Efficient Approximation for Models of Multiprogramming with Shared Domains

Alexandre Brandwajn & William M. McCormack
Amdahl Corporation

Abstract:

Queueing network models of multiprogramming systems with memory constraints and multiple classes of jobs are important in representing large commercial computer systems. Typically, an exact analytical solution of such models is unavailable, and, given the size of their state space, the solution of models of this type is approached through simulation and/or approximation techniques. Recently, a computationally efficient iterative technique has been proposed by Brandwajn, Lazowska and Zahorjan for models of systems in which each job is subject to a separate memory constraint, i.e., has its own memory domain. In some important applications, it is not unusual, however, to have several jobs of different classes share a single memory "domain" (e.g., IBM's Information Management System). We present a simple approximate solution to the shared domain problem. The approach is inspired by the recently proposed technique which is complemented by a few approximations to preserve the conceptual simplicity and computational efficiency of this technique. The accuracy of the results is generally in fair agreement with simulation.

1. Introduction

Queueing network models of multiprogramming systems with multiple classes of jobs and memory constraints are important in representing large commercial computer systems, like the well known IBM's MVS [1]. Such models typically do not possess a known exact analytical solution, and the direct applicability of standard numeric techniques to the state equations is hindered by the fast growth of the number of states as the number of job classes and jobs per class increases. Thus, the solution of models of this type is approached through simulation or approximation techniques, or a mixture of both (e.g., [2,3,4,5]). In this paper, we take the route of approximate analysis that can handle relatively efficiently the inherent complexity of the model.

By applying state aggregation (or as some call it, equivalence), it is possible to reduce the models under consideration to queueing networks with a single job class and state-dependent mutually coupled queues, each queue representing a job class of the original model. Because of the coupling between queues, the solution of this reduced network is still a difficult task (cf. [1]), and the size of the network state space, though considerably scaled down, still increases rapidly with the number of classes and jobs per class. An approximate solution of such a reduced model can be attempted through repeated use of equivalence (state aggregation) and decomposition (cf. [6,5]), but this approach does not escape from the computational complexity since it enumerates the states of the reduced network. Recently, Brandwajn [7] and Lazowska and Zahorjan [8] proposed an iterative technique for solving the reduced network. The basic idea of their approach is to de-couple the queues in the network by assuming that the influence of other job classes on any given class of jobs can be adequately represented through the use of the average numbers of jobs of the other classes (as opposed to instantaneous numbers of users) in the state-dependent service rates of the coupled queues.

This technique is computationally efficient for systems in which each job class is subject to a separate memory constraint, i.e., has its own memory "domain." In some applications, it is not unusual, however, to have several jobs of different classes share a single domain. A typical example is the IBM Information Management System [9] where a set of different transactions may queue for a common set of multiprogramming slots called "message regions." In presence of such "shared domains," the use of average numbers of jobs to represent the influence of other job classes is not, by itself, sufficient to achieve complete decoupling of queues. The extension proposed in [8] to deal with shared domains defeats the computational simplicity of the

iterative technique, while [7] disregards shared domains altogether.

It is our goal in this note to present a simple approximate solution to the shared domain problem. The approach is inspired by the Brandwajn, Lazowska and Zahorjan technique which is complemented by a few approximations to preserve the conceptual simplicity and computational efficiency of this technique.

Led by what appear to be the most common applications of shared domains, we concentrate our attention on transaction processing systems. We allow the transaction arrivals to be represented as generated by a finite number of terminals for some classes and an infinite source for others.

In the next section, we describe in more detail the queueing network model under consideration and we present the approach proposed. Section 3 is devoted to numerical examples illustrating the accuracy of the approximate solution.

2. Model and Approximate Solution

Consider the system depicted in Figure 1. There are two types of transactions (jobs) with respect to arrivals to the system: transactions generated by a finite set of terminals, referred to by the acronym TS, and those originating at an infinite source, called TP transactions. We assume that there are f classes of TS transactions, each characterized by the number of terminals, N_i , the average think time, $1/\lambda_i$, and a set of values for system resource requirements. Similarly, there are o classes of TP transactions, each characterized by the source rate Λ_i , and a set of demands for system resources. Within each class, transactions are assumed to be statistically identical and independent.

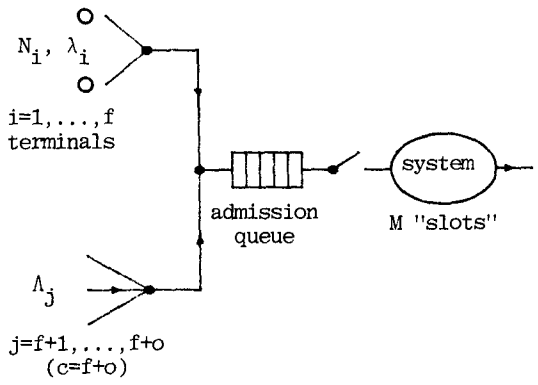


Figure 1: View of system considered

The system possesses a limited number (M) of "slots" for the admission of jobs of all classes. When no free slots are available, arriving transactions wait for entry into this shared domain in a single admission queue which we assume to be of a First-In-First-Out discipline.

TS type transactions are assumed to follow a cycle comprising a think time at the terminal and an interaction time with the system. For simplicity, we assume that the think times of TS job classes are exponentially distributed random variables, that TP transaction arrivals from an infinite source constitute a Poisson process, and that the demands placed upon system resources by a job class can be represented as service times following Coxian [10] probability distributions.

With these assumptions, the system under consideration can be represented by the aggregated queueing network shown in Figure 2. For each class n of jobs, n = 1,...,c, the execution in the shared memory domain is represented as passage through a private exponential queue whose instantaneous service rate depends on the current numbers of transactions of each class in the domain. These service rates, denoted by $v_n(m_1,...,m_n,...,m_c)$, can usually be assessed using a decomposition technique (cf. [5, 6]). Here we assume that they are known, and our goal is to develop an efficient solution for the queueing network of Figure 2.

Note that, even though all the servers in this queueing system are Markovian, the network, in general, does not possess a product-form solution [11]. The two reasons for this are the coupling between the queues in the form of state dependent service rates $v_n(m_1,...,m_n,...,m_c)$, and the admission control based on the total number of users in the system.

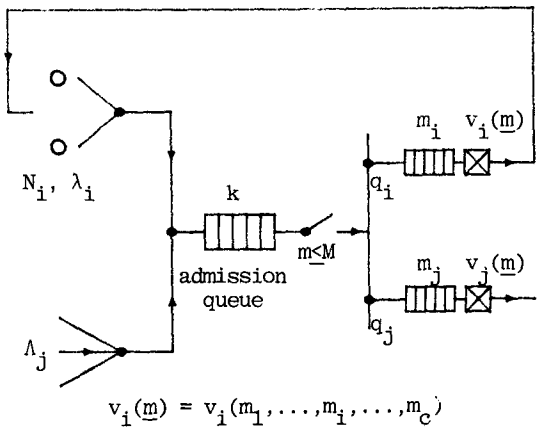


Figure 2: A queueing network model of system considered

As a first step in our approximate solution, we assume, following [7, 8], that, for any given job class, the influence of other classes is adequately represented by their average multiprogramming degrees, i.e., numbers of transactions in the domain. Thus, $\bar{m}_i$ being the average number of class i transactions in the domain, the service rate for class n becomes $v_n(\bar{m}_1,\ldots,m_n,\ldots,\bar{m}_c)$. Note that this rate is a function of m_n only for any fixed set of network parameter.

Denote by m the total current number of transactions in the domain

$$m = \sum_{n=1}^{c} m_n \ .$$

Denote also by k the current length of the admission queue. In our second step, we choose to look at the system from the standpoint of these two state variables. It is not difficult to show that the queueing system of Figure 2 can be reduced to the simple exponential queue of Figure 3 with a single class of jobs. The total arrival rate of transactions when the state of the queue is (k,m) is given by

$$\lambda(k,m) = \sum_{i=1}^{f} [N_i - \bar{k}_i(k) - \bar{m}_i(k,m)]\lambda_i + \sum_{j=f+1}^{f+o} \Lambda_j \ , \qquad (1)$$

where

$\bar{k}_i(k)$ is the average number of class i jobs in the admission queue given that its length is k;

$\bar{m}_i(k,m)$ is the average number of class i jobs in the domain given that there are a total of m jobs in the domain and k transactions in the admission queue.

The instantaneous service rate of the exponential domain server when the state of the queue is (k,m) is denoted by u(k,m). It is simply the conditional completion rate for transactions in the domain given the number in admission queue and current number of jobs in the domain.

We make the additional simplifying assumption that both $\bar{m}_i(k,m)$ and u (k,m) are functions of m only. (Since k = 0 for all $m < M$, this assumption actually pertains only to values for $m \geq M$). As a result, the arrival rate to the queue and its service rate become

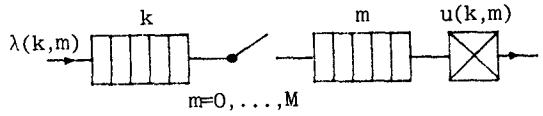


Figure 3: A reduced (equivalent) queueing model

$$\lambda(k,m) \simeq \sum_{i=1}^{f} [N_i - \bar{k}_i(k) - \bar{m}_i(m)]\lambda_i + \sum_{j=f+1}^{f+o} \Lambda_j \qquad (2)$$

and u(m), respectively.

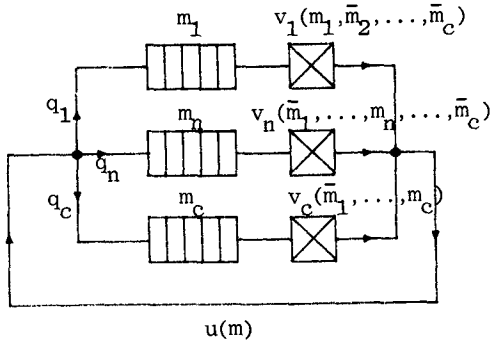
The solution of this queue is quite straightforward provided we have a means of evaluating $\bar{k}_i(k)$ and $\bar{m}_i(m)$ for TS job classes, and the conditional completion rate for transactions u(m). A simple approximation for $\bar{k}_i(k)$ as well as alternatives for the solution of the queue of Figure 3 are discussed in the Appendix. Assuming $\bar{m}_n(m)$ and u(m) known, we obtain from the solution of this queue the following quantities of interest

- p(m): the distribution of the number of jobs in the domain;
- $\bar{k}_n$, $\bar{m}_n$: the average numbers of class n, $n = 1,\ldots,c$, transactions awaiting admission and in the domain, respectively;
- θ_n: the throughput for class n transactions.

The average response time for a transaction of class n, R_n, can then be computed using Little's law [12]

$$R_n = (\bar{k}_n + \bar{m}_n)/\theta_n \ . \qquad (3)$$

The third and last step in the derivation of our approximate solution is the evaluation of $\bar{m}_n(m)$ and u(m), i.e., the average number of class n jobs in the domain given that the total domain population is m, and the conditional transaction throughput given m, respectively. To obtain these quantities, we apply the "short-circuit" approximation [13] which allows us to analyze the domain in isolation, under a constant load of m transactions.



$$m_1 + \ldots + m_c = m$$

$$m = 1,\ldots,M$$

Figure 4: Closed queueing network for computation of u(m)

The resulting closed queueing network is represented in Figure 4. In order for this system to possess a computationally efficient product-form solution, we let the routing probabilities q_n of joining the queue that represents class n transactions be independent of the network states and proportional to the average class n throughput

$$q_n = \theta_n \Big/ \sum_{i=1}^{c} \theta_i \quad . \tag{4}$$

This can be rewritten as

$$q_i = [N_i - \bar{k}_i - \bar{m}_i]\lambda_i/\theta, \quad i=1,\ldots,f \ ;$$

for TS transaction classes, and

$$q_j = \Lambda_j \Big/ \theta \ , \qquad j=f+1,\ldots,f+o \ ;$$

for TP transaction classes, where

$$\theta = \sum_{i=1}^{f} [N_i - \bar{k}_i - \bar{m}_i]\lambda_i + \sum_{j=f+1}^{f+o} \Lambda_j \quad .$$

Assume for the moment that the values for the service rates $v_n(\bar{m}_1,\ldots,m_n,\ldots,\bar{m}_c)$ and for the routing probabilities q_n are known. The service rates in the closed exponential queueing network of Figure 4 are functions of local queue lengths only so that the solution of this network can be obtained by any of a number of standard methods, e.g., [14,15]. Given that we are mainly interested in the average queue lengths and job throughputs in this network, Mean Value Analysis [15] is a convenient approach.

In accordance with the idea of the "short-circuit" approximation method, the values obtained for the average queue lengths and throughputs with a population of m jobs in the system of Figure 4 approximate the corresponding conditional queue lengths $\bar{m}_n(m)$ and throughput u(m). Note that with Mean Value Analysis, the throughputs for individual queues (classes of transactions), denoted by $u_n(m)$, are obtained at no additional cost.

In the approach outlined, we have reduced the solution of our original queueing network of Figure 2 to the solution of two simpler systems: the single server of Figure 3 and the closed network with c exponential servers of Figure 4. Since the values of $\bar{m}_n$ and $\bar{k}_i$ are needed to compute u(m), which in turn is needed to compute the $\bar{m}_n$ and $\bar{k}_i$, an iterative scheme suggests itself.

1. Choose initial values for $\bar{m}_n$, n = 1,...,c, and $\bar{k}_i$, i = 1,...,f, so that

$$\sum_{n=1}^{c} \bar{m}_n \leq M, \quad \text{and} \quad \bar{k}_i < N_i - \bar{m}_i \ , \quad i=1,\ldots,f.$$

2. Determine the service rates $v_n(\bar{m}_1,\ldots,m_n,\ldots,\bar{m}_c)$, n = 1,...,c. Let m_j^* be the closest integer to $\bar{m}_j$, j = 1,...,c. The following simple estimate, borrowed from [7], has been found to work well in practice

$$v_n(\bar{m}_1,\ldots,m_n,\ldots,\bar{m}_c) \approx v_n(m_1^*,\ldots,m_n,\ldots,m_c^*). \tag{5}$$

3. Solve the closed queueing network of Figure 4 with a total of m = 1,...,M jobs to obtain $\bar{m}_i(m)$ and u(m).

   The service rates are taken from step 2, and the routing probabilities q_n are computed using (4). Note that if the number of terminals for a TS class, N_i, is less than the domain capacity M, it is possible for the value of $\bar{m}_i(m)$ in the network of Figure 4 to exceed N_i in the course of iteration. (With the state independent values for q_i, nothing prevents states with $m_i > N_i$). In practice, this turns out to be an infrequent occurrence that can be remedied through simple scaling as described in the Appendix.

4. Solve the single server queue on Figure 3 to obtain p(m), $\bar{m}_n$, $\bar{k}_n$, θ_n (see Appendix).

5. If the maximum difference between old and new values for $\bar{m}_n$ is less than a given limit, stop the iteration. Otherwise, set $\bar{m}_n$ to the arithmetic average of new and old values for $\bar{m}_n$ (n = 1,...,c), and similarly for $\bar{k}_n$, and return to step 2.

   The averaging of iterates is used to avoid oscillations which otherwise may occur in some cases.

6. Compute R_n, the average response times for transactions of each class (n = 1,...,c).

We do not have a formal convergence proof for the above algorithm. It has been our experience, however, that it converges within a small number of iterations.

The main potential sources of inaccuracies of this approach lie in

a) the representation of the influence of other classes and, in particular, the use of $v_n(m_1^*,\ldots,m_n,\ldots,m_c^*)$ as the service rate for class n.

b) the use of state independent branching probabilities q_i for TS job classes.

c) the assumption used to obtain $\bar{k}_i(k)$ and $\bar{k}_i$ in the single server system (cf. Appendix).

d) the use of the "short-circuit" technique to obtain conditional queue lengths and throughputs.

Items b) and c) potentially affect more models with low numbers of terminals. Despite this impressive list of ills, in practice, the accuracy of our approach turns out to be fair, even with TS classes with low numbers of terminals. This is illustrated in the next section.

3. Numerical Results

A number of discrete-event simulations were run to assess the accuracy of the approach presented in the preceding section. In order to relate to previous work on shared domains [4,8], we have included in our runs simulations of models corresponding to cases 10 through 18 in Sauer's paper [4], used also in [8]. Note that, unlike [4], we simulated the queueing network of Figure 2 with service rates $v_n(m_1,\ldots,m_n,\ldots,m_c)$ obtained by a decomposition technique from a CPU-I/O Model. (These rates were also used in the analytical approximate solution.) Thus, our simulation results, although close to those of [4], pertain to a somewhat different model.

In all our examples, we report the values for the average number of transactions of each class in the admission queue, $\bar{k}_n$, in the domain, $\bar{m}_n$, and the average response time for transactions of class n, R_n (n=1,...,c). Simulation results are presented in the form of 90% level confidence intervals obtained using a regenerative technique with the same stopping rule as in [4]. For the analytical approximation, in addition to the actual values of $\bar{k}_n$, $\bar{m}_n$ and R_n, we report in parentheses the percentage deviation from the simulation point estimate.

Our first example corresponds to Sauer's case 18 (2 job classes, total domain capacity of 6) and represents one of the poorest showings of our approach. We observe that, even though most of the approximation values do not fall into the simulation confidence interval, the relative deviations remain moderate (20%).

Our second example illustrates what appears to be the typical accuracy of the approximation technique. It corresponds to Sauer's case 12 (2 job classes, total domain capacity of 2). Here, many approximation values fall into the confidence intervals, and the relative deviations from the simulation point estimate is in the 10% to 15% range.

Example 3 corresponds to case 17 (2 job classes, total domain capacity of 12) in [4], and is also one of the better cases for our approach. All approximation values are contained in the confidence intervals, and most of the relative deviations are on the order of 5%.

As a whole, our approximation fares quite well. These results were obtained using the first method described in the Appendix for the solution of the single server model. The second method discussed in the Appendix yields slightly less accurate results. This is illustrated in the Appendix.

Example 1: 2 job classes, domain capacity of 6

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(4.58,4.59)	(12.04,12.74)	(3.57,3.74)
	2	(1.38,1.39)	(0.49,0.54)	(8.76,9.29)
analyt	1	4.53(1.1%)	13.69(10.5%)	4.18(14.5%)
	2	1.46(5.6%)	0.61(18.3%)	10.74(19.1%)

Example 2: 2 job classes, domain capacity of 2

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(1.50,1.54)	(5.75,6.24)	(2.84,3.12)
	2	(0.40,0.45)	(0.26,0.29)	(5.10,5.64)
analyt	1	1.50(1.5%)	5.94(1.0%)	2.96(0.5%)
	2	0.47(9.6%)	0.29(6.5%)	6.15(14.6%)

Example 3: 2 job classes, domain capacity of 12

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(8.10,8.37)	(1.88,2.29)	(1.68,1.79)
	2	(2.06,2.22)	(0.04,0.06)	(10.84,13.39)
analyt	1	8.37(1.6%)	1.90(1.0%)	1.74(0.5%)
	2	2.22(3.9%)	0.06(7.3%)	13.24(9.3%)

With respect to computational complexity, the number of steps is on the order of

$$c\,M^2 + c\,(\sum_{i=1}^{f} N_i)$$

when using the first method for the solution of the single server model, and

$$c\,M^2$$

when using the second method. The computations performed for each step are relatively simple. This complexity compares favorably with the complexity of other approaches to the shared domain problem.

4. Conclusion

We have presented a simple and efficient approximation technique for models of multiprogrammed systems with shared domains. The approach reduces the solution of the original model with coupled queues and limited total multiprogramming level to the solution of a single server exponential queue and that of a simple closed queueing network. Both resulting systems possess a product-form solution.

Iteration between these two models is performed until a self consistent state is reached. Usually, a small number of iterations is needed, and the computational complexity grows only moderately with the number of classes and the maximum multiprogramming degree. The accuracy of the results is in fair agreement with discrete event simulation.

5. References

1. W.W. Chiu and We-Min Chow, "A Performance Model of MVS," IBM Syst. J. 17 (1978).

2. A. Brandwajn and J.A. Hernandez, "A Study of a Mechanism for Controlling Multiprogrammed Memory in an Interactive System," IEEE Trans. on Soft. Engin. SE-7 (1981) 321-331.

3. R.M. Brown, J.C. Browne and K.M. Chandy, "Memory Management and Response Time," Comm. ACM 20 (1977) 153-165.

4. C. Sauer, "Approximate Solution of Queueing Networks with Simultaneous Resource Possession," IBM J. Res. Develop. 25 (1981) 894-903.

5. D.A. Menasce and V.A.F. Almeida, "Operational Analysis of Multiclass Systems with Variable Multiprogramming Level and Memory Queueing," Computer Performance 3 (1982) 145-159.

6. A. Brandwajn, "A Model of a Time-Sharing System with Two Classes of Processes," Gesellschaft fuer Informatik, 5 Jahrestagung, Lecture Notes in Computer Science, vol. 34 (1975), Springer Verlag, 547-566.

7. A. Brandwajn, "Fast Approximate Solution of Multiprogramming Models," Proc. 1982 SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 141-149.

8. E. Lazowska and J. Zahorjan, "Multiple Class Memory Constrained Queueing Networks," Proc. 1982 SIGMETRICS Conference on Measurement and Modelling of Computer Systems, 130-140.

9. IBM, IMS/VS Version 1 General Information Manual, GH20-1260-11, June, 1982.

10. D.R. Cox, "A Use of Complex Probabilities in the Theory of Stochastic Processes," Proc. Camb. Phil. Soc. 51 (1955) 313-319.

11. F. Baskett, F.M. Chandy, R.R. Muntz and F.G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers," J. Ass. Comput. Mach. 22 (1975) 248-260.

12. J.D. Little, "A Proof of the Queueing Formula $L = \lambda W$", Oper. Res. 9 (1961) 383-387.

13. K.M. Chandy, U. Herzog and L. Woo, "Parametric Analysis of Queueing Networks," IBM J. Res. Develop. 19 (1975) 36-42.

14. J. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers," Comm. ACM 16 (1973) 527-531.

15. M. Reiser and S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," J. Ass. Comput. Mach. 27 (1980) 313-322.

16. P. Schweitzer, "Approximate Analysis of Multiclass Closed Networks of Queues," Presented at the Int. Conf. Stochastic Control and Optimization, Amsterdam, 1979.

Appendix

1. Solution of reduced single server model

Let $\ell = k + m$ be the total number of transactions of all classes in the system. Probably the most straightforward approach to the solution of the system of Figure 3 is to view this model as a single server state dependent M/M/1 queue. The state of the queue is given by ℓ, the arrival rate is

$$\alpha(\ell) = \begin{cases} \lambda(0,\ell) , & \ell=0,\ldots,M-1 \\ \lambda(\ell-M,M) , & \ell=M,M+1,\ldots ; \end{cases} \qquad (6)$$

and its service rate is

$$\beta(\ell) = \begin{cases} u(\ell) , & \ell=1,\ldots,M-1 \\ u(M) , & \ell=M,M+1,\ldots . \end{cases} \qquad (7)$$

$u(m)$ and the $\bar{m}_i(m)$ entering into the expression of $\lambda(k,m)$ are computed in the solution of the system of Figure 4. Assume that the only missing element, the $\bar{k}_i(k)$ in $\lambda(k,m)$, is known.

Denote by p_ℓ the equilibrium probability of ℓ transactions in the system. Formally,

$$p_\ell = \frac{1}{G} \prod_{n=1}^{\ell} \alpha(n-1)/\beta(n), \qquad \ell=0,1,\ldots \qquad (8)$$

where G is a normalization constant

$$G = \sum_{\ell=0}^{\infty} \prod_{n=1}^{\ell} \alpha(n-1)/\beta(n) . \qquad (9)$$

This solution exists iff G converges. If there are only TS classes, the maximum value of ℓ is $L = \sum_{i=1}^{f} N_i$, and G always converges.
With a mixture of TS and TP classes, we define an arbitrary value for ℓ, ℓ_m ($\ell_m \geq \max(L,M)$), at which we assume that all TS transactions are in the system so that

$$\alpha(\ell) = \sum_{j=f+1}^{f+o} \Lambda_j , \qquad \ell \geq \ell_m .$$

Note that the service rate for ℓ_m is a constant. The computation of p_ℓ is thus partitioned into a finite part where (8) is used for $\ell = 0,\ldots,\ell_m-1$, and an infinite part where

$$p_\ell = p_{\ell_m-1} \rho^{\ell-\ell_m+1} \qquad (10)$$

with $\rho = \alpha(\ell_m)/\beta(\ell_m)$. The evaluation of the infinite part contribution to the normalizing constant G is straightforward.

The $\bar{k}_i(k)$ needed to compute $\lambda(k,m)$ can be estimated as follows. Let $\hat{N}_i = N_i - m_i(m)$. $[\hat{N}_i - \bar{k}_i(k)]\lambda_i$ is the rate of arrivals of class i transactions given k. By analogy with the case of TP classes only, we assume that $\bar{k}_i(k)$ is proportional to $[\hat{N}_i - k_i(k)]\lambda_i$. Hence,

$$\bar{k}_i(k) = [\hat{N}_i - \bar{k}_i(k)]\lambda_i \; k/\theta(k), \quad i=1,\ldots,f; \qquad (11)$$

$$\bar{k}_j(k) = \Lambda_j \; k/\theta(k) , \qquad j=f+1,\ldots,f+o,$$

where

$$\theta(k) = \sum_{i=1}^{f} [\hat{N}_i - \bar{k}_i(k)]\lambda_i + \sum_{j=f+1}^{f+o} \Lambda_j .$$

The system of equations (11) can be easily solved by a fast converging iteration.

Note that (11) implies an equal waiting time in the admission queue for all job classes. Strictly speaking, this is the case for TP classes but not, in general, for TS classes, although one would expect the approximation to become better as the numbers of terminals increase.

Using the same idea of equal waiting time in the admission queue, it is possible to simplify the computation of $\bar{k}_n$, the average number of class n jobs in this queue, and θ_n, the transaction throughput. Given p_ℓ, we readily obtain the average number of class n jobs in the domain

$$\bar{m}_n = \sum_{\ell=0}^{M-1} \bar{m}_n(\ell)p_\ell + \sum_{\ell=M}^{\infty} \bar{m}_n(M)p_\ell , \qquad (12)$$

the total average number of transactions awaiting admission

$$\bar{k} = \sum_{\ell=M}^{\infty} (\ell-M)\, p_\ell , \qquad (13)$$

and total job throughput

$$\theta = \sum_{\ell=0}^{M-1} u(\ell)p_\ell + \sum_{\ell=M}^{\infty} u(M)p_\ell . \qquad (14)$$

Assuming equal waiting time for admission yields

$$\bar{k}_i = (N_i - \bar{m}_i)\lambda_i/[\theta/\bar{k} + \lambda_i] \quad \text{and} \qquad (15)$$

$$\theta_i = \bar{k}_i\theta/\bar{k} ,$$

for TS classes, and

$$\bar{k}_j = \Lambda_j\bar{k}/\theta , \qquad (16)$$

for a TP class.

It is also possible to devise an approach in which only a subset of values of p_ℓ are computed explicitly. Denoting by $p(m)$, $m=0,\ldots,M$, the probability distribution for the total number of jobs in the domain, we have

$$p(m) = \frac{1}{H} \prod_{i=1}^{m} \lambda(0,i-1)/w(m) , \quad m=0,\ldots,M, \qquad (17)$$

where

$$w(m) = \begin{cases} u(m) , & m=1,\ldots,M-1; \\ u(M)\ \text{Prob}\{k=0 \mid m=M\} , & m=M; \end{cases} \qquad (18)$$

and H is a normalization constant.

In order to use (17), we miss the value of the conditional probability that the admission queue is empty given that the domain is fully populated. As an approximation, consider the admission queue with the number of jobs in the domain maintained at M. Let k(M) be the current number of transactions in the admission queue, $k(M)=0,\ldots$. The rate of departures from this queue is $u(M)$ $(k(M)=1,\ldots)$, and arrivals of TS transactions of class i, $i=1,\ldots,o$, are generated by $\hat{N}_i = N_i - \bar{m}_i(M)$ terminals with average think time $1/\lambda_i$; TP transactions arrive with rate Λ_j, $j=f+1,\ldots,f+o$.

Disregarding the fact that, in general, the $\hat{N}_j$ are not integer numbers, we solve this queue through approximate mean value analysis (cf. [16]). Denote by $W_n(M)$ the average waiting time of jobs of class n, by $\theta_n(M)$ the transaction throughput, and by $\bar{k}_n(M)$ the average number of class n jobs in the queue, $n=1,\ldots,c$. We use

$$\begin{aligned} W_i(M) &= [\tilde{k}_i + 1] \ / \ u(M), \\ \theta_i(M) &= \hat{N}_i \ / \ [W_i(M) + 1/\lambda_i \ ], \\ \bar{k}_i(M) &= \theta_i(M)\ W_i(M), \quad i = 1,\ldots,f; \end{aligned} \qquad (19)$$

for TS classes, and

$$\begin{aligned} W_j(M) &= [\bar{k}(M) + 1] \ / \ u(M), \\ \bar{k}_j(M) &= \theta_j(M)\ W_j(M), \quad j = f+1,\ldots,f+o; \end{aligned} \qquad (20)$$

for TP classes, where $\theta_j(M) = \Lambda_j$, and

$$\tilde{k}_i = \max\ [1-1/\hat{N}_i,\ 0]\ \bar{k}_i(M) + a\ [\bar{k}(M)-\bar{k}_i(M)], \qquad (21)$$

with

$$a = \begin{cases} 1 - 1 \ / \ \sum_{i=1}^{f} \hat{N}_i & \text{, when there are TS classes only} \\ 1 & \text{, when TP classes are present.} \end{cases}$$

This set of equations is solved through iteration.

We let

$$\text{Prob}\{k=0 \mid m=M\} \simeq 1 - \sum_{n=1}^{c} \theta_n(M) \ / \ u(M) .$$

This allows us to evaluate the $p(m)$, and we obtain $\bar{k}$, the average total number of transactions in the admission queue, from

$$\bar{k} = \bar{k}(M)\ p(M) .$$

$\bar{k}_n$ and θ_i are obtained from (15) and (16).

2. Scaling Procedure

If, following step 3 in the iteration, $\bar{m}_i(m) > N_i$ for a TS class, then the following scaling procedure is used.

1. Check all TS classes and find $\bar{m}_i(m)$ such that $\bar{m}_i(m) > N_i$. If $\bar{m}_i(m) \le N_i$ for all TS classes, then the procedure can stop; otherwise correct $\bar{m}_i(m)$ as follows.

2. Let

$$\text{scale factor} = \frac{[\bar{m}_i(m) - N_i]}{m - \sum_{n} \bar{m}_n(m)}$$

where the sum is over all TP classes and TS classes such that $\bar{m}_n(m) < N_n$, and set $\bar{m}_i(m)$ to N_i .

3. For all TP classes and all TS classes with $\bar{m}_n(m) < N_n$ (but not $\bar{m}_n(m) = N_n$) :

$$\bar{m}_n(m) = \bar{m}_n(m) * (1 + \text{scale factor}) .$$

4. Return to step 1.

The procedure will terminate in at most f passes since one $\bar{m}_i(m)$ is corrected on each pass, and, if there are no TP classes, $m \le \sum N_i$. Note that the sum of the $\bar{m}_n(m)$ will still equal m when the procedure is completed.

3. Description of Models Used

The models used correspond to cases 10 through 18 in Sauer's paper [4]. They are described below using the notation of our paper.

Case	c	M	N_1	N_2	$1/\lambda_1$	$1/\lambda_2$
10	2	6	20	2	5 s	10 s
11	2	4	20	2	5 s	10 s
12	2	2	20	2	5 s	10 s
13	2	9	30	3	5 s	10 s
14	2	6	30	3	5 s	10 s
15	2	3	30	3	5 s	10 s
16	2	18	40	4	5 s	10 s
17	2	12	40	4	5 s	10 s
18	2	6	40	4	5 s	10 s

4. Results for Second Method

Example 1: 2 job classes, domain capacity of 6

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(4.58,4.59)	(12.04,12.74)	(3.57,3.74)
	2	(1.38,1.39)	(0.49,0.54)	(8.76,9.29)
analyt	1	4.48(2.2%)	13.96(12.7%)	4.28(17.0%)
	2	1.46(5.0%)	0.62(21.3%)	10.82(20.0%)

Example 2: 2 job classes, domain capacity of 2

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(1.50,1.54)	(5.75,6.24)	(2.84,3.12)
	2	(0.40,0.45)	(0.26,0.29)	(5.10,5.64)
analyt	1	1.47(3.1%)	6.16(2.6%)	3.08(3.6%)
	2	0.44(2.8%)	0.31(13.1%)	6.01(12.0%)

Example 3: 2 job classes, domain capacity of 12

solution	class	$\bar{m}_n$	$\bar{k}_n$	R_n
simul	1	(8.10,8.37)	(1.88,2.29)	(1.68,1.79)
	2	(2.06,2.22)	(0.04,0.06)	(10.84,13.39)
analyt	1	8.31(0.9%)	2.11(0.9%)	1.76(1.4%)
	2	2.22(3.7%)	0.06(19.4%)	13.27(9.5%)