Performance Evaluation 17 (1993) 141–154
North-Holland

Aspects of the solution of some multiclass loss systems

Alexandre Brandwajn and Anil K. Sahai
Department of Computer Engineering, University of California, Santa Cruz, CA, USA

Received 16 May 1991

Abstract

Brandwajn, A. and A.K. Sahai, Aspects of the solution of some multiclass loss systems, Performance Evaluation 17 (1993) 141–154.

Stochastic service systems in which no queueing is allowed have been used for quite some time as models of telephone systems. Recently, such loss systems with a finite number of request sources were shown to be of interest in the modeling of path contention occurring in many current I/O computer architectures.

This paper considers two aspects of finite source loss systems with several classes of requests ("customers") arising, for instance, in the representation of more realistic I/O workloads. First, we present a simple approach to the solution of such systems in the case of classical multiple servers. This approach is based on the use of recurrence relationships among conditional averages, and leads to an efficient computation of server utilizations and loss probabilities.

In the second part of this paper, we consider generalized loss systems in which the service resources are viewed as a global quantity (rather than a specific fixed number of servers), and each class of customers requests a given amount of the global resource. Such systems are useful, for example, as models of bus bandwidth shared among several types of transfers. We find that, although these systems exhibit a product-form solution, the simple recurrences derived for regular multiserver loss systems do not carry over to the generalized systems. Therefore, we present an alternative approach based on equivalence techniques which produces the solution of such generalized systems through repeated solutions of a single-class loss model. A comparison with the existing convolution method for evaluating the product-form solution indicates that the proposed method is significantly more efficient both in terms of computational complexity and computer memory requirements.

Keywords: multiserver loss systems; multiple classes of requests; generalized Engset model; simultaneous server acquisition; recurrent solution; equivalence approach; application to bus bandwidth sharing.

1. Introduction

If no queueing is allowed in a service system, requests that arrive to find all servers busy are subject to loss (i.e., forced to leave the system), which may or may not mean that they will be reissued again later ("blocked calls cleared" or "block calls retried", in telephone terminology). Such systems have been widely used in the past as models of telephone exchanges. Recently, the evaluation of the performance of current computer disk I/O subsystems has provided a new domain of application for loss models [6]. Indeed, a disk device typically needs to use one of the data transfer paths that link it to the central processing unit only during a small portion of the overall request service time. Hence, it is common to organize some number of disks (several tens) into a "string" sharing a given smaller number of transfer paths. The devices disconnect from the paths for the duration of mechanical orientation motion, and request reconnection to a transfer path when the required data area is about to pass under the read/write head. Of course, since there are typically many more disks in a string than

Correspondence to: A. Brandwajn, Department of Computer Engineering, University of California, Santa Cruz, CA 95064, USA. Email: alexb@cse.ucsc.edu or A.K. Sahai, Mail Stop 139, Building 03, Amdahl Corporation, 1250 East Arquest Avenue, Sunnyvale, CA 94088, USA. Email: aks20@spg.amdahl.com.

there are transfer paths, it can happen that no path is available when a device needs it. The disk then temporarily loses its opportunity to transfer data, and will try again to acquire a transfer path after a full device revolution.

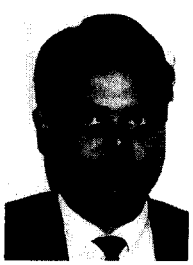
There is a conceptual similarity between an attempt to dial and redial a call and an attempt by a disk to reconnect to one of the string transfer paths, and comparisons with discrete event simulations indicate that a loss model is a valuable tool when assessing the expected delays caused by the inability of a disk device to acquire a transfer path when needed. In I/O subsystem modeling, the disk devices may be viewed as sources of requests (customers) competing for the use of the transfer paths (the servers). As mentioned above, the number of disks in a string is relatively low, and, since the level of activity and the transfer times (i.e., the path holding times) can vary widely from one disk to another, it is natural to represent the reconnection process through a finite source loss model with several classes of customers and the number of servers equal to the number of paths.

The resulting model is known in the teletraffic literature as the generalized Engset model, and a number of its variations have been considered by several authors e.g. [10,12,16,26]. The Engset models have been shown to possess a product-form solution and to exhibit properties of distributional robustness. As noted in [20], the computational evaluation of their solution presents a challenge because the size of the system state space grows extremely rapidly with the dimensions of the model. It is on the computationally efficient solution of such models that we focus in this paper.

In the modeling of telephone trunks or discrete I/O transfer paths, the number of servers is given and fixed. This is not necessarily the case when a loss system is used to model the acquisition of bus bandwidth in a modern disk controller. Indeed, some controllers (Amdahl Storage Processor [2]) include an internal bus used for all transfers. In order to proceed, a disk transfer must acquire the bandwidth corresponding to its data rate. With increasing disk data rates, the available bus bandwidth may not be sufficient to always provide for all transfers. Also, recent announcements of IBM disks [1] allow one to mix faster (3390) and slower (3380) devices on the same controller. This leads to a generalized multiclass loss model in which the service resource gets partitioned into a variable number of servers, depending on the requirements of the customers (requests) currently in service. In the

Alexandre Brandwajn received the Ingéneur Civil degree from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1971, and the Doctor-Ingénieur and Docteur ès-Sciences degrees from the University of Paris VI in 1972 and 1975, respectively. From 1971 to 1975, he was a researcher at IRIA-Laboria, France, in the area of operating system performance evaluation and solution methods for queueing models of computer performance. In 1975 he joined the Ecole Nationale Supérieure de Télécommunications where he was a professor of computer science until 1979. From 1979 through 1985, he was with Amdahl Corporation, Sunnyvale, California, where he was the manager of Systems Analysis. He is currently professor of computer engineering at the University of California, Santa Cruz. His research interests include the areas of analytical and numerical methods for queueing models, computer architecture, and fault tolerance techniques.

Dr. Brandwajn is the author of a number of articles and conference presentations, and a member of the Association for Computing Machinery.



Anil K. Sahai received his B.A. (Honors) in mathematics from St. Stephen's College, Delhi, India, his M.S. in mathematics/computer science from Ohio University, Athens, OH, his M.S. in computer science from the University of Pittsburgh, PA, and his Ph.D. in computer and information science from the University of California, Santa Cruz, CA. He has been working in the Systems Performance Architecture group at Amdahl Corporation in Sunnyvale, CA, since April 1990. His research interests are in the areas of network performance modeling, performance analysis of storage systems, queueing systems, decision theory, and design and analysis of algorithms. Presently, he is involved in the architecture, design and performance modeling of I/O subsystems.

case of I/O busses, the bus bandwidth is the total resource, and the data rates of individual transfers define its partitioning.

Systems in which customers acquire simultaneously several servers have been studied by a number of authors, e.g. [14,15,19,20]. In particular, a model similar (albeit different) to ours is discussed in [20]. This generalized Engset system possesses a product-form solution, and Iversen proposes a convolution type of algorithm for its computational evaluation. Note that in our model, unlike in [20], the requirements of individual request may represent arbitrary fractions of the total available service resource. We introduce a new computational approach to the evaluation of the product-form solution of our system. This approach appears to significantly reduce the computational complexity of the solution of our model.

Both in telephone systems and in computer modeling, the probability of successfully acquiring the resource (or, equivalently, its complement, the loss probability referred to as call congestion in teletraffic literature) is among the performance measures of direct interest. Denote by C the total number of distinct request classes, and let N_i be the number of sources of class i requests ($i = 1,\ldots,C$). Let also U_i be the average number of class i requests in service (using the resource). By considering the ratio of the number of requests successfully processed to the total number of requests issued by a given customer class, it is easy to see that the probability of successfully acquiring the service resource by a class i customer can be expressed as

$$s_i = \frac{U_i}{\rho_i(N_i - U_i)} \tag{1}$$

where $\rho_i = \lambda_i/\mu_i$, λ_i is the (idle time) request rate of a class i source, and μ_i is the service rate for a customer of class i ($i = 1,\ldots,C$).

Note that (1) is valid both in the case of classical regular servers and of global service resource. The use of this Little's type of relationship avoids the explicit enumeration of states in the computation of the success probability, and thus contributes to the reduction of overall computational complexity.

If the ρ_i are known, the solution of the loss systems model must yield the average numbers in service (U_i). In computer modeling applications, one often analyzes the performance for a specific assumed I/O rate, so that the attained ("carried") rates of requests for each class are taken as given. Then, a fixed-point iteration can be used to determine the ρ_is from the known U_is. In such an iteration, the loss system is solved a number of times with different ρ_i values, and the obtained average numbers of customers in service are compared with the target values U_i, $i = 1,\ldots,C$. Thus, both when the ρ_i are known and when the U_i are given, it is important to be able to efficiently compute the average numbers of requests in service in a multiclass loss system.

Section 2 of this paper presents a simple approach to the solution of such systems in the case

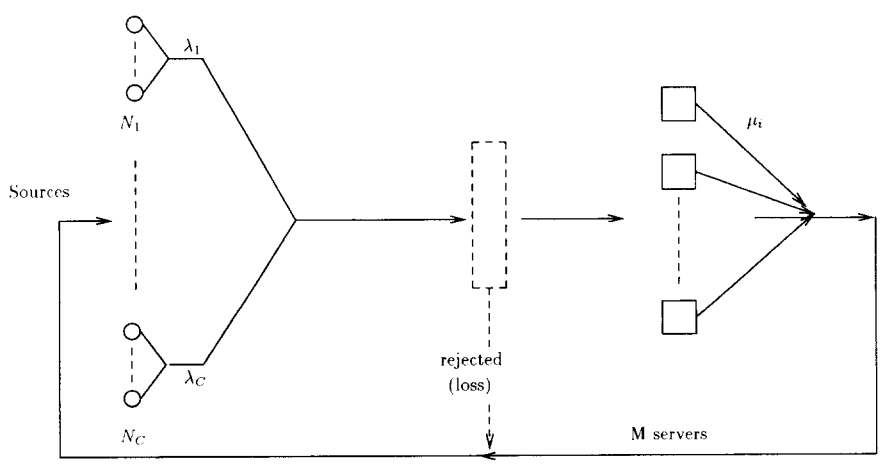


Fig. 1. Multiclass multiserver loss system.

of a classical generalized Engset loss model. This approach is based on the use of recurrence relationships which are shown to exist among conditional averages, and leads to an efficient computation of server utilizations (and, hence, loss probabilities, i.e., call congestion).

In Section 3, we consider a generalized loss system with a global service resource. Although such a model exhibits a product-form solution, we find that the simple recurrences derived in Section 2 for the "regular" Engset loss system do not carry over to this generalized model. Therefore, we present an alternative approach based on state equivalence techniques which reduces the solution of such generalized systems to repeated solutions of a simple single-class loss model. A comparison with the convolution algorithm of [20] shows that our method tends to result in significant computational savings.

In Section 4, we present an application of the generalized loss system to the evaluation of a bus shared by disk devices with different data transfer rates. These results show some unexpected trade-offs related to bus bandwidth contention.

Finally, Section 5 briefly summarizes the conclusions of this paper and addresses the issue of adequacy of a memoryless loss system as a model of disk service reconnection.

2. Recurrent solution of multiclass loss models

Consider an Engset loss system with a total of M identical servers ($M \geqslant 1$) and C classes of customers (Fig. 1). As discussed in the introduction, N_i denotes the number of request sources of class i, λ_i is the idle time request rate generated by a class i source, and μ_i is the rate of service for a class i customer. We assume memoryless distributions for the request generation times, i.e., the times customers of each class spend at their "source". We also assume that, if a request finds all servers busy, the source will generate another request after an exponentially distributed time with the same mean $1/\lambda_i$ as for originating requests. For simplicity, the service times are also taken to be exponentially distributed with means $1/\mu_i$. Such a generalized Engset loss system is known to be insensitive to the distributions of inter-arrival and service times (cf. [10]).

We focus our attention on the steady-state behavior of the system described. Under our memoryless assumptions, it is straightforward to obtain the global balance equations for the loss model of Fig. 1. These equations are given in Appendix A. Note that the Markov chain underlying our model is clearly irreducible and aperiodic, so that, given the finite number of states, the existence of a steady state is guaranteed (cf. [3]). It is easy to show that local balance (separately for each request class) holds in such a loss system, so that its equations possess a product-form solution for the joint probability distribution $p(m_1, \ldots, m_C)$, where m_i is the number of class i requests in service (cf. [10,16,23,26]). This solution involves products of factors pertaining to each request class, as well as a normalizing constant. As is usually the case, the cardinality of the state space grows quickly with the number of classes and serves. Recall that, for our purposes, we are mostly interested in the average numbers of requests of each class in service. For queueing networks with product-form solutions, several efficient algorithms exist allowing the computation of specific averages without the explicit evaluation of the probability of every state [5,8,9,11, 18,21,22]. Unfortunately, they do not seem to extend readily to the loss system under consideration. It is, however, possible to base a quick computation of the performance measures of interest on a simple recurrence for conditional averages.

Let $p(m_i, m)$ denote the stationary probability that there are m_i customers of class i in service and a total of m servers are busy ($m_i = 0, \ldots, \min(N_i, m)$, $m = 0, \ldots, M$). Local balance holding for this system implies the following relationship

$$p(m_i, m)\mu_i m_i = p(m_i - 1, m - 1)(N_i - m_i + 1)\lambda_i \qquad (2)$$

Denote by $p(m_i \mid m)$ the conditional probability of having m_i class i requests in service given that a total of m servers are busy, and by $p(m)$ the probability of the latter event. With these notations, (2) yields

$$p(m_i \mid m)m_i = p(m_i - 1 \mid m - 1)(N_i - m_i + 1) \times \rho_i p(m - 1)/p(m) \qquad (3)$$

By summing (3) over values of $m_i \geqslant 1$, we get

$$\overline{m}_i(m) = \left[N_i - \overline{m}_i(m-1)\right]\rho_i p(m-1)/p(m) \tag{4}$$

where $\overline{m}_i(m)$ is the conditional average number of class i requests in service given that a total of m servers are busy. Of course, we must have $\sum_{i=1}^{C}\overline{m}_i(m) = m$, so that

$$\overline{m}_i(m) = m\left[N_i - \overline{m}_i(m-1)\right]\rho_i \Big/ \sum_{j=1}^{C}\left[N_j - \overline{m}_j(m-1)\right]\rho_j \tag{5}$$

and

$$p(m) = \frac{1}{G}\prod_{i=1}^{m}\left[\sum_{j=1}^{C}\left[N_j - \overline{m}_j(i-1)\right]\rho_j/i\right], \quad m = 0, 1, \ldots, M, \tag{6}$$

where G is a normalizing constant for (5).

Note that (5) represents in fact a simple recurrence for the conditional averages $\overline{m}_i(m)$. Starting with the trivially known $\overline{m}_i(0) = 0$, it allows us to consecutively compute conditional average numbers of requests of each class in service for increasing numbers of busy servers. The regular nonconditional averages can be easily computed as

$$U_i = \sum_{m=1}^{M}\overline{m}_i(m)p(m) \tag{7}$$

The actual computation can take advantage of the fact that the numerator in the solution factor of (6) involves the same sum of expressions with the conditional averages for $m-1$ as the right hand side of (5). This allows us to proceed as follows. For increasing values of m, we first compute the sum $\sum_{j=1}^{C}\overline{m}_j(m-1)\rho_j$. We use it both to compute the next value of the factor in (6), and to obtain new values for the $\overline{m}_i(m)$. Since the averages U_i can be cumulated as we go, it is clear that a single array of values can be used for the $\overline{m}_i(m)$ across all values of m.

From a computational standpoint, the numerical stability of a recurrence like (5) is always of concern. In particular, with very different ρ_is and N_is the sum of expressions in the denominator of (5) can be subject to loss of accuracy, leading to incorrect values for the $\overline{m}_i(m)$ at the next step. Indeed, we have observed such loss of accuracy under extreme conditions. Fortunately, the following simple steps can be applied in practice. A loss of accuracy for class i becomes obvious when the conditional average with a total of m busy servers computed from (5) exceeds N_i (or becomes less than the average with $m-1$). From that point on, we set the conditional average for class i to N_i, and we remove this class from the pool considered in the sum in (5), i.e., we use $(m - N_i)$ as the sum of the remaining conditional averages. Clearly, it is possible that more than one class of requests will be thus removed.

Table 1
Results for average number of users in service

Method	Class 1	Class 2	Class 3
Direct solution of equations	16.0181	0.2037	34.0071
Recurrence with removal	16.0057	0.2000	33.9994
Relative error (%)	0.0774	1.8163	0.0226

Parameters: no. of classes = 3; no. of servers = 60; $N_1 = 20$, $\mu_1 = 1.0$, $\lambda_1 = 4.0228$, $N_2 = 30$, $\mu_2 = 1.0$, $\lambda_2 = 6.8380$, $N_3 = 35$, $\mu_3 = 1.0$, $\lambda_3 = 3.4249$.

The occurrence of such a visible loss of accuracy tends to be confined to rather extreme cases, and can be minimized by rearranging the computation of the sum in (6) so as to cumulate smallest terms first. As indicated by a number of direct numerical solutions of the balance equations as well as by discrete-event simulations, the procedure outlined above tends to have little adverse effect on the overall accuracy of the averages U_i. Table 1 shows a comparison of the results of a direct numerical solution of the balance equations and of our recurrence for 3 classes and 60 servers which happens to be one of the worst cases among the ones explored. Here, class no. 3 was most subject to removal by our procedure.

We observe that the relative errors remain quite low.

Overall, we think that (5) and (6) represent a useful way of computing the average numbers of requests of each class in service in a multiserver loss system without having to enumerate detailed system states. As a "free" by-product, we also obtain the probability distribution of the occupation of the servers. Note that the computational complexity of the proposed algorithm is proportional to the product CM, i.e., grows linearly with the number of classes and the number of servers. The storage requirements of the algorithm are quite modest.

The multiclass Engset loss system considered in this section has been studied by a number of authors. The product-form of the solution is given, in particular, in [16] but without an efficient method to evaluate it. Cohen [10] has studied the distributional robustness of the Engset formula. His paper contains also an expression for the call congestion for each class (formula 4.4 in [10]). However, the evaluation of this formula requires the enumeration of system states, and, most probably, an application of an iterative solution method, e.g., of the Newton–Raphson type. Thus, we believe that our method represents a novel computationally efficient approach to the evaluation of average numbers in service, and, hence, individual call congestion.

The next section is devoted to the efficient evaluation of generalized loss systems with a total service resource partitionable into varying numbers of servers.

3. Generalized multiclass loss models

Consider now a generalized Engset system with a total service resource of, say, B units (e.g., bus bandwidth of B MBytes/s) as shown in Fig. 2. Again, there are C classes of customers, N_i is the number of sources of requests of class i, and λ_i is the idle time rate of requests generated by a class i source. A class i request requires a "chunk" of size b_i of the global resource (e.g., disk data rate of b_i MBytes/s), and uses it for an average time $1/\mu_i$ (e.g., data transfer time). For the sake of simplicity, both the sources and the resource holding times are taken to be memoryless.

As before, we are interested in the average numbers of requests of each class in service. Unlike in the classical loss system, however, we now have no fixed number of servers. Rather, the service resource gets partitioned into variable numbers of servers, depending on the class make-up of the requests being served. The system considered exhibits some similarity to multiserver models in which requests may require several servers to proceed (cf. [5,13,14,15,17,18,20,21,24, 25]) with the difference that, in our case, the individual resource requirements b_i can represent arbitrary fractions of the total service resource B.

The system described possesses a simple product-form solution for the joint probability $p(m_1,\ldots,m_C)$ where m_i, $i=1,\ldots,c$ is the number of class i requests being served. For each class, local balance holds between service activations and completions in a feasible state, leading to

$$p(m_1,\ldots,m_C) = \frac{1}{G}\prod_{i_1=1}^{m_1}(N_1-i_1+1)\rho_1/i_1 \prod_{i_2=1}^{m_2}(N_2-i_2+1) \times\rho_2/i_2\cdots\prod_{i_C=1}^{m_C}(N_C-i_C+1)\rho_C/i_C \qquad (8)$$

where G is a normalizing constant corresponding to the sum of the product terms over all feasible states, i.e., those compatible with the total resource and individual class requirements.

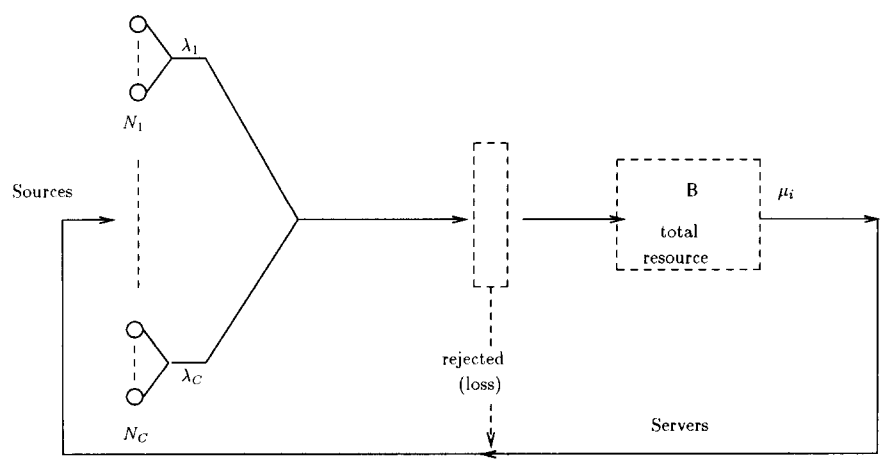


Fig. 2. Generalized multiclass loss model.

Note that, in most cases, our model can be mapped onto the multi-service loss system considered in [20] by converting all resource quantities involved (i.e., B and the b_is) into integers. The resulting total resource is then viewed as a given, potentially very large, number of servers. Iversen [20] gives the product-form solution for such a model in terms of a state description involving the numbers of busy servers, and states the insensitivity of the steady-state solution to the distribution of holding times. Here, we concentrate on efficient computation of the probability that a request is able to successfully acquire the needed resource (or, equivalently, the call congestion) for each class of requests. For reasons which will become apparent shortly, we keep the state description introduced earlier in this section, i.e., in terms of the current numbers of requests of each class in service.

Unlike for a regular loss system with a fixed number of servers, the maximum number of customers of a given class in service does not necessarily increase when the total number of busy servers increases. For example, consider a system with a total resource of 10 and 2 classes of requests with resource requirements of 4 and 2, respectively. With a total of 3 requests in service, there can be 0, 1 or 2 class 1 requests in service (and thus 3, 2, and 1 requests of class 2). When there are a total of 4 requests being served, however, there can be only 0 or 1 customers in service, since 2 requests of class 1 would leave only 2 units of resource, i.e., enough for only one customer of class 2 (while 2 would be required for a total of 4). For this reason, a recurrence like (5) does not appear to hold for the generalized loss system, so that we look for another way to efficiently evaluate its solution.

An efficient way of organizing the enumeration of system states, which allows a significant degree of computational savings, is an approach based on a repeated application of the equivalence and decomposition method (cf. [7]). Rather than develop general notations, let us illustrate this approach using an example with three classes of requests.

Denote by $p(m_1, m_2, m_3)$ the joint probability distribution of requests in service. The basis for the approach is the following probability identity:

$$p(m_1, m_2, m_3) = p(m_3 \mid m_2, m_1) \cdot p(m_2 \mid m_1) \cdot p(m_1) \qquad (9)$$

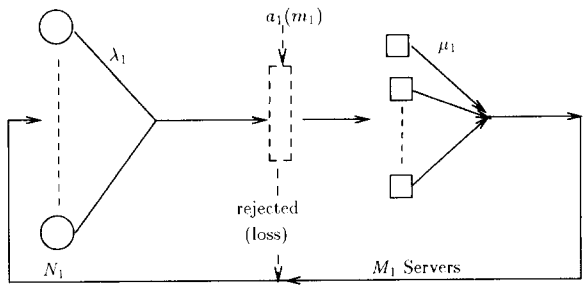


Fig. 3. Equivalent loss system for class 1.

where $p(m_3 \mid m_2, m_1)$ is the conditional probability distribution of m_3 given m_2 and m_1, $p(m_2 \mid m_1)$ is the conditional distribution of m_2 given m_1, and $p(m_1)$ is the marginal probability of having m_1 class 1 customers in service. Consider class 1 requests. The available resource allows for a maximum of $M_1 = \lfloor B/b_1 \rfloor$ requests in service ($\lfloor \cdot \rfloor$ denotes the 'integer' function), so that, for class 1, the system appears as a loss system with M_1 servers (see Fig. 3). The interference of customers of other classes is represented through the "activation" function $a_1(m_1)$ defined as follows

$$a_1(m_1) = \sum_{m_2} \hat{a}_1(m_2, m_1) p(m_2 \mid m_1) \qquad (10)$$

where

$$\hat{a}_1(m_2, m_1) = \sum_{m_3 : B - m_1 b_1 - m_2 b_2 - m_3 b_3 \geq b_1} p(m_3 \mid m_2, m_1) \qquad (11)$$

Clearly, the activation function $a_1(m_1)$ corresponds simply to the probability that (given m_1) other classes leave enough resource for at least one additional class 1 request. Indeed, the summation in (11) is over all values of m_2 such that, with the given values of m_1 and m_2, there is at least b_1 resource left available. With known $a_1(m_1)$, the solution of such a loss model is quite simple:

$$p(m_1) = \frac{1}{G} \prod_{i=1}^{m_1} (N_1 - i + 1) a_1(i-1) \rho_1 / i \qquad (12)$$

To compute $a_1(m_1)$, consider our system with m_1 customers of class 1 in service. This leaves $B - b_1 m_1$ as the available system resource. Thus, for class 2 customers, given that m_1 customers of class 1 are in service, the system can be viewed as

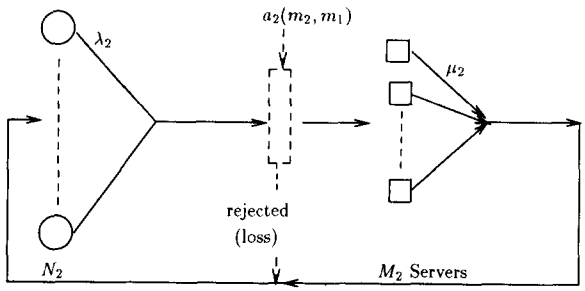


Fig. 4. Equivalent loss system for class 2 given in class 1.

a loss system with $M_2(m_1) = \lfloor (B - b_1 m_1)/b_2 \rfloor$ servers (see Fig. 4). The activation function $a_2(m_2, m_1)$ is used to represent the interference of class 3 customers, and is formally defined as

$$a_2(m_2, m_1) = \sum_{m_3 : B - m_1 b_1 - m_2 b_2 - m_3 b_3 \geq b_2} p(m_3 \mid m_2, m_1) \tag{13}$$

Again, this is simply the conditional probability that class 3 requests leave enough resource for at least one additional request of class 2. When the activation function is known, the solution of this system yields the conditional probability $p(m_2 \mid m_1)$. The solution is analogous to (12).

Now consider our system with m_1 and m_2 customers of class 1 and class 2 in service, respectively. The service resource left is $B - m_1 b_1 - m_2 b_2$. Given m_1 and m_2 customers in service, for class 3 users, the system can be viewed as the simple loss model represented in Fig. 5. Note that there are $M_3(m_2, m_1) = \lfloor (B - m_1 b_1 - m_2 b_2)/b_3 \rfloor$ servers, and, since class 3 is the last in our analysis, there is no need for an activation function. Indeed, all interference of other classes is already accounted for in the reduced amount of the service resource. The solution of this simple Engset model (cf. [3]) produces the conditional probability $p(m_3 \mid m_2, m_1)$, as well as the functions $\hat{a}_1(m_2, m_1)$ and $a_2(m_2, m_1)$.

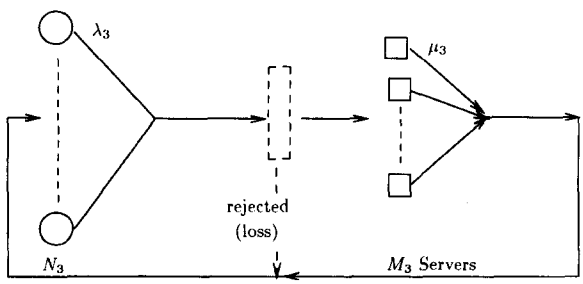


Fig. 5. Equivalent loss system for class 3 given classes 1 and 2.

Using the product-form of the joint probability distribution (8), it is easy to show that the approach based on equivalence and decomposition generates the exact solution for the generalized Engset loss system under consideration.

We outline in Appendix B the computation flow for our method using as an example the 3 class system discussed above.

As mentioned earlier in this section, in many cases, our generalized loss model can be mapped onto the multi-service Engset system studied in [20]. As a matter of fact, Iversen proposes in his paper a computational approach to the evaluation of the product-form solution of such a system through the use of multiple convolutions, based on the distribution of the number of busy servers. Because we use a different state description (the number of requests in service and not the number of busy servers), our approach tends to be considerably more efficient than that of [20]. The precise savings depend on system parameters. As an example, consider the case of a total resource $B = 18$ shared by three classes with $b_1 = 3$, $b_2 = 4.2$, and $b_3 = 6$. This can be mapped onto a full availability system with 30 servers, and individual class requests for 5, 7, and 10 servers, respectively. A direct comparison with the complexity numbers as provided by Iversen, makes our approach an order of magnitude faster. Even if we modify the algorithm in [20] so as to avoid unnecessary computations, our method is still significantly more efficient.

In the case when every class of requests needs the same amount of the service resource, our model becomes clearly equivalent to the multiclass Engset loss system considered in the preceding section. It is interesting to note that, in such a case, significant additional computational savings can be achieved using our approach. Indeed, we notice that then all conditional quantities, such as the activation functions or the conditional average numbers of requests of a given class in service, depend only on the total number of servers busy with requests from "outer" classes. E.g., the activation function $a(m_2, m_1)$ becomes the same for classes 2 and 1, and is a function only of the sum $m_1 + m_2$. Hence, a highly efficient computation can be organized by computing (and storing)

a single activation function as well as individual class averages only once for each decomposition level. For instance, in a system with four classes, the underlying probability identity is

$$p(m_1, m_2, m_3, m_4) = p(m_4 \mid m_3, m_2, m_1)p(m_3 \mid m_2, m_1) \times p(m_2 \mid m_1)p(m_1) \quad (14)$$

First, we consider $p(m_4 \mid m_3, m_2, m_1)$, which is in fact a function of the total $m_3 + m_2 + m_1$. Thus, we compute and store the values of the activation function (conditional probability that at least one server is available) $a(\cdot)$, and the conditional average $\overline{m}_4(\cdot)$ for all values of the total number of requests of "outer" classes (3, 2, or 1) in service. Then we use the activation function to solve the loss model for $p(m_3 \mid m_2, m_1)$. Again, this is a function of the total number $m_2 + m_1$ and not individual values of m_2, m_1. We compute and store the values of the activation function for the next level and the conditional averages $\overline{m}_4(\cdot)$ and $\overline{m}_3(\cdot)$. Note that the values from the preceding analysis level are no longer needed at this point, and can be overwritten. This results in very modest storage requirements of our method. Continuing our procedure, we use the latest activation function to obtain $p(m_2 \mid m_1)$. At this level, we compute the new activation function $a(m_1)$, as well as the conditional averages $\overline{m}_4(m_1)$, $\overline{m}_3(m_1)$, and $\overline{m}_2(m_1)$. Finally, at the last level, we analyze $p(m_1)$, and obtain the unconditional averages $\overline{m}_4$, $\overline{m}_3$, $\overline{m}_2$, and $\overline{m}_1$.

It is clear that the computational complexity of our approach in this case is proportional to the product CM of the number of classes times the total number of servers. By contrast, the convolution method of [20] exhibits then a computational complexity of the order of CM^2. Additionally, the storage requirements of the convolution method, not discussed in [20], are likely to be considerable.

It is interesting to note that our approach based on state equivalence can be readily applied to the limited availability model considered in [20]. Indeed, it suffices to view the total resource B as the number of available trunks and the individual bandwidth requirements b_i as the number of trunks needed for a request of class i. The call intensity for class i, $\lambda_i(m_i)$, replaces our arrival rate $(N_i - m_i)\lambda_i$. Also, the expression for the success probability for class i becomes

$$s_i = \overline{m}_i / \bar{\lambda}_i \quad (15)$$

where $\bar{\lambda}_i = \sum_i p(m_i)\lambda_i(m_i)$, so that we have to include the $\bar{\lambda}_i$s in the quantities evaluated. The limited availability appears simply as an additional limit on the number of requests at each decomposition level.

In the next section, as an example, we apply our generalized loss model to the performance analysis of a modern I/O controller, and, in particular, we look at the effects of the sharing of the bus bandwidth by transfers of different speeds.

4. Application to bus bandwidth sharing

As mentioned in the introduction, multiclass loss models can be used to represent the contention for transfer paths experienced by disk devices in many Input/Output commercial computer organizations. Some modern disk control units [1] utilize an internal bus to handle data transfers for a relatively large number of devices (e.g., 128 disks). Such a bus possesses a given finite bandwidth corresponding to the maximum aggregate data rate that it can sustain. For transfers synchronous with the mechanical disk device, when the device is correctly positioned for data transfer, there must be enough bus bandwidth left to accommodate the device data rate, or else the device will be unable to successfully reconnect to the bus, i.e., will miss a revolution, and will have to reattempt bandwidth acquisition after a full rotation. It is the performance evaluation of such systems that motivated our interest in the generalized Engset loss model of the preceding section.

Note that in a regular disk I/O operation, the service of a request by the disk device involves several phases. First, the disk movable arm may have to be positioned on the correct cylinder. This phase is the disk seek time. Then, when the seek is completed, the disk will typically find itself in a random angular position with respect to the target sector marking the beginning of the data on the disk surface. Thus, the second phase in disk service time is the rotational latency until the device reaches the target sector. Next, the

disk must reconnect to the transfer path. This may involve some number of unsuccessful attempts, resulting in a missed revolution delay. Finally, after successful reconnection the device will be able to transfer the data records requested in the I/O operation. Thus, the disk service time (exclusive of queueing for the device availability) may be seen as composed of seek time, rotational latency, missed revolution delay, and data transfer time. In reality, there are some additional overhead times necessary to process the operation. For simplicity these are thought of as included in the data transfer time in this analysis.

An interesting question arising in connection with such systems is whether devices with faster data rate always outperform slower devices, and, since it is possible to mix several device types, whether such a mixing is always a good idea from a performance standpoint. Note that for a given transfer length (i.e., number of bytes transferred), a higher data rate will result in a shorter transfer time, but possibly in more contention for the bus bandwidth, and thus in a longer bandwidth acquisition delay. The issue is whether the net result is a shorter total service time.

To provide elements of answer to these questions, we consider an idealized system with two classes of devices and restricted bandwidth. We set the number of devices in each group to 16, and we start by assuming that the devices differ only in their data rates. We keep the data rate for the first group (b_1) at 3 Mbytes/s, and we vary the transfer rate for the second group (b_2) between 3 Mbytes/s and 9 Mbytes/s. The total bandwidth is taken to be only 12 Mbytes/s.

We take the seek time and the rotational latency as given (the latter is simply set to one half the device revolution time), and we use the generalized loss model of the preceding section to evaluate the probability that a disk device is able to successfully reconnect to the controller bus. Having obtained this probability for each group of disks, it is easy to assess the expected number of missed revolutions per Input/Output operation, i.e. the missed reconnection delay (cf. [6]). The expected transfer time is computed by considering the number of data bytes transferred and the disk transfer speed. Figures 6, 7, and 8 illustrate the I/O service times obtained for a bus bandwidth system with data transfers of 4 Kbytes, 8 Kbytes, and 12 Kbytes, respectively.

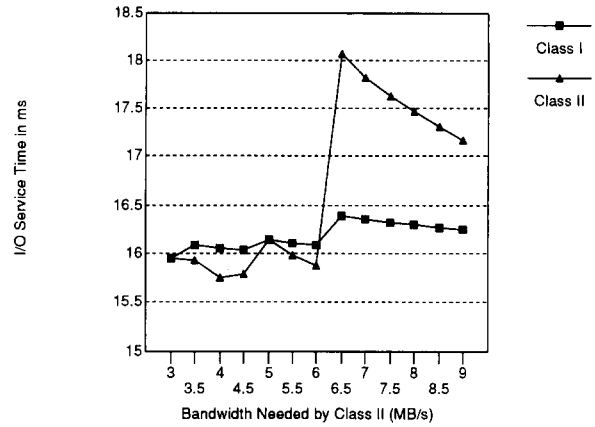


Fig. 6. Effect of bandwidth on I/O service times: 4 Kbyte block transfers, total bandwidth 12 MB/s, $b_1 = 3$ MB/s.

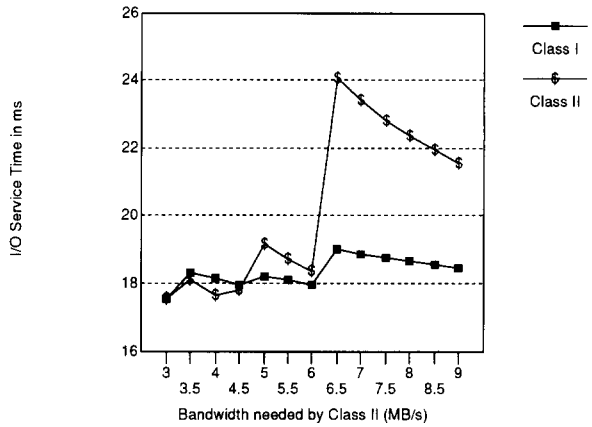


Fig. 7. Effect of bandwidth on I/O service times: 8 Kbyte block transfers, total bandwidth 12 MB/s, $b_1 = 3$ MB/s.

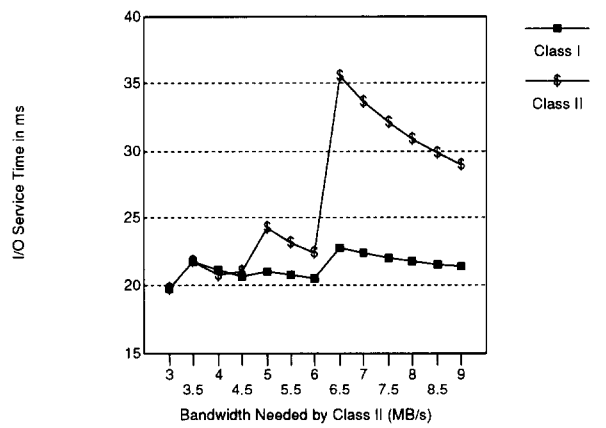


Fig. 8. Effect of bandwidth on I/O service times: 12 Kbyte block transfers, total bandwidth 12 MB/s, $b_1 = 3$ MB/s.

These results correspond to an average seek time of 6.25 ms, disk revolution time of 16.67 ms, and an I/O rate of 200 I/Os per second to each device class. We note that, with the limited bandwidth assumed in this example, especially for the longer transfers of 8 Kbytes and 12 Kbytes, the service time for devices of group 2 increases quite abruptly with the data rate for this group (b_2). Not surprisingly, the changes are non-uniform, in the sense that once a given lower degree of concurrency has been attained, it is best to proceed at the highest data rate that still corresponds to the same degree of concurrency of operation. What is rather surprising is the fact that the best performance for both groups of devices can be achieved when both groups proceed at the lowest data rate of 3 Mbytes/s.

This is apparent from the results for 8 Kbytes and 12 Kbytes transfers. For the shorter 4 Kbytes transfers, the performance for devices of class 2 initially shows a substantial improvement without much impact on class 1 service time. However, as the data rate b_2 continues to increase, we observe a sudden degradation in the performance of both groups of devices.

It is worthwhile noting that discrete-event simulation results fully confirm the findings of our analysis. Table 2 compares the probability of successfully acquiring the bandwidth on a first attempt estimated in the simulation with values computed using the solution approach of Section 3. We observe that the generalized loss model predicts quite accurately the success probability. The obtained results strongly suggest that, when total available resource is limited, unequal individual requirements tend to interfere with each other. The extent of interference can be such that it more than offsets any gains in the time the resource is used. In Fig. 9, we provide a further example of this phenomenon. Here, we again consider two types of devices. Type 1 has a data rate of 3 Mbytes/s and a revolution time of 16.67 ms. Type 2 has not only a faster data rate of 4.2 Mbytes/s, but also a shorter revolution time of 14.1 ms. The total available bus bandwidth is again assumed to be 12 Mbytes/s. The results in Fig. 9 show the service time (exclusive of queueing) for a mixed environment where we have two groups of 16 devices of each type present simultaneously, as well as the two cases where all 32 devices are of a single type (1 or 2). The I/O rates are 200 I/Os per second for each group of 16 devices, and the average seek times are taken to be 6.25 ms for disks in each group. We observe that it is possible for the mixed environment to exhibit poorer performance than either of the two types alone.

Thus, mixing device types is not always a good idea. It should be stressed that these results pertain to a rather limited total available resource. As the bus bandwidth increases, differences in performance, as well as interference between requests of different classes, tend to dissipate.

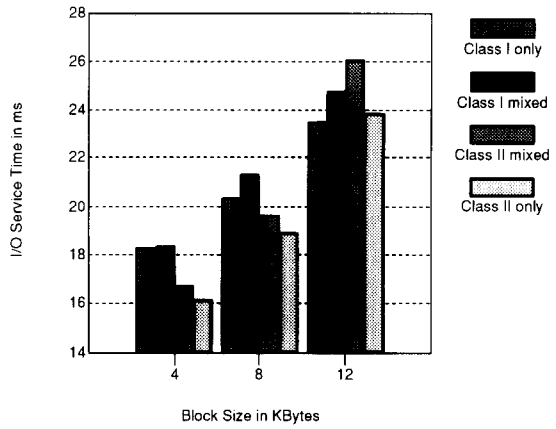


Fig. 9. Performance with mixed versus single disk types: comparison of I/O service time, mixed versus single disk types.

Table 2
Comparison of disk simulation and loss model results

Bandwidth of Class II	Class I success probability		Class II success probability	
	Simulation	Analytical	Simulation	Analytical
3.0	0.982±0.005	0.982	0.982±0.005	0.982
3.5	0.939±0.009	0.940	0.925±0.012	0.931
4.0	0.946±0.004	0.950	0.938±0.010	0.940
4.5	0.957±0.009	0.959	0.917±0.007	0.922
5.0	0.945±0.006	0.946	0.849±0.011	0.849
6.5	0.901±0.013	0.905	0.664±0.022	0.669
7.0	0.910±0.018	0.912	0.687±0.018	0.685
7.5	0.916±0.018	0.918	0.693±0.029	0.700
8.0	0.921±0.006	0.923	0.709±0.014	0.712
8.5	0.927±0.010	0.928	0.721±0.010	0.723
9.0	0.933±0.011	0.932	0.735±0.010	0.733

5. Conclusion

We have considered two related aspects of Engset loss models with multiple classes of customers. For systems with "classical" multiple servers, we have presented a method for efficiently computing the average numbers of customers in service as well as the probability distribution of the number of busy servers. Our method is based on recurrences that exist between conditional average numbers of customers in service given the total number of busy servers. As a related point, it is interesting to note that the recurrence derived for multiserver loss systems applies also to classical finite source queueing systems with product-form solutions [4].

Motivated by modern I/O architectures, as an extension of such models we have considered "generalized" multiclass loss systems, in which the total available service resource gets partitioned into a variable number of servers according to individual class requirements. The recurrences derived for classical models do not appear to hold for the generalized systems. Therefore, we have presented a different approach to the evaluation of these systems. Our approach relies on a repeated use of state equivalence and decomposition.

A comparison with the existing convolution method indicates that the proposed approach is significantly more efficient in terms of both computational complexity and computer memory requirements. Our equivalence and decomposition approach can also be applied to the "classical" Engset multiclass multiserver model considered in Section 2. Its computational complexity then is of the same order as that of the recurrence method, so that is may represent a valuable alternative, especially when quantities other than the average numbers in service or the call congestion are desired. Indeed, any quantity computable from the joint distribution of the numbers of requests of each class in service can be represented in terms of the conditional probabilities used, and thus evaluated from our approach.

As a practical application, we have looked at bus based systems simultaneously supporting two classes of disks with different data rates, i.e., bandwidth requirements. We have found that, when the available bandwidth is limited, such a mixture may not be a good idea from a performance standpoint. In particular, it is possible to have a situation where the contention for bus bandwidth more than offsets the shorter data transfer times afforded by a faster data rate.

An intriguing point relates to the application of the Engset loss system as a model of disk reconnection. In the physical disk system, there is a clear distinction between first reconnection attempts which arrive "at random" after the completion of the seek and rotational latency, and subsequent attempts which have a constant request generation time of one revolution. In our model, we ignore this distinction in the request generation times. Experimental evidence indicates that the results produced by our model tend to match quite closely simulation figures for the success probabilities on the first reconnection attempt for a new request. Subsequent attempts may see typically a somewhat lower success probability. While several "corrections" have been proposed to account for this difference [6], in our opinion, no truly satisfactory approach exists. The practical consequence of using the loss model is that the overall success probability tends to be an underestimation of the missed reconnection delay component of the I/O service time.

Acknowledgment

We wish to thank anonymous referees for their thorough review of an earlier version of this paper and, in particular, for pointing out several omissions in our reference list. Their constructive criticism has been most helpful in preparing this revision.

Appendix A

Balance equations

Recall the multiclass multiserver system of Fig. 1. We obtain the following global balance equations.

$$p(m_1, m_2, \ldots, m_C)\left[\sum_i (N_i - m_i)\lambda_i + \sum_j m_j\mu_j\right] = \sum_i p(\ldots, m_i + 1, \ldots)(m_i + 1)\mu_i + \sum_j p(\ldots, m_j - 1, \ldots)(N_j - m_j + 1)\lambda_j \tag{16}$$

with $m = \Sigma_k m_k < M$ and $m_i < N_i$

The solution of the above system is given by

$$p(m_1,\ldots,m_C) = \frac{1}{G}\prod_{i_1=1}^{m_1}(N_1-i_1+1)\rho_1/i_1\prod_{i_2=1}^{m_2}(N_2-i_2+1) \times\rho_2/i_2\cdots\prod_{i_C=1}^{m_C}(N_C-i_C+1)\rho_C/i_C \quad (17)$$

where G is a normalizing constant corresponding to the sum of the product terms over all feasible states.

The product-form solution given by (16) corresponds to the following local balance equations:

$$p(\ldots, m_j, \ldots)m_j\mu_j = p(\ldots, m_j-1,\ldots)(N_j-m_j+1) \quad (18)$$

$$p(\ldots, m_i, \ldots)(N_i-m_i) = p(\ldots, m_i+1,\ldots)(m_i+1)\mu_i \quad (19)$$

Appendix B

Computation flow for algorithm of Section 3

In practice the actual computation in our algorithm starts with the "innermost" class, i.e., class 3 in our three class example. In order to avoid recomputing the same expressions, we precompute and store in memory several arrays of values. The first one contains the product factors of the form $\prod_{i=1}^{m_3}(N_3-i+1)\rho_3/i$ corresponding to the terms of the solution for $p(m_3\mid m_2, m_1)$ for values of $m_3 = 0, 1,\ldots, \lfloor B/b_3\rfloor$. We denote by $f_3(m_3)$ the elements of this array. In the second array, we keep the running cumulative sum of the corresponding entries in the first array. Thus, the elements of the second array provide precomputed values for the normalizing constants in the solution for $p(m_3\mid m_2, m_1)$, and we denote them by $s_3(m_3)$. We have $s_3(m_3) = \sum_{i=0}^{m_3} f_3(i)$. For class 2, we precompute and store only a quantity analogous to $f_3(m_3)$, denoted by $f_2(m_2)$, where

$$f_2(m_2) = \prod_{i=1}^{m_2}(N_2-i+1)\rho_2/i.$$

We need also auxiliary memory locations to store the current factor in the solution for $p(m_2\mid m_1)$ and the running sum of such factors used to normalize the solution. We denote these storage locations by h_2 and H_2, respectively. Similar memory locations for $p(m_1)$ are denoted by h_1, and H_1; h_1 is initialized to 1, H_1 to 0. Also h_2 is reset to 1 and H_2 to 0 for each new value of m_1. An additional memory location, denoted by v_2, is used to hold the values of the activation function $a_2(m_2, m_1)$ during the computation.

After this initialization phase, we consider our system for values of $m_1 = 0,\ldots, M_1$, and $m_2 = 0,\ldots, M_2(m_1)$, i.e., we enumerate the numbers of requests of the "outer" classes in lexicographical order. For a given value of m_1, and each value of $m_2 = 0,\ldots, M_2(m_1)$, we obtain $\hat{a}_1(m_2, m_1)$, $a_2(m_2, m_1)$ using (11) and (13), and the conditional average number of class 3 customers in service $\overline{m}_3(m_2, m_1)$ as

$$\overline{m}_3(m_2, m_1) = \sum m_3 p(m_3\mid m_2, m_1).$$

Note that the precomputed $f_3(m_3)$ can be used instead of $p(m_3\mid m_2, m_1)$ in these expressions, and the values thus obtained for $\bar{a}_1(m_2, m_1)$, $a_2(m_2, m_1)$, and $\overline{m}_3(m_2, m_1)$ can be normalized simply at the end of the summation by dividing by $s_3(M_3(m_2, m_1))$.

At this point, we use the factor h_2 to compute one term in the sums representing $\overline{m}_3(m_1)$, $a_1(m_1)$ as well as $\overline{m}_2(m_1)$. The latter is, of course, just the product m_2h_2. We also update the cumulative sum H_2. Then, except for the last $m_2 = M_2(m_1)$, we update the product of activation factors in v_2 by multiplying by $a_2(m_2, m_1)$, then we compute a new value for the factor h_2 as $f_2(m_2+1)v_2$. When the last value of m_2 for the given m_1 (i.e. $M_2(m_1)$) has been considered, we normalize the accumulated sums $\overline{m}_3(m_1)$, $\overline{m}_2(m_1)$, and $a_1(m_1)$ dividing them by H_2. We are also ready to update the overall averages $\overline{m}_3$, $\overline{m}_2$, as well as $\overline{m}_1$ using the factor h_1. The cumulative sum H_1 is also updated by simply adding h_1. Except for the last value of m_1 (i.e., M_1), we compute a new value for the factor h_1 using (12). Note that, in practice, this means that we multiply the preceding value by the factor $(N_1-m_1)a_1(m_1)\rho_1/(m_1+1)$.

When the last value for m_1 has been considered, it simply remains to normalize the quantities computed by dividing them by H_1.

References

[1] IBM 3990 Storage Control Reference, IBM Publication, GA32-009-3, 1989.

[2] Amdahl 6100 Storage Processor Announcement, Amdahl Corporation, May 1988.

[3] O.A. Allen, *Probability, Statistics, and Queueing Theory* (Academic Press, Orlando, FL, 1978).

[4] F. Baskett, K.M. Chandy, R.R. Muntz and F.G. Palacios, Open, closed and mixed networks of queues with different classes of customers, *J. ACM* **22**(2) (1975) 248–260.

[5] J.R. Birge and S.M. Pollock, Using parallel iteration for approximate analysis of a multiple server queueing system, *Oper. Res.* **37**(5) (1989) 769–779.

[6] A. Brandwajn, Models of DASD subsystems with multiple access paths: a throughput-driven approach, *IEEE Trans. Comput.* **32**(5) (1983) 451–463.

[7] A. Brandwajn, Equivalence and decomposition in queueing systems—a unified approach, *Performance Evaluation* **5** (1985) 175–186.

[8] J.P. Buzen, Computational algorithms for closed queueing networks with exponential servers, *Comm. ACM* **16**(9) (1973) 527–531.

[9] K.M. Chandy and C.H. Sauer, Computational algorithms for product form queueing networks, *Comm. ACM* **23** (1980) 573–583.

[10] J.W. Cohen, The generalized Engset formulae, *Philips Telecomm.* **18** (1957) 150–170.

[11] A.E. Conway and N.D. Georganas, RECAL—a new efficient algorithm for the exact analysis of multiple chain queueing networks, *J. ACM* **33** (1986) 768–791.

[12] L.E.N. Delbrouck, The uses of Kosten's systems in the provisioning of alternate trunk groups carrying heterogeneous traffic, *IEEE Trans. Comm.* **31**(6) (1983) 741–749.

[13] G. Dobson and U.S. Karmarka, Simultaneous resource scheduling to minimize weighted flow times, *Oper. Res.* **37**(4) (1989) 592–600.

[14] D. Fakinos, The M/G/k group-arrival group-departure loss system, *J. Appl. Probab.* **19** (1982) 826–834.

[15] D. Fakinos and S. Sirakoulis, Product-form distributions for an M/G/k group-arrival group-departure system, *Adv. Appl. Probab.* **21** (1989) 721–724.

[16] B.W. Gnedenko and D. Koenig, *Handbuch der Bedieningstheorie II* (Akademie, Berlin, 1984).

[17] L. Green, A queueing system in which customers require random numbers of servers, *Oper. Res.* **28** (1980) 1335–1345.

[18] P.J. Hunt, Implied costs in loss networks, *Adv. Appl. Probab.* **21** (1989) 661–680.

[19] V.B. Iversen, The A-formula, *Teleteknik* (2) (1980) 64–79.

[20] V.B. Iversen, The exact evaluation of multi-service loss systems with access control, *Teleteknik* (2) (1987) 56–61.

[21] D. Mitra, Asymptotic analysis and computational methods for a class of simple circuit-switched networks with blocking, *Adv. Appl. Probab.* **19** (1987) 219–239.

[22] M. Reiser and S.S. Lavenberg, Mean value analysis of closed multi-chain queueing networks, *J. ACM* **27**(2) (1980) 313–322.

[23] C.H. Sauer and K. Mani Chandi, *Computer Systems Performance Modeling* (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[24] A.F. Seila, On waiting times in a queue in which customers require simultaneous service from a random number of servers, *Oper. Res.* **32** (1984) 1181–1185.

[25] K. Stecke and J.J. Solberg, The optimality of unbalancing both workloads and machine group sizes in closed queueing networks of multiserver queues, *Oper. Res.* **33** (1985) 882–910.

[26] R. Syski, *Introduction to Congestion Theory in Telephone Systems* (North-Holland, Amsterdam, 1985).