Approximate Analysis of Central Server Models with Non-exponential Service Distributions, Different Classes of Customers and Priority Queueing Disciplines*

C.H. Sauer, K.M. Chandy
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712

TR-45

ABSTRACT

Service time distributions at computer processing units are often non-exponential. Empirical studies show that different programs may have markedly different processing time requirements. When queueing disciplines are first come first served, preemptive priority or non-preemptive priority models reflecting these characteristics are difficult to analyze exactly. Available approximate techniques are often too expensive for parametric analysis. Inexpensive approximate techniques for solution of central server models with the above characteristics are presented. The results of these techniques are validated with simulation results.

November 1974

*Research supported in part by National Science Foundation grants GJ-1084 and GJ 35109.

1. INTRODUCTION

Central server queueing network models have been widely used in the analysis of computing systems. (1, 3, 13, 16, 17) These models assume that a fixed number of customers (programs) traverse a closed network consisting of the central processor (CPU) and the input/output (I/O) devices. A customer alternately receives service from the CPU and one of the I/O devices. A customer may have to wait in a queue if the server is busy. After completing service at the CPU, a customer selects an I/O device according to probabilities associated with that device and the given customer. These probabilities are independent of the state of the system. The service time of a customer on a device may depend upon the device, the customer, and the queue lengths for that device, but is otherwise independent of the state of the system. Figure 1 illustrates a central server model with three I/O devices.

Often the models used are such that solutions for the equilibrium behavior can be determined using the techniques of local balance (2, 4). If the model is to have first come first served (FCFS) queueing disciplines, and if the techniques of local balance are to be used in the solution of the model, then it must be assumed that, at the servers with FCFS disciplines, the service distributions are exponential and independent of the customer being served. Local balance techniques do not allow priority queueing disciplines.

Empirical studies on real computing systems show that CPU service distributions are often hyper-exponential (the standard deviation is greater than the mean) and that I/O device service distributions may be

hypo-exponential (the standard deviation is less than the mean). Studies (11) have shown that mean service times and service distributions are dependent on the customer being served. When one makes assumptions that distributions are exponential and all customers have the same distributions, significant inaccuracy may be introduced into the model. Clearly, distinctions must be made between customers if priority CPU distributions are considered. Therefore, (a) many realistic problems do not satisfy local balance and (b) customer differentiation is often required for realistic models.

Chandy, Herzog and Woo (5) have developed accurate approximate iterative techniques for analysis of general queueing networks with non-exponential service distributions and distributions dependent on customer class. The iterative techniques of Wallace and Rosenberg (18) may also be used to obtain exact solutions for models with non-exponential distributions. The techniques of Crane and Iglehart (8, 9) may be used to obtain confidence intervals for simulation results for these models and thus to obtain accurate simulation results. However, these techniques are relatively expensive to apply. In many instances it will not be practical to survey a large variety of models using these techniques.

We present here approximate solution techniques specifically intended for, but not limited to, central server models of computing systems. Our techniques are considerably less expensive to apply than the above mentioned techniques, but are sufficiently accurate for the initial stages of computer system design. Our techniques complement the previous techniques in that ours can be used to study and compare a large variety of models, and then more accurate, more expensive techniques may be used, to study more carefully, a small subset of the original group of models.

Section 2 of the paper summarizes central server models in local balance and gives examples of inaccuracies of "local balance assumptions." Section 3 describes "Norton's Theorem" on locally balanced queueing networks (6) as applied to central server models. Our approximations are based on the results of Norton's Theorem. Section 4 presents the approximations for models with non-exponential distributions, section 5 presents techniques for class dependent service distributions, and section 6 presents techniques for models with priority CPU disciplines based on customer class. In section 7 we compare the results of our techniques with results of simulations; our techniques are validated by comparison with over 125 different simulations.

2. LOCAL BALANCE

A central server model will be in local balance (2) if 1) branching probabilities are dependent only on the device and the customer class, 2) all queueing disciplines are FCFS, processor sharing (PS) or last come first served preemptive resume (LCFSPR), 3) servers with FCFS discipline have exponential distributions independent of customer class, and 4) servers with PS or LCFSPR disciplines have differentiable service distributions (which may be dependent on customer class). In these models the equilibrium state probabilities will have the "product form," and are easily calculated (2). From the state probabilities one can determine model statistics such as throughput, server utilization, queue length distributions and waiting time distributions.

The following example illustrates the inaccuracy which may be introduced by using local balance solutions for models violating local balance assumptions. This example is by no means a worst case, but illustrates that results of assuming local balance are likely to be unsatisfactory.

Suppose that a system to be modeled has one I/O device and two classes

of customers, with one customer per class. Further, both service disciplines are FCFS, all service distributions are exponential, the mean CPU service time for class one is 2, the mean CPU service time for class two is .2, and the mean I/O service time for both classes is 1. Suppose we are interested in the overall throughput of customers through the CPU. This model is small enough that exact solution of the Markov balance equations is practical, and from the solution of these equations the throughput is .5941. If we assume that the results for a similar model with PS CPU discipline will be accurate enough, the value we get for throughput will be .84, an error of more than 40%. If we apply the techniques of section 5, the value we get for throughput is .6375, an error of about 7%.

Other examples illustrating the inaccuracy introduced by local balance assumptions are found in (5).

3. NORTON'S THEOREM APPLIED TO CENTRAL SERVER MODELS

This section reviews earlier work on Norton's Theorem in subsection 3.1 and presents two examples: in 3.2 a multiclass problem is presented, and a single-class example is worked out in 3.3.

3.1 Norton's Theorem: a discussion

Norton's Theorem (6) may be used to transform a central server model in local balance into one with a single "composite" I/O which represents the combined effects of the I/O devices in the original model at steady state. See fig. 2. Values determined for equilibrium throughputs, server utilizations, and CPU queue length and waiting time distributions of the two-queue model will be the same as those calculated for the original model. The transformation is independent of the CPU parameters, so if a variety of CPU parameters are to be studied, effort may be saved by applying Norton's Theorem and studying the reduced model as the CPU parameters are varied.

The approximation technique presented here is also especially well suited for parametric analysis of the CPU.

In describing Norton's theorem we shall assume without loss of generality that there are m classes of customers and exactly one customer of each class. The composite I/O processes all customers in parallel in the two queue, CPU-composite I/O model. The composite I/O service rate for the customer of any given class i at any given time depends upon i and upon the number of customers N_j of class j (N_j = 0 or 1), j=1,..,m, in the composite I/O queue at that time. These composite I/O service rates are determined by analyzing a modified version of the original network in which the CPU has been "shorted," i.e. the mean CPU service time for all customers is set to zero. See fig. 3. The composite I/O service rate of a customer of any given class i, when there are N_j customers of class j, (N_j = 0 or 1), j=1,...,m, in the composite I/O queue, is set equal to the throughput of the customer in class i through the shorted CPU when there is a population of N_j customers of class j, j=1,...,m in the shorted CPU model. The solutions of the two-queue, CPU-composite I/O model, with the same CPU parameters as in the original model and these queue-dependent composite I/O service rates, will be identical to those of the original model for the equilibrium statistics mentioned above.

3.2 Example

Consider the following two-class example of a locally-balanced central-server model with a processor-shared CPU and two I/Os labeled 1 and 2, two non-identical customers, one of class A and the other of class B. The class A customer uses I/O's 1 and 2 with equal probability, while the class B customer uses I/O 2 exclusively. The mean service time for each I/O is inde-

pendent of customer class. The mean service times for I/Os 1 and 2 are 1 and 2, respectively. All I/O service times have negative-exponential distributions.

Both class A and B customers are assumed to be serviced in parallel in the composite I/O queue. The service rates for class A and class B customers depend upon the numbers of class A and B customers in the composite I/O queue. We next discuss the computation of these rates by analyzing the modified version of the original network in which the CPU has been shorted (fig. 3). When only the class A customer is present in the CPU-shorted network, the throughput of the class A customer through the shorted CPU is 2/3; when only the class B customer is present the throughput is 1; and when both are present, the throughputs for classes A and B are 1/2 and 3/4 respectively. The composite I/O service rates when there is one customer of class A and none of class B in the composite I/O queue is set to 2/3 for class A (and 0 for class B); when there is one customer of class B and none of class A the rate is set to 1 for class B (and 0 for class A); and when there is one customer of each class the rates are set to 1/2 for class A and 3/4 for class B. The solution of the CPU-composite-I/O model with the same CPU parameters as in the original model and these queue-dependent composite I/O service rates, will be identical to the solutions of the original model for the equilibrium statistics mentioned above.

3.3 Example 2 Consider the following single customer class example. A locally balanced central server model has two identical customers and two I/O devices labeled 1,2. The probabilities that a customer will branch to the ith I/O device, $i = 1,2$ are .5 and .5 respectively. The mean service times for I/Os 1 and 2 are 4.0 and 2.0 respectively. All I/O queues have the FCFS discipline. With the CPU shorted and j customers in the shorted network, $j=1,2$, the throughputs through the shorted CPU are 1/3 and 3/7 respectively.

In the two-queue, CPU-composite-I/O model, the total service rates of the composite I/O when there are j customers in the composite I/O queue, j=1,2, are set to 1/3 and 3/7 respectively. Equivalently, with all customers served in parallel, the rate for each customer when there are j customers in the queue is 1/3 for j=1, and (1/2) x (3/7) = 3/14 for j=2.

The equilibrium statistics computed for the CPU-composite-I/O model with the same CPU parameters as in the original model, and these queue-length dependent composite I/O service rates, will be identical to the equilibrium statistics computed for the original model.

4. FCFS CENTRAL SERVER MODELS WITH NON-EXPONENTIAL SERVICE TIMES

We first discuss the overall technique generally (4.1), then study composite I/O representations (4.2), present the detailed algorithm (4.3), and finally work out an example (4.4).

4.1 Overview

We now restrict our attention to central server models with all customers identical, FCFS disciplines at all servers, and arbitrary service distributions having rational Laplace transforms. Even though this class of models is not in local balance except when all service distributions are exponential, we shall apply Norton's Theorem and show that the composite I/O model yields solutions close to those of the original model. (In making the composite I/O transformation we assume that the I/O devices have exponential distributions with the same means as the actual distributions. See example below.) Chandy, Herzog and Woo (5) use an approximate application of Norton's Theorem in their iterative method. In order to compensate for the inaccuracy introduced, we adjust the distributions for the composite I/O to reflect the non-exponential character of the actual distributions.

After applying Norton's Theorem and adjusting the distributions, we have a central server model with a single composite I/O, with both service

distributions non-exponential. This model is solved by an efficient recursive technique which is an application of the technique developed by Herzog, Woo and Chandy (10). Their technique assumes distributions of the generalized Erlang form developed by Cox (7). This generalized form includes arbitrary distributions with rational Laplace transform. Our technique assumes that both the CPU and the I/O distributions are of this general form. Details of our two queue analysis are given in (15).

Our adjustment for the non-exponential nature of the I/O distributions is simple and effective. More sophisticated adjustments could potentially increase the accuracy of the final results. We characterize each I/O distribution by its mean and coefficient of variation (standard deviation divided by the mean). For the means of the composite I/O distributions we use the queue length dependent values as shown earlier. We assume that the composite I/O coefficient of variation is the weighted sum of the coefficients of variation of the individual distributions, with the weights being the I/O branching probabilities. The composite I/O coefficient of variation is a constant, independent of queue length. Of course, the mean and coefficient of variation do not completely specify the distribution. If the composite I/O coefficient of variation is greater than one, we assume that the composite I/O service time is a standard two stage hyper-exponential as in fig. 4. If the coefficient of variation is one, we assume the service time is exponential. If the coefficient of variation is less than one, we assume the service time is of the generalized Erlang form with the minimum number of stages necessary to obtain the given coefficient of variation, all stages having the same mean, and all branching probabilities zero, with the possible exception of the branch after the first stage, as in fig. 5.

4.2 The Composite I/O Distribution

We desire the composite I/O distribution to represent the aggregate of all the individual I/O distributions. Intuitively, we expect the distribution of a given I/O to influence the composite I/O distribution more than distributions of other I/Os, if the given I/O processes more customers than other I/Os. We decided to restrict attention to the first two moments to keep computation simple. The means of composite I/O service times are obtained by aggregating individual I/O mean service times via Norton's Theorem. The composite coefficient of variation is obtained by aggregating individual coefficients of variation , weighting each I/O by its branching probability since I/O branching probabilities are directly proportional to I/O throughputs. Note that though the mean composite service time is queue-length dependent, the coefficient of variation is not dependent on queue length. Note also that if all the I/Os have the same coefficient of variation, then the composite I/O will have that coefficient of variation too.

The first two moments do not completely specify a distribution. We decided to model composite service times using either two-stage hyperexponential (fig. 4) or generalized Erlang (fig. 5) random variables since these are common ways of representing service times in computing systems. Note that the particular forms of the hyperexponential and generalized Erlang random variables are such that the first two moments uniquely specify the distributions. The selection of these particular composite I/O distributions were made with modeling convenience and reasonability in mind; clearly other choices could also have been made. However, note that if the original model satisfies local balance, then our technique gives <u>exact</u> results, since the

composite I/O distribution obtained via our technique is the same as that obtained via Norton's Theorem.

The hyperexponential

Let k_c be the coefficient of variation of the composite I/O. We shall use a standard hyperexponential random variable to model composite I/O service times if $k_c > 1$. The relationship between k_c and parameter p (fig. 4) of the hyperexponential is shown below:

$$p = \frac{k_c^2 + 1 - \sqrt{k_c^4 - 1}}{2(k_c^2 + 1)} \qquad (4.2.1)$$

Note that k_c uniquely specifies p. The means for each stage of this hyperexponential are uniquely specified by p and the mean composite service time.

$$\text{Mean of stage 1} = \frac{\text{mean composite service time}}{2p} \qquad (4.2.2)$$

$$\text{Mean of stage 2} = \frac{\text{mean composite service time}}{2(1-p)} \qquad (4.2.3)$$

Generalized Erlang

Consider the generalized Erlang (fig. 5) with n stages, n = 2,3,4,.... After a customer completes the first stage, he may finish service with probability p, or he may continue through the remaining n - 1 stages with probability 1-p. All stages have the same mean time, and all stage holding times are independent exponential random variables. By varying p from 0 to 1 the coefficient of variation ranges from $1/\sqrt{n}$ to 1. We wish to keep the number of stages small to minimize computation. Hence, we shall use n stages if and only if, $1/\sqrt{n-1} > k_c \geq 1/\sqrt{n}$; The value of n is directly determined from k_c. n and k_c together uniquely specify p. See eqn. (4.2.4) below. The means for each stage are uniquely specified by n, k_c, p and the

means of the composite I/O service times.

$$P = \frac{2nk_c^2 + n - 2 - \sqrt{n^2+4-4nk^2}}{2(k_c^2+1)(n-1)} \qquad (4.2.4)$$

$$\text{Mean of each stage} = \frac{\text{mean composite service time}}{n - p(n-1)} \qquad (4.2.5)$$

In conclusion, the generalized Erlang and hyperexponential random variables shown in figs. 4 and 5, are completely specified by the first two moments, and have a wide range of coefficients of variation. The parameters p are independent of composite I/O mean service times and the mean times for all stages in both distributions are directly proportional to the composite I/O mean service time; this simple relationship is an advantage in modeling queue-dependent service rates.

4.3 The Algorithm

We now present the algorithm after explaining some notation. Let there be R I/O queues indexed 1,...,r,...,R. We shall use the subscript r to denote the rth I/O in the original model and the subscript c to denote the composite I/O in the CPU-composite-I/O model. Let p_r be the probability that a customer branches to the rth I/O device after finishing CPU service. Let k denote the coefficient of variation: k_c for the composite I/O and k_r for the rth I/O device. We shall use the subscript 0 (zero) for the CPU. Let U_r be the utilization and t_r the throughput for the rth queue, r= 0,1,...,R. Let λ_r be the service rate for the rth I/O device. Let $\bar{q}$ and $\bar{w}$ be the mean CPU queue length and wait times and let σ_q and σ_w be the corresponding standard deviations.

ALGORITHM A

Step 1. Composite I/O service rates

Consider the given (non-locally-balanced) model. Construct the

shorted-CPU model in which all I/O service times are assumed to be independent exponential random variables and the CPU service time is set to zero. The shorted-CPU model satisfies local balance and can be analyzed easily. Determine queue-dependent composite I/O service rates by analyzing the shorted-CPU model.

Step 2. Composite I/O coefficient of variation

Compute $$k_c = \sum_{r=1}^{R} k_r \cdot p_r$$

Step 3. Determine exponential stage representations for composite I/O service times from k_c and composite I/O mean service times.

If $k_c > 1$ use standard hyper-exponential random variable. (fig. 4)

If $k_c = 1$ use exponential random variable

If $k_c < 1$ use generalized Erlang random variable. (fig. 5)

Step 4. Solve the two queue, CPU-composite I/O model.

The CPU parameters in this model are set to the same values as in the original model. The composite I/O parameters are completely and uniquely specified by step 3. The two-queue model is completely specified. Analyze this model to determine U_0, t_0, $\bar{q}$, σ_q, $\bar{w}$, and σ_w.

Step 5. I/O Utilizations

Compute $$t_r = t_0 \times p_r \qquad \text{for } r = 1, \ldots, R$$
$$U_r = t_r/\lambda_r \qquad \text{for } r = 1, \ldots, R$$

stop.

4.4 Example

Consider example 2 in section 3.3 except that I/O 1 has an exponential service time, I/O 2 has a generalized Erlangian service time with a coefficient

of variation of .414 and the CPU has a standard hyper-exponential service time with a coefficient of variation of 2 and a mean of 2. We shall now follow through the five steps of the algorithm.

Step 1. The composite I/O service rates (from section 3.3) when there are j customers in the composite I/O queue are 1/3 and 3/7 respectively.

Step 2. $k_c = (0.5 \times 1.0) + (0.5 \times 0.414) = 0.707$

Step 3. Since $k_c < 1$ the generalized Erlang representation is used. In this case n will be 2 and p will be zero. (The rate for each stage is clearly twice the composite I/O service rate.)

Step 4. We now have a two-queue model where the CPU service time is a two-stage hyper-exponential and the composite I/O service time is a two-stage Erlang. The balance equations for the resulting Markov states are solved to obtain $U_0 = .571$, $t_0 = .286$, $\bar{q}_0 = .837$, $\bar{w}_0 = 2.93$

Step 5. $t_1 = t_0 \times 0.5 = .143$, $t_2 = t_0 \times 0.5 = .143$

$U_1 = t_1/\lambda_1 = .571$, $U_2 = t_2/\lambda_2 = .286$

stop.

5. FCFS CENTRAL SERVER MODELS WITH CLASS DEPENDENT SERVICE RATES

This section is divided into three subsections. In 5.1 we discuss the technique generally, in 5.2, the algorithm is presented and an example is worked out in 5.3.

5.1 Discussion

In this section, we restrict ourselves to models with several classes of customers, FCFS, all service distributions exponential, all I/O service rates independent of customer class, and the CPU service rates dependent on

customer class (14). The assumption of class independent I/O service rates can be justified by observing that the largest portion of most I/O services is spent on primarily program independent operations such as acquiring channels, positioning disk arms, and waiting for device rotation. The techniques presented here have been extended to non-exponential CPU distributions and can also easily be extended to non-exponential I/O distributions. They are extended to priority disciplines in the next section, using the techniques of the previous section. Our techniques may also be extended to other, more general models.

When we consider such central server models, even the reduced model obtained by applying the Norton's Theorem approximation to the I/O subnetwork is difficult to analyze. As the number of classes and/or the numbers of customers per class attain even moderate values, e.g. 4, the analysis becomes too complex to be of practical value.

To reduce the complexity of analysis, we transform the more general original model to an approximately equivalent one with only two classes of customers: a designated class with only one customer and a composite class representing all of the other customers in the network. This further reduced model can be analyzed relatively easily, by applying the Norton's Theorem approximation. By designating each class in the original model and in turn analyzing the corresponding reduced model, we can obtain approximate values for the interesting statistics by each customer class in the original model.

In transforming the original model to the one with only two classes, the customer of the designated class is given the same I/O branching probabilities and CPU service distribution as in the original model. For each I/O device, the composite class branching probability is determined as

a weighted sum of the branching probabilities of the classes being "coalesced" from the original model. The weights used are the relative throughputs of the corresponding customers in a model identical to the original model, except that the CPU is processor-shared; this PS model satisfies local balance and is easily analyzed. The CPU service distribution for the composite class is chosen to be the standard two stage hyperexponential distribution with mean and second moment determined from weighted sums of the means and second moments of the CPU service distributions of the classes being coalesced from the original model.

After this transformation is applied, the Norton's Theorem approximation is applied. The resulting model, with the composite class and composite I/O queue is analyzed by techniques similar to those used in section 4.

5.2 Algorithms. In this subsection we describe two algorithms, the main program, algorithm B, is presented in 5.2.1., and a subprogram, algorithm C, which approximates an N-class problem by a two-class problem, is in 5.2.2.

5.2.1 Algorithm B

Assume that there are N classes of customers. For purposes of exposition, we assume (without loss of generality) that there is only one customer in each class.

Step 1. For each class i in turn, i = 1, ..., N, do steps 2-i through 5-i and thus compute the throughputs and utilizations for all queues for class i, and also the means and variances of CPU queue lengths and wait times for class i. The algorithm stops after all N classes have been considered.

Step 2-i Use algorithm C to approximate the given N-class problem by a two-class problem where the two classes are the designated class and a "coalesced class" which represents all customers except those in the designated class. We shall refer to the original central-server model as model A

and this two-class approximation as model B. Note that B and A have exactly the same central-server network structure; only the number of classes is changed. The parameters for the designated class are the same in A and B. CPU service time for the coalesced class is assumed to be hyperexponential in B. I/O service times are identical in A and B.

Step 3-i. Compute composite I/O service rates for the designated and coalesced classes of model B in the usual manner (i.e. by computing throughputs through the shorted CPU of model B and assuming all I/O service times are exponential).

Step 4-i. Consider the resulting two-queue, two-class network consisting of the CPU and I/O queues and the designated and coalesced classes; we shall refer to this network as model C. Solve Markov balance equations to determine steady-state probabilities of model C. Determine CPU throughput t_{0i}, utilization U_{0i}, mean and variance of CPU queue length and wait time for designated class i from the equilibrium state probabilities of model C. (Statistics for the coalesced class are not computed).

Step 5-i. Determine I/O throughputs t_{ri}, and utilizations U_{ri}, for each I/O r, r = 1, ..., R, for the designated class i. Let p_{ri} be the probability that a customer of class i branches to I/O r after CPU service. Then

$$t_{ri} = t_{0i} \times p_{ri} \qquad \text{for } r = 1, \ldots, R$$

and

$$U_{ri} = t_{ri}/\lambda_i \qquad \text{for } r = 1, \ldots, R$$

Statistics for the coalesced class are not computed.

Fig. 6 shows the relationships between models A, B and C.

5.2.2. Algorithm C for determining CPU service distributions and I/O branching probabilities for the coalesced class.

Step 1. Consider a network identical to the given network (model A) except that the CPU is processor-shared; we shall refer to this network as model D. Model D satisfies local balance and is easily analyzable. For the purposes of algorithm C only, we shall approximate the CPU throughputs of model A by those of model D. Compute t'_j, the CPU throughput of class j in model D, for j = 1, ..., N.

Step 2. Compute the conditional probability V_j that a random customer who finishes I/O service in model D is in class j given that he is not in designated class i.

$$V_j = t'_j \Big/ \sum_{h \neq i} t'_h \qquad \text{for } j \neq i$$

$$= 0 \qquad \text{for } j = i$$

Step 3. Compute the first two moments of the CPU service time for the coalesced class. Let $E[S^n]$ and $E[S_j{}^n]$ be the nth moment of the CPU service time for the coalesced class and class j respectively, j = 1, ..., N. Then:

$$E[S] = \sum_j E[S_j] \cdot V_j$$

$$E[S^2] = \sum_j E[S_j{}^2] \cdot V_j$$

Represent CPU service time for the coalesced class by a standard hyperexponential random variable (fig. 4) with the above first two moments.

Step 4. Approximate I/O branching probabilities for the composite class by

$$p_{rc} = \sum_j p_{rj} \cdot V_j$$

Stop.

5.3 Example

Consider a model with two I/Os and three classes of customers. The mean service times for I/O 1 and I/O 2 are both 2 time units. The branching probabilities for the first I/O are 1., 0, .5, for classes 1,2 and 3 respectively, and 0, 1., .5, for the second I/O. CPU mean times for classes 1,2,3 are 1,2,3 respectively. All service times are assumed to be independent, exponential random variables.

Algorithm B- Step 1. We shall carry out steps 2-i through 5-i, for i = 1.

We first call algorithm C to obtain the 2-class approximation.

Algorithm C- step 1. Analyzing model D we get

$$t_2' = .159 \qquad t_3' = .111$$

Algorithm C-step 2. $V_1 = 0$, $V_2 = .589$, $V_3 = .411$

Algorithm C-step 3. $E[S] = (2 \times .589) + (3 \times .411) = 2.41$

$$E[S^2] = (8 \times .589) + (18 \times .411) = 12.12$$

The hyperexponential representation for the CPU service time has parameter $p = 0.398$.

Algorithm C-step 4.

$$p_{1c} = (0 \times .589) + (.5 \times .411) = .206$$

$$p_{2c} = (1 \times .589) + (.5 \times .411) = .795$$

We now have a two-class problem the CPU service time for the coalesced class is hyper-exponential with mean 2.41 and I/O branching probabilities for device 1 is .206 and for device 2 is .795.

Algorithm B- Step 3.1. The composite I/O service rates for class 1 and the coalesced class for different queue conditions are shown below.

Number of class 1 customers	Number of coalesced class customers	Total rate for class 1	Total rate for coalesced class
1	0	.5	0
0	1	0	.5
0	2	0	.598
1	1	.415	.415
1	2	.386	.556

Algorithm B-Step 4-1. Model C is analyzed to obtain $t_{01} = .173$, $U_{01} = .173$, $\bar{q}_1 = .59$, $w_1 = 3.43$, $\sigma_{q_1} = .49$, $\sigma_{w_1} = 3.09$.

6. APPROXIMATIONS FOR MODELS WITH PRIORITY CPU DISCIPLINES

Now we consider central server models with the same characteristics as in the previous section, except that the CPU discipline will be a priority discipline with priority based on customer class. We will restrict consideration to preemptive and non-preemptive priority based on customer class, but these techniques are directly applicable to other priority disciplines.

Again, we do not try to apply Norton's Theorem approximation directly, but rather coalesce the classes of customers in the original model to simplify the analysis. The reduced model we consider has three classes of customers: a designated class, which we do not restrict to a single customer as in the FCFS model, and two composite classes, one of a higher priority than the designated class, and one of lower priority. The coalescing of classes into these three classes is similar to the technique used in the previous section. The coalescing is done separately for each of the two composite classes. The CPU distribution used for each of the composite

classes is an exponential distribution with mean taken as the weighted sum of the means of the classes being coalesced into that composite class. The weights are the relative throughputs of classes within the composite class. In other respects, the analysis is essentially the same as that already described.

7. VALIDATION, IMPLEMENTATION AND PERFORMANCE

We have constructed a simulator which employs the confidence interval techniques of Crane and Iglehart (8, 9). This simulator can be used with general queueing networks with a variety of disciplines, heterogeneous classes of customers, and generalized Erlang service distributions. The simulator determines confidence intervals during the simulation, and continues the simulation until satisfactory intervals are obtained. Details of the simulator are found in (15). This simulator has been used to determine results for the various models described below. Crane and Iglehart show how to obtain confidence intervals for results of simulations of Markov models with finite or countable state spaces. In general, the confidence intervals obtained are as follows: For utilization, the 90% intervals are at most .05 wide. For those cases where queue lengths and waiting times are obtained, the 90% intervals for the means are at most $\pm$ 6% of the point estimates, and the 80% intervals for the standard deviations are at most $\pm$ 16% of the point estimates. In many of the cases the intervals are considerably tighter. However, we were unable to obtain confidence intervals for the FCFS models with 6 classes of customers. For these models the state space is very large, and we were unable to select a state that the system would return to frequently; this is necessary to

apply the Crane and Iglehart techniques. We used predetermined simulation run lengths for the 6 class FCFS models, with the run lengths based on experience with 4 class FCFS models.

We have implemented our approximation techniques as a set of Fortran programs for a CDC 6600. Over 125 models have been validated to assure a thorough sampling of problems.

~~48~~ 56 of the models validated are of the class described in section 4, i.e. single class, non-exponential. These models included from 2 to 12 customers, from 1 to 2 CPU's from 3 to 8 I/O devices, and a wide variety of combinations of distributions, with coefficients of variation ranging from .577 to 5. In general the models were fairly well balanced, but some of the models were strongly CPU bound or I/O bound. Error tolerances were determined in the manner used in (5) for utilizations, CPU queue lengths and CPU waiting times. Results are said to be within a tolerance z if 1) the difference in utilization is not more than z, 2) the differences in the means and standard deviations of queue length are not more than z times the number of customers in the network, and 3) the differences in the means and standard deviations of the wait times are not more than z times the cycle time. For the 48 models studied, the results are generally within a tolerance of .05, with a maximum tolerance of ~~.11~~ .18. In (5) a tolerance of .05 is considered to be good, and a tolerance of .10 is considered adequate. By these standards the results are good or adequate for ~~47~~ 51 of the ~~48~~ 56 models. For these models, the computer time required per model was negligible, approximately 75 milliseconds per model. Tables 1, 2, and 3 show results for 12 of these models.

44 models of the class described in section 5, i.e. FCFS with different classes of customers, including 4 with hyper-exponential CPU distribu-

tions, have been validated. These models include from 2 to 8 customers, with from 2 to 6 classes of customers, and 3 or 4 I/O devices. Utilizations and throughputs, both overall and by class, were validated for all of these models. For 8 of the models, queue lengths and wait times for each class were also validated. We did not explicitly determine tolerances as in the single class models, but in general the results showed good accuracy for utilization and reasonable accuracy overall. Tables 4 and 5 give results for 2 of the models. For the 44 models, the programs required approximately 400 milliseconds computation per model.

36 priority models were validated, 24 preemptive and 12 non-preemptive. These models included from 4 to 6 customers, with from 3 to 6 classes, and 3 or 4 I/O devices. Again, utilizations and throughputs were validated for all models. CPU queue lengths and mean CPU wait times were validated for 12 preemptive models and all non-preemptive models. Tables 6 and 7 show results for 2 models. For the 36 models, the computation per model was approximately 400 milliseconds.

In addition to providing reasonable accuracy for models not in local balance, these programs give exact results for models in local balance where class coalescing is not necessary. Though the coalescing techniques do not necessarily give exact results for locally balanced models, the results are very close. In the above validation process, for all FCFS models requiring coalescing, the coalescing process was applied to a locally balanced model similar to the non-locally balanced model being studied. Individual class throughputs and utilizations were compared for the locally balanced model with and without coalescing. The differences were never more than 1% and usually less than that.

These programs are more than an order of magnitude faster than existing implementations of other techniques.

8. CONCLUSIONS

We have presented approximate solution techniques for several classes of models which are very important in the modeling of computing systems. These techniques are computationally very inexpensive and of great practical value. They complement previous techniques which may be more accurate but are more computationally expensive, and may be used directly in conjunction with previous techniques.

Our techniques give exact results for several interesting classes of models, and are reasonably accurate for typical models of computing systems. Our techniques have been validated extensively. Our methods may be extended to consider more general networks. Our techniques are compatible with the techniques of Keller and Chandy (12) for including the effects of passive resources in central server models. Williams and Bhandiwad have also used approximations of Norton's theorem in analyzing three class preemptive, exponential models (19).

Three sets of programs were used in constructing and validating these models: (1) programs to approximate central server models by two-queue networks, (2) programs to analyze the resulting two-queue networks and (3) the Crane-Iglehart simulator. Four variations of the two-queue analysis technique were programmed: 1) single class non-exponential, 2) multiple class FCFS, 3) multiple class preemptive priority and 4) multiple class non-preemptive priority. Each of these cases (except the last two) required slightly different programs to construct two-queue approximations of the given central server problem. The simulator handles arbitrarily interconnected networks, a large number of customer classes and customers, and a variety of disciplines.

ACKNOWLEDGEMENT

The problem was first suggested in discussions with U. Herzog and L. Woo of IBM Research. We are grateful for their continued encouragement and criticism.

REFERENCES

(1) Baskett, F. Mathematical Models of Multiprogrammed Computer Systems. TSN-17, Computation Center, The University of Texas at Austin. (January 1971)

(2) Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios-Gomez, F. Open, Closed and Mixed Networks of Queues with Different Classes of Customers. To appear JACM. (January 1975)

(3) Buzen, J. Queueing Network Models of Multiprogramming. Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University. (June, 1971)

(4) Chandy, K.M. The Analysis and Solutions for General Queueing Networks. Proc. Sixth Annual Princeton Conference on Information Sciences, Princeton University. (March 1972)

(5) Chandy, K.M., Herzog, U., and Woo, L. Approximate Analysis of General Queueing Networks. To appear IBM Journal of Research and Development. (January 1975)

(6) Chandy, K.M., Herzog, U., and Woo, L. Parametric Analysis of Queueing Network Models. To appear, IBM Journal of Research and Development. (January 1975)

(7) Cox, D.R. A Use of Complex Probabilities in the Theory of Stochastic Processes. Proc. Cambridge Philosophical Society, 51, (1955), p. 313-319.

(8) Crane, M.A. and Iglehart, D.I. Simulation of Stable Stochastic Systems I: General Multiserver Queues. JACM 21, 1 (January 1974) p. 103-113.

(9) Crane, M.A. and Iglehart, D.I. Simulation of Stable Stochastic Systems II: Markov Chains, JACM 21, 1 (January 1974), p. 114-123.

(10) Herzog, U., Woo, L., and Chandy, K.M. Solution of Queueing Problems by a Recursive Technique. To appear, IBM Research Report, Yorktown Heights, New York.

(11) Johnson, D.S. A Process-Oriented Model of Resource Demands in Large, Multiprocessing Computer Utilities. TSN-29, Computation Center, The University of Texas at Austin, (August 1972).

(12) Keller, T.W. and Chandy, K.M. Computer Models with Constrained Parallel Processors. 1974 Sagamore Conference on Parallel Processing.

(13) Lee, C.C. Queueing Models of Device Utilization in Multiprogrammed Computer Systems, TR-7, Department of Computer Sciences, The University of Texas at Austin, (December 1972)

(14) Sauer, C.H. and Chandy, K.M. Approximate Analysis of Computer System Models with Different Classes of Customers. Presented at ORSA/TIMS Puerto Rico meeting. (October 1974)

(15) Sauer, C.H. Design of Computing Systems Using Approximate Solution of Queueing Network Models. Ph.D. Thesis, University of Texas at Austin.

(16) Shedler, G.S. A Cyclic Queue Model of A Paging Machine. IBM Research Report, RC 2814, Yorktown Heights, New York. (March 1970).

(17) Smith, J.L. An Analysis of Time Sharing Computer Systems Using Markov Models. SJCC, (1966).

(18) Wallace, V.L., Rosenberg, Richard S. Markovian Models and Numerical Analysis of Computer System Behavior. SJCC, (1966).

(19) Williams, A.C. and Bhandiwad, R. Private communication.

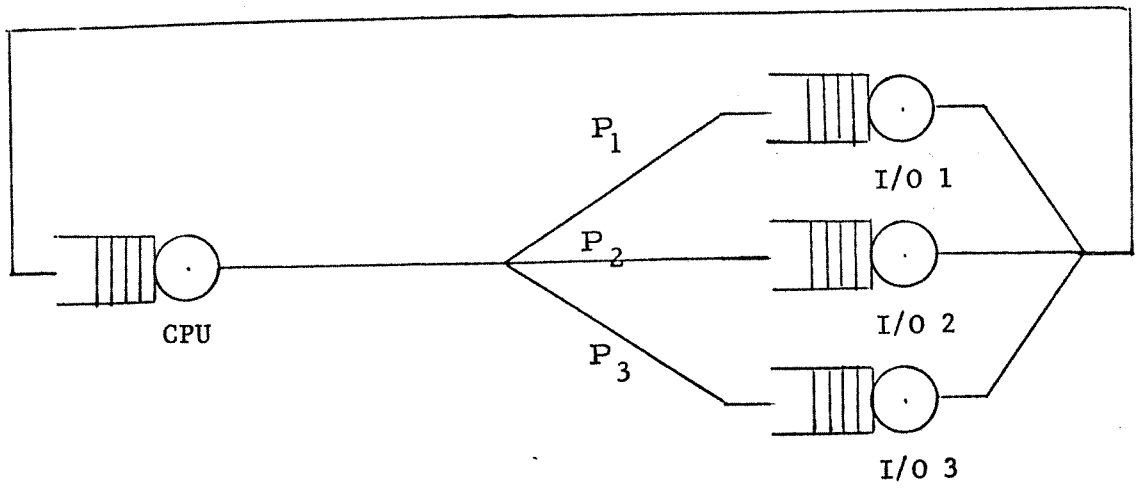

Figure 1 - Central Server Model

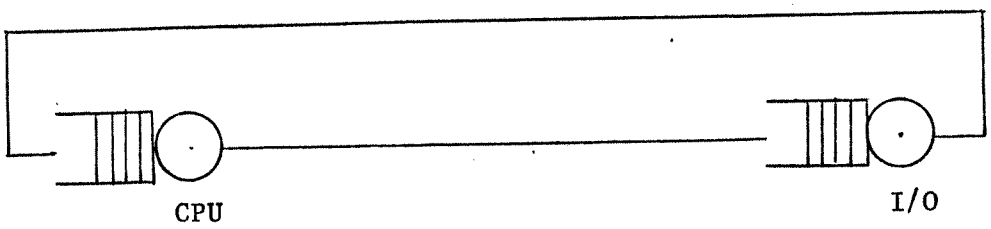

Figure 2 - CPU Queue and Composite I/O Queue

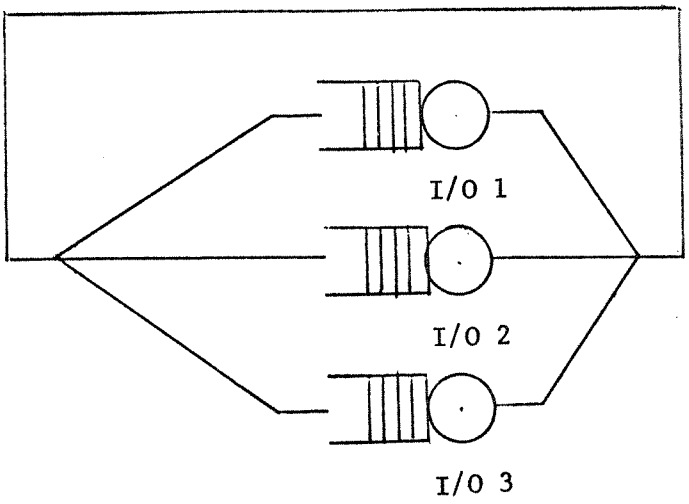

Figure 3 - Central Server Model with Shorted CPU

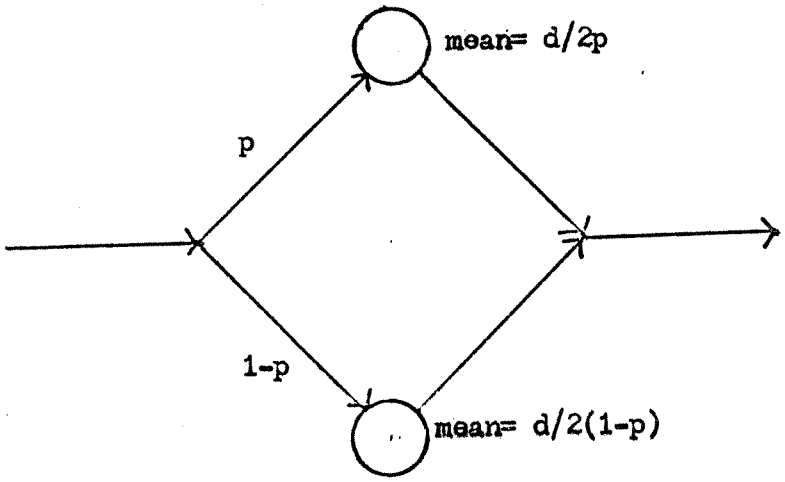

Figure 4 - Hyperexponential distribution form used for coefficient of variation ≥ 1. Mean = d.
$0 < p < 1$

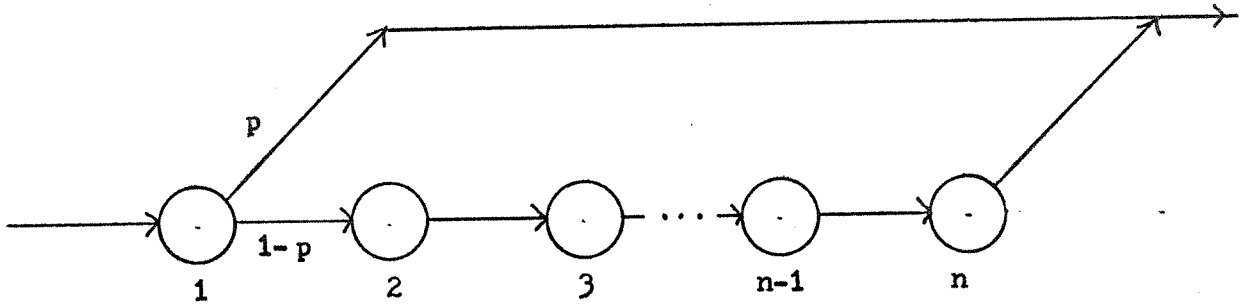

Figure 5 - n stage ($n > 1$) generalized Erlang distribution form used for $1/\sqrt{n-1}$ > coefficient of variation $\geq 1/\sqrt{n}$.
Mean of each stage = m. Distribution mean = $p/m + (1-p)/nm$
$0 \leq p < 1$

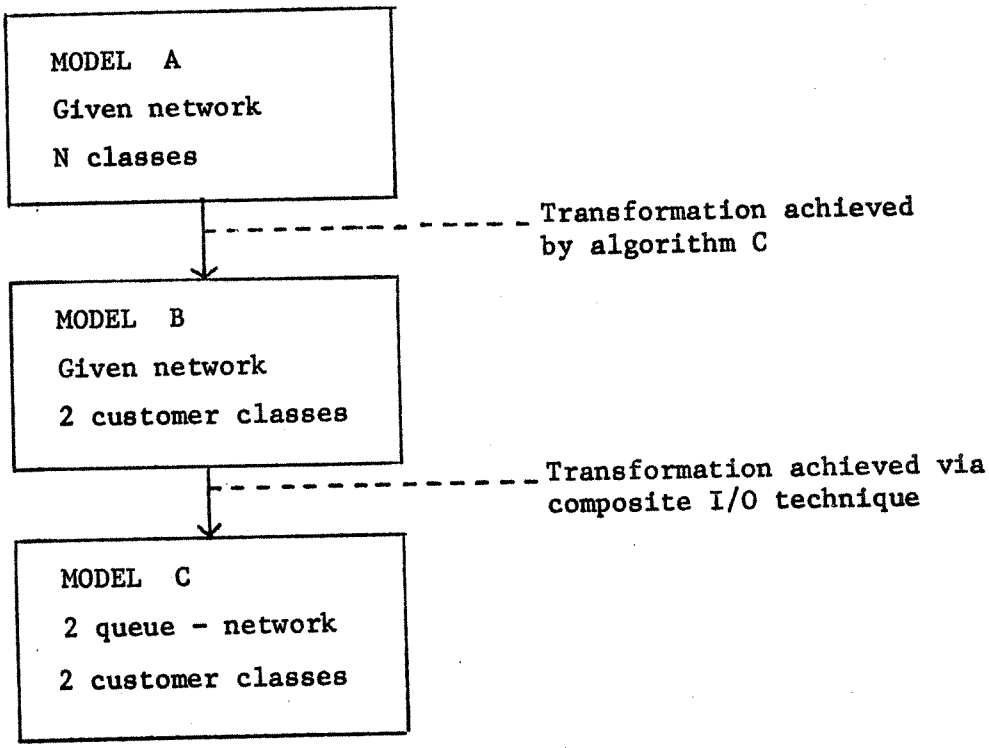
MODEL A
Given network
N classes
Transformation achieved by algorithm C
MODEL B
Given network
2 customer classes
Transformation achieved via composite I/O technique
MODEL C
2 queue - network
2 customer classes

FIGURE 6

Table 1

Hyper-exponential CPU, Exponential I/O

Number of Customers	2	4	8	12
CPU Utilization	.590	.713	.802	.846
(simulation)	.588	.715	.835	.864
(local balance)	.623	.783	.884	.921
CPU Mean Queue Length	.912	1.85	3.66	5.51
	.894	1.82	3.79	5.65
CPU Standard Deviation of Queue Length	.851	1.56	2.93	4.26
	.840	1.54	2.80	4.10
CPU Mean Wait Time	1.54	2.59	4.56	6.52
	1.50	2.57	4.55	6.49
CPU Standard Deviation of Wait Time	2.90	4.02	5.71	7.14
	2.73	4.07	5.53	7.24
IO-1 Utilization	.590	.713	.802	.846
	.606	.730	.824	.864
IO-2 Utilization	.148	.178	.201	.212
	.147	.176	.214	.212
IO-3 Utilization	.036	.045	.050	.053
	.036	.043	.054	.056
CPU Mean Service	1.			
Coef. of Variation	2.134			
I/O 1 Mean Service	2.			
Coef. of Variation	1.			
Branching Probability	.5			
I/O 2 Mean Service	1.			
Coef. of Variation	1.			
Branching Probability	.25			
I/O 3 Mean Service	.25			
Coef. of Variation	1.			
Branching Probability	.25			

Table 2

Exponential CPU, Erlang I/O

Number of Customers	2	4	8	12
CPU Utilization	.646	.813	.906	.937
(simulation)	.652	.821	.914	.940
(local balance)	.623	.783	.884	.921
CPU Mean Queue Length	.881	1.83	3.79	5.77
	.917	1.90	3.88	5.99
CPU Standard Deviation of Queue Length	.759	1.28	2.39	3.52
	.778	1.31	2.37	3.62
CPU Mean Wait Time	1.36	2.25	4.18	6.16
	1.43	2.34	4.23	6.42
CPU Standard Deviation of Wait Time	1.26	1.83	2.97	4.12
	1.30	1.86	2.92	4.29
IO-1 Utilization	.646	.813	.906	.937
	.629	.801	.904	.921
IO-2 Utilization	.162	.203	.227	.234
	.165	.213	.228	.232
IO-3 Utilization	.040	.051	.057	.059
	.040	.051	.057	.060

CPU Mean Service	1.
Coef. of Variation	1.
I/O 1 Mean Service	2.
Coef. of Variation	.707
Branching Probability	.5
I/O 2 Mean Service	1.
Coef. of Variation	.707
Branching Probability	.25
I/O 3 Mean Service	.25
Coef. of Variation	.707
Branching Probability	.25

Table 3

Hyper-exponential CPU, Erlang I/O

Number of Customers	2	4	8	12
CPU Utilization	.604	.730	.815	.857
(simulation)	.604	.751	.851	.893
(local balance)	.623	.783	.884	.921
CPU Mean Queue Length	.904	1.81	3.58	5.42
	.912	1.88	3.75	5.92
CPU Standard Deviation of Queue Length	.828	1.52	2.89	4.21
	.834	1.50	2.73	3.99
CPU Mean Wait Time	1.50	2.48	4.39	6.33
	1.51	2.57	4.44	6.68
CPU Standard Deviation of Wait Time	2.89	4.00	5.67	7.10
	2.86	4.15	5.31	6.89
IO-1 Utilization	.604	.730	.815	.857
	.610	.732	.845	.874
IO-2 Utilization	.151	.183	.204	.214
	.150	.187	.206	.232
IO-3 Utilization	.038	.046	.051	.054
	.038	.045	.053	.056

CPU Mean Service	1.
Coef. of Variation	2.134
I/O 1 Mean Service	2.
Coef. of Variation	.707
Branching Probability	.5
I/O 2 Mean Service	1.
Coef. of Variation	.707
Branching Probability	.25
I/O 3 Mean Service	.25
Coef. of Variation	.707
Branching Probability	.25

Table 4

FCFS, 4 Classes, 1 Customer/class, 3 I/O devices

Class	1	2	3	4	all
CPU Throughput	.182	.192	.110	.168	.652
(simulation)	.144	.156	.122	.126	.548
(local balance)	.230	.272	.090	.208	.800
CPU Utilization	.091	.048	.552	.084	.775
	.072	.039	.612	.063	.786
	.115	.068	.451	.104	.738
CPU Mean Queue Length	.36	.35	.59	.35	
(simulation)	.51	.49	.65	.48	
CPU Standard Deviation	.48	.48	.49	.48	
of Queue Length	.50	.50	.48	.50	
CPU Mean Wait Time	1.97	1.82	5.32	2.08	
	3.45	3.22	5.44	3.67	
CPU Standard Deviation	3.95	3.87	4.94	4.08	
of Wait Time	4.63	4.58	5.13	4.70	
I/O 1 Utilization	.219	.154	.044	.067	
	.176	.121	.047	.053	
I/O 2 Utilization	.058	.123	.071	.054	
	.047	.095	.077	.044	
I/O 3 Utilization	.097	.103	.118	.268	
	.080	.081	.128	.214	
CPU Mean Service	.500	.250	5.00	.500	
Coef. of Variation	1.00	1.00	1.00	1.00	
I/O 1 Probability (Mean Serv. = 2.00)	.6	.4	.2	.2	
I/O 2 Probability (Mean Serv. = 1.60)	.2	.4	.4	.2	
I/O 3 Probability (Mean Serv. = 2.67)	.2	.2	.4	.6	

Table 5

FCFS, 4 Classes, 1 Customer/class, 3 I/O devices

Class	1	2	3	4	all
CPU Throughput	.111	.182	.162	.152	.607
(simulation)	.118	.142	.130	.128	.518
(local balance)	.093	.239	.216	.230	.778
CPU Utilization	.555	.091	.081	.038	.765
	.590	.071	.065	.032	.758
	.465	.119	.108	.058	.750
CPU Mean Queue Length	.61	.39	.37	.36	
(simulation)	.64	.52	.50	.48	
CPU Standard Deviation	.49	.49	.48	.48	
of Queue Length	.48	.50	.50	.50	
CPU Mean Wait Time	5.51	2.13	2.30	2.37	
	5.28	3.69	3.75	3.68	
CPU Standard Deviation	5.08	6.90	7.25	7.56	
of Wait Time	9.48	8.67	8.83	8.89	
I/O 1 Utilization	.133	.145	.064	.061	
	.145	.115	.053	.050	
I/O 2 Utilization	.036	.116	.103	.049	
	.040	.089	.086	.041	
I/O 3 Utilization	.059	.097	.172	.245	
	.062	.077	.149	.209	
CPU Mean Service	5.00	.500	.500	.250	
Coef. of Variation	2.00	2.00	2.00	2.00	
I/O 1 Probability (Mean Serv. = 2.00)	.6	.4	.2	.2	
I/O 2 Probability (Mean Serv. = 1.60)	.2	.4	.4	.2	
I/O 3 Probability (Mean Serv. = 2.67)	.2	.2	.4	.6	

Table 6

Preemptive, 6 Classes, 1 Customer/class, 3 I/O devices

Class	1	2	3	4	5	6	all
CPU Throughput	.119	.166	.149	.138	.143	.138	.873
(simulation)	.148	.162	.141	.126	.114	.102	.793
(local balance)	.104	.195	.169	.161	.172	.172	.973
CPU Utilization	.396	.055	.050	.023	.048	.046	.618
	.493	.054	.047	.021	.038	.034	.687
	.347	.065	.056	.027	.057	.057	.609
CPU Mean Queue Length	.40	.29	.25	.21	.25	.27	
(simulation)	.49	.34	.34	.31	.37	.38	
CPU Standard Deviation	.49	.45	.44	.41	.44	.45	
of Queue Length	.50	.47	.47	.46	.48	.49	
CPU Mean Wait Time	3.33	1.76	1.70	1.53	1.78	1.99	
	3.39	2.06	2.44	2.54	3.16	3.59	
I/O 1 Utilization	.143	.133	.060	.055	.095	.092	
	.169	.129	.056	.047	.077	.067	
I/O 2 Utilization	.038	.106	.096	.044	.076	.073	
	.047	.102	.087	.037	.062	.062	
I/O 3 Utilization	.063	.088	.159	.220	.127	.122	
	.076	.093	.146	.202	.101	.093	
CPU Mean Service	3.33	.333	.333	.167	.333	.333	
I/O 1 Probability (Mean Serv. = 2.00)	.6	.4	.2	.2	.333	.333	
I/O 2 Probability (Mean Serv. = 1.60)	.2	.4	.4	.2	.333	.333	
I/O 3 Probability (Mean Serv. = 2.67)	.2	.2	.4	.6	.333	.333	

Table 7

Non-preemptive, 6 Classes, 1 Customer/class, 4 I/O devices

Class	1	2	3	4	5	6	all
CPU Throughput	.211	.354	.435	.384	.327	.326	2.036
(simulation)	.239	.381	.408	.348	.288	.270	1.934
(local balance)	.184	.334	.529	.438	.368	.373	2.226
CPU Utilization	.211	.236	.036	.032	.027	.022	.564
	.239	.254	.034	.029	.024	.018	.598
	.184	.223	.044	.037	.031	.025	.544
CPU Mean Queue Length	.23	.33	.25	.19	.16	.16	
(simulation)	.29	.36	.27	.26	.24	.25	
CPU Standard Deviation	.42	.47	.43	.39	.37	.36	
of Queue Length	.46	.48	.44	.44	.43	.43	
CPU Mean Wait Time	1.08	.963	.508	.478	.487	.480	
	1.24	.946	.641	.737	.857	.897	
I/O 1 Utilization	.263	.071	.087	.096	.163	.163	
	.305	.067	.083	.081	.146	.128	
I/O 2 Utilization	.026	.106	.174	.048	.082	.081	
	.030	.112	.177	.046	.070	.074	
I/O 3 Utilization	.013	.053	.087	.024	.041	.041	
	.015	.057	.082	.024	.038	.035	
I/O 4 Utilization	.026	.106	.043	.240	.082	.081	
	.028	.114	.041	.216	.067	.070	
CPU Mean Service	1.00	.667	.083	.083	.083	.067	
I/O 1 Probability (Mean Serv. = 2.00)	.625	.100	.100	.125	.250	.250	
I/O 2 Probability (Mean Serv. = 1.00)	.125	.300	.400	.125	.250	.250	
I/O 3 Probability (Mean Serv. = .500)	.125	.300	.400	.125	.250	.250	
I/O 4 Probability (Mean Serv. = 1.00)	.125	.300	.100	.625	.250	.250	