beta\rho^{n_0}/G(n),$$

where $\rho=\mu/\nu_1$ (recall that $\mu_1=\mu_2=\mu$, $\beta(N)=\beta$, $\nu_1(n)=\nu_1$) and $G(n)=\sum_{n_0=0}^{n}\rho^{n_0}$. We also have

$$A_0^n=\rho G(n-1)/G(n).$$

Hence

$$f_{n^0}^{n}-A_0^nf_{n-1}^{n_0-1}=\alpha\{\rho^{n_0}/G(n)-[\rho G(n-1)/G(n)]\rho^{n_0-1}/G(n-1)\}$$
$$=\alpha\{\rho^{n_0}/G(n)-\rho^{n_0}/G(n)\}=0,$$

and similarly for $g_{n^0}^{n}$.

Thus, though the quantities that appear in (40) pertain to System 2, they reflect the departure of *our model* (of Figure 1(a)) from a queueing network with a product form solution. This is because System 2 is a subsystem of our model and the form of the overall solution depends on the characteristics of the subsystem.

Appendix 3

Recurrence relations of the iterative procedure as are follows:

$$-(\mu_1 + u_2)\epsilon_n^k(0) + \nu_1(n)\epsilon_n^k(1) = Q_{1n}^k(0), \quad -(\mu_2 + u_2)\eta_n^k(0) + \nu_1(n)\eta_n^k(1) = Q_{2n}^k(0),$$
$$\text{for } n = 1, \ldots, N, \; k = 0, 1, \ldots;$$

where

$$Q_{1n}^k(0) = \begin{cases} Q_{1n}^0/A_0^{n+1}, & k = 0, \\ -u_2\{\epsilon_{n+1}^{k-1}(1) + f_n^0[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1}, & k > 0, \end{cases}$$

$$Q_{2n}^k(0) = \begin{cases} Q_{2n}^0/A_0^{n+1}, & k = 0, \\ -u_2\{\eta_{n+1}^{k-1}(1) + g_n^0[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1}, & k > 0; \end{cases}$$

$$-[\nu_1(n) + \mu_1 + \nu_2 + u_2]\epsilon_n^k(n_0) + \alpha(n)\mu_1\epsilon_n^k(n_0 - 1) + \alpha(n)\mu_2\eta_n^k(n_0 - 1) + \nu_1(n)\epsilon_n^k(n_0 + 1)$$
$$+ \nu_2 A_0^n\epsilon_{n-1}^k(n_0 - 1) - \nu_2 f_{n-1}^{n_0-1}[\epsilon_n^k(0) + \eta_n^k(0)] = Q_{1n}^k(n_0),$$

$$-[\nu_1(n) + \mu_2 + \nu_2 + u_2)\eta_n^k(n_0) + \beta(n)\mu_1\epsilon_n^k(n_0 - 1) + \beta(n)\mu_2\eta_n^k(n_0 - 1) + \nu_1(n)\eta_n^k(n_0 + 1)$$
$$+ \nu_2 A_0^n\eta_{n-1}^k(n_0 - 1) - \nu_2 g_{n-1}^{n_0-1}[\epsilon_n^k(0) + \eta_n^k(0)] = Q_{2n}^k(n_0),$$
$$\text{for } n_0 = 1, \ldots, n - 1, \; n = 2, \ldots, N - 1,$$

where

$$Q_{1n}^k(n_0) = \begin{cases} Q_{1n}^{n_0}/A_0^{n+1}, & k = 0, \\ \nu_2\{(A_0^n f_{n-1}^{n_0-1} - f_{n^0}^{n_0})[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1} - u_2\{\epsilon_{n+1}^{k-1}(n_0 + 1) \\ \quad + f_{n^0}^{n_0}[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1}, & k > 0, \end{cases}$$

$$Q_{2n}^k(n_0) = \begin{cases} Q_{2n}^{n_0}/A_0^{n+1}, & k = 0, \\ \nu_2\{(A_0^n g_{n-1}^{n_0-1} - g_{n^0}^{n_0})[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1} - u_2\{\eta_{n+1}^{k-1}(n_0 + 1) \\ \quad + g_{n^0}^{n_0}[\epsilon_{n+1}^{k-1}(0) + \eta_{n+1}^{k-1}(0)]\}/A_0^{n+1}, & k > 0; \end{cases}$$

$$-\mu_1\epsilon_N^k(0) + \nu_1(N)\epsilon_N^k(1) = 0 \quad \text{for all } k,$$
$$-\mu_2\eta_N^k(0) + \nu_1(N)\eta_N^k(1) = 0 \quad \text{for all } k;$$

$$-[\nu_1(N) + \mu_1 + \nu_2]\epsilon_N^k(n_0) + \alpha(N)\mu_1\epsilon_N^k(n_0 - 1) + \alpha(N)\mu_2\eta_N^k(n_0 - 1)$$
$$+ \nu_1(N)\epsilon_N^k(n_0 + 1) + \nu_2 A_0^N\epsilon_{N-1}^k(n_0 - 1) - \nu_2 f_{N-1}^{n_0-1}[\epsilon_N^k(0) + \eta_N^k(0)] = Q_{1N}^k(n_0),$$

$$-[\nu_1(N) + \mu_2 + \nu_2]\eta_N^k(n_0) + \beta(N)\mu_1\epsilon_N^k(n_0 - 1) + \beta(N)\mu_2\eta_N^k(n_0 - 1)$$
$$+ \nu_1(N)\eta_N^k(n_0 + 1) + \nu_2 A_0^N\eta_{N-1}^k(n_0 - 1) - \nu_2 g_{N-1}^{n_0-1}[\epsilon_N^k(0) + \eta_N^k(0)] = Q_{2N}^k(n_0)$$
$$\text{for } n_0 = 1, \ldots, N - 1,$$

where

$$Q_{1N}^k(n_0) = \begin{cases} Q_{1N}^{n_0}, & k = 0, \\ 0, & k > 0, \end{cases} \qquad Q_{2N}^k(n_0) = \begin{cases} Q_{2N}^{n_0}, & k = 0, \\ 0, & k > 0; \end{cases}$$

$$\beta(n)\epsilon_n^k(n) = \alpha(n)\eta_n^k(n), \quad n = 1, \ldots, \quad k = 0, 1, \ldots;$$

$$\sum_{n_0=0}^{n} [\epsilon_n^k(n_0) + \eta_n^k(n_0)] = 0, \quad n = 1, \ldots, \quad k = 0, 1, \ldots.$$

Appendix 4

The evaluated errors in the case $M = 128$ pages, $k = 1.5$, $t_{SM} = 5$ msec, $t_{FD} = 50$ msec, $r = 30$ msec, $N = 5$ processes with "floating multiprogramming" organization are as follows:

$n = 1, A_0^n = 0.743$,

n_0:	0	1
$f_{n^0}^n$:	.257	.743
$\epsilon_n(n_0)$.	.013	− 018
$g_{n^0}^n$:	.000	.000
$\eta_n(n_0)$·	006	000

$n = 2, A_0^n = 0.655$:

	0	1	2
	.298	.305	.307
	.008	−.009	−.002
	.047	.024	018
	.003	−.001	−.000

$n = 3, A_0^n = 0.466$

n_0:	0	1	2	3
$f_{n^0}^n$	.402	.224	117	063
$\epsilon_n(n_0)$	005	− 005	000	.004
$g_{n^0}^n$	131	.037	018	.008
$\eta_n(n_0)$.	− 003	−.001	−.001	.000

$n = 4, A_0^n = 0\ 308$.

	0	1	2	3	4
	.467	169	055	018	006
	005	− 002	.000	001	001
	225	.041	013	004	001
	−.006	000	−.000	− 000	000

$n = 5, A_0^n = 0.212$:

n_0:	0	1	2	3	4	5
$f_{n^0}^n$:	477	123	028	.006	.001	000
$\epsilon_n(n_0)$	− 003	− 001	001	.000	.000	000
$g_{n^0}^n$.	311	.040	.009	002	000	000
$\eta_n(n_0)$.	001	000	000	000	000	.000

A (the CPU utilization) is overestimated by 0.30 percent.

ACKNOWLEDGMENTS. I would like to thank Professor J. Buzen (Harvard University) for his valuable remarks. I would also like to thank the referees for their constructive criticism.

REFERENCES

(Note References [1, 17] are not cited in the text)

1. ANDO, A., AND FISHER, F.M. Near-decomposability, partition and aggregation, and the relevance of stability discussions *Int Econ Rev 4,* 1 (Jan. 1963), 53-67
2 AVI-ITZHAK, B , AND HEYMAN, D.P Approximate queueing models for multiprogramming computer systems. *Oper Res 21,* 6 (Nov -Dec. 1973), 1212-1230
3. BASKETT, F , ET AL Open, closed, and mixed networks of queues with different classes of customers. *J ACM 22,* 2 (April 1975), 248-260
4. BELADY, L , AND KUEHNER, C J Dynamic space-sharing in computer systems *Comm ACM 12,* 5 (May 1969), 282-288
5. BETOURNE, C , ET AL Process management and resource sharing in the multi-access system ESOPE *Comm. ACM 12,* 12 (Dec 1970), 727-733.
6 BRANDWAJN, A. A model of a time-sharing virtual memory system solved using equivalence and decomposition methods *Acta Informatica 4* (1974), 11-47
7. BRANDWAJN, A A model of a virtual memory system *Acta Informatica 6* (1976), 365-386
8 BRANDWAJN, A. An iterative solution of two-dimensional birth and death processes. Submitted to a technical journal.
9. BUZEN, J Queueing network models of multiprogramming. Ph D Th , Harvard U , Cambridge, Mass., 1971
10. CHANDY, K M , ET AL. Approximate analysis of general queueing networks IBM Res Rep. RC 4931, IBM Thomas J. Watson Res. Ctr., Yorktown Heights, N Y., July 1974
11. COURTOIS, P.J. Decomposability, instabilities, and saturation in multiprogramming systems *Comm. ACM 18,* 7 (July 1975), 371-376
12. GORDON, W.J., AND NEWELL, G F Closed queueing systems with exponential servers. *Oper. Res 15* (April 1967), 254-265.
13. JACKSON, J R. Jobshop-like queueing systems *Manage. Sci. 10* (1963), 131-142
14. JORDAN, C Calculus of finite differences Chelsea, New York, 1965.
15. MOORE, C.G Network models for large scale time-sharing systems Ph D Th., U of Michigan, Ann Arbor, Mich., 1971.
16. SCHERR, A.L An analysis of time-shared computer systems. M I T Press, Cambridge, Mass., 1967.
17. SIMON, H., AND ANDO, A. Aggregation of variables in dynamic systems *Econometrica 20,* 2 (April 1961), 111-138

RECEIVED DECEMBER 1974, REVISED APRIL 1976