Approximate Analysis of a Multi-class Queue with Class-Dependent Window-Flow Control*

Raif O. Onvural

BNR
Network Intelligence Division
RTP, NC 27709-3478

Center for Communications and Signal Processing
North Carolina State University

CCSP-TR-88/24

August 1988

Supported in part by a grant from AIRMICS through the Center for Communications and Signal Processing and by the National Science Foundation under grant no: CCR-87-02258

APPROXIMATE ANALYSIS OF A MULTI-CLASS QUEUE WITH CLASS DEPENDENT WINDOW-FLOW CONTROL*

by

Raif O. Onvural

BNR
Network Intelligence Division
RTP, NC 27709-3478

and

Center for Communications and Signal Processing
North Carolina State University
Raleigh, NC 27695-8206

Abstract

A single server queue shared by N different classes of jobs is considered. Class i jobs arrive at the queue in a poisson fashion with a class dependent rate. The total number of class i jobs allowed to wait in the queue is w_i, the window size of this class. A class i job that arrives to find w_i class i jobs in the shared queue is forced to wait in an input queue. There is one input queue for each class. All classes receive the same service at the shared queue, which is exponentially distributed. An approximate decomposition algorithm is developed which permits to study any number of classes utilizing a numerical procedure for a particular two-class queue. Validation tests show that the approximation procedure is fairly accurate. The applicability of the algorithm to networks of queues is also discussed.

* Supported in part by a grant from AIRMICS through the Center for Communications and Signal Processing, NC State University, and by the National Science Foundation under grant no: CCR-87-02258

1. Introduction

One of the resource management tasks which critically affects the performance of computer communication networks is the flow control, i.e. the regulation of accepted traffic such that the network is not overloaded. Unless proper flow control is enacted, throughput of the network may significantly degrade below its optimum value. The principle of flow control is to shift congestion from the interior of the network to the points of admittance. The most widely known protocol to control the flow in a network is the window-flow control. The idea is simply to refuse admittance of new traffic when a certain number of messages in a region of the network is unacknowledged. The permissible upper bound of unacknowledged messages is called the window size (cf. Reiser [7]).

In this paper, we consider the case where multiple connections use the same resource, depicted by a single server queue. The traffic along each connection is regulated with a window-flow mechanism. In general, most of the studies reported in the literature on window-flow control deal with a single connection (see Fdida, Perros and Wilk [1] for a literature review). Reiser [8] modeled a computer communication network consisting of many virtual routes with end-to-end window flow control as a closed multi-chain queueing network. Each chain represented a different virtual route under the assumption of a loss system. That is, it was assumed that packets that arrive to find a full window are discarded. A computationally efficient approximation procedure was developed based on the mean value analysis for evaluating large multi-chain closed networks. The model is used to

obtain transit delays and blocking probabilities. Lam [3] extended the class of multi-chain queueing networks of the product form type to include mechanisms of state dependent lost and triggered arrivals.

The problem of analyzing multiple window control mechanisms can be formulated as a single or multi-class queueing networks with population constraints. The population consraint is imposed as follows: In the single class case, it is assumed that only upto a predefined number of customers are allowed in a particular subnetwork. The remaining customers are queued in an input queue. In the multi-class case, for a particular subnetwork, each class (or a group of classes) has its own population constraint in a subnetwork. These models have been analyzed approximately as closed queueing networks by several authors. A review of these approximations can be found in Thomasian and Bay [9]. The reader is also referred to Krzeinski and Teunissen [2] for a more recent approximation. We note that these models were originally developed for multiprogramming systems, and they may not be suitable for modelling window-flow control mechanisms, which naturally give rise to open queueing networks. It is necessary, therefore, to develop solution techniques for open multi-class queueing networks with population constraints.

In this paper, we study a queueing model consisting of a single server multi-class queue as shown in Figure 1 (hereafter referred to as the shared queue). The total number of class i jobs allowed to wait in this queue may not exceed a finite number, w_i (hereafter referred to as the window size), i=1,...,N, where N is the number of classes. A class i job that arrives to the network at a time that there are w_i jobs in the shared

queue is forced to wait in an input queue. There is one input queue per class. We assume that the input queues have infinite capacities. A class i job waiting in the input queue is allowed to enter the shared queue only when a class i job departs from the network.

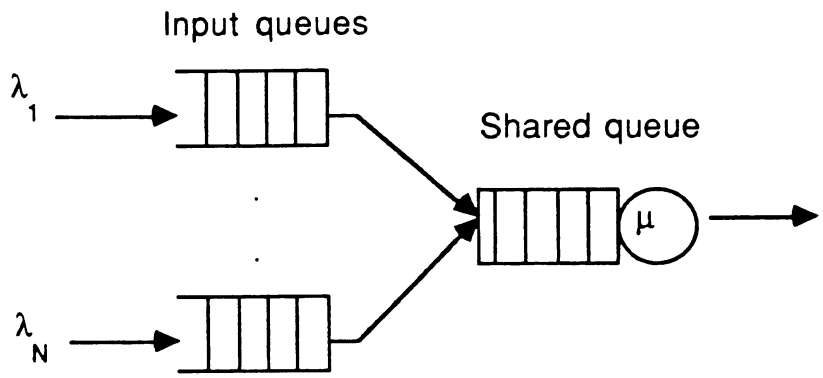

Figure 1: A Multi-class Open Queueing Model

Let λ_i, $1/\mu$ be respectively the rate of arrivals of class i jobs and the mean service time of a job in the shared queue. All interarrival times and the service times are assumed to be exponentially distributed. Jobs at each input queue move into the shared queue in a FIFO manner. The service discipline at the shared queue is assumed to be processor sharing. We note that the marginal probabilities of classes are the same whether the service discipline at the shared queue is processor sharing, service in random order, or FIFO. This can be easily verified by comparing the steady state equations for each discipline (cf. Onvural, Perros and Koerner [6]).

The following theorem states that if we do not distinguish between classes, then this multi-class queue behaves as a single class M/M/1 queue (cf. [6]).

Theorem 1: Consider an N-class queueing model as illustrated in figure 1. Let $\pi(n)$ be the probability that the total number of jobs in the multi-class queue is n. If the input queues have infinite capacities for all classes, then the system behaves like a single class M/M/1 queue with an arrival rate being equal to the sum of arrival rates of all classes in the original queue.

When we distinguish between classes, exact analysis of this simple model becomes difficult. In particular, this system does not possess a product form solution. A numerical study of the model involves the following four step procedure: i) determining the finite capacities of the input queues, C_i, such that $\sum_{j=C_{i+1}}^{\infty}\pi_{i(j)}$ is negligible for all i, where $\pi_i(j)$ is the marginal probability of having j class i jobs in the system; ii) Generation of states; iii) Creating the rate matrix, Q, and iv) Numerical solution of the system $Q^TP(\underline{n})=0$. An interested reader may refer to [6] for details. In particular, let $K_j=C_j+w_j$. Then the number of states of this single queue is equal to $\prod_{j=1}^{N}\{K_j+1\}$. Hence, the numerical approach becomes infeasible as the number of classes increases.

In this paper, an approximation algorithm is developed to study this multi-class open queue. The procedure decomposes the queue by class which permits the algorithm to accommodate any number of classes. In particular, to obtain the marginal queue length distribution of class k, we aggregate all classes other than class k into one class (hereafter referred to as the aggregated class). Then this two-class queue is studied numerically with revised window-sizes and revised arrival rates. We

note that the two-class model developed here has a matrix-geometric form and it can be analyzed rather easily. The approximation procedure is presented next in section 2. The applicability of the algorithm to network of queues is discussed in section 3. Finally, the conclusions are given in section 4.

2. The Approximation Procedure

In this section, we present a decomposition procedure for the approximate analysis of N-class open queueing model illustrated in figure 1. Without loss of generality, consider class k, and aggregate all classes other than class k into one class. The window-size and the arrival rate of class k jobs in this two-class model is the same as in the original system, i.e. w_k and λ_k. In the original system, the number of aggregated class jobs in the shared queue varies between 0 and the sum of the window capacities of all classes other than class k. Therefore, we have $w_A = \sum_{j \neq k} w_j$, where w_A denote the window size of the aggregated class. The accuracy of the algorithm depends on the values of the arrival rate, λ_A, and the capacity of the input queue, C_A. We note that the arrival rate of the aggregated class is equal to the sum of the arrival rates of those classes that forms the aggregated class as long as the number of aggregated class jobs in the shared queue is less than the minimum window capacity of all classes that forms the class. Once this limit is reached, in the real system, jobs belonging to the class with the smallest window size may not enter the shared queue. Similarly, there are other points in $[0,w_A]$ that customers of other classes may be forced to wait in their respective input queues once the number of aggregated jobs in the

shared queue reaches certain limits. In view of this, the arrival rate of the aggregated class depends on the number of aggregated class jobs in the shared queue.

The approximation algorithm presented next is based on the two assumptions that $C_A=1$, and λ_A is independent of the number of aggregated class jobs in the shared queue. Then, the value of λ_A should be such that the effective arrival rate of aggregated class jobs (the rate at which such jobs are accepted to the multi-class queue) is equal to $\sum_{j\in k}\lambda_j$. That is

(cons. 1) $$\lambda_A=\sum_{j\in k}\lambda_j \;/\{1-P_A(w_A+1)\}$$

where $P_A(w_A+1)$ is the marginal probability of having w_A+1 (which is the total capacity of the aggregated class) aggregated class jobs in the two class queue. In view of this constraint, λ_A is obtained using the following fixed point problem which can be easily shown to be convergent.

Algorithm 1:

Step 0: Let $\lambda_A=\sum_{i\in k}\lambda_i$, $w_A=\sum_{i\in k}w_i$, $C_A=1$. Furthermore, let λ_k and w_k be the same as in the original system.

Step 1: Solve this two-class queue numerically to obtain $P_A(w_A+1)$.

Step 2: Calculate $\gamma_A=\sum_{j\in k}\lambda_j/\{1-P_A(w_A+1)\}$. If $|\gamma_A-\lambda_A| < \varepsilon$ then STOP else set $\lambda_A=\gamma_A$ and go to step 1.

Algorithm 1 produces the joint probabilities, $P(n_k,n_A)$, of having n_k and n_A class k and aggregated class jobs in the queue respectively, where

$0 \le n_k \le \infty$ and $0 \le n_A \le w_A+1$. Let $P_k(n_k)$ be the marginal probability of having n_k class k jobs in the queue. Then, by definition, $P_k(n_k) = \sum_{n_A=0}^{w_A+1} P(n_k, n_A)$.

Algorithm 1 was observed to produce fairly accurate results. However, its performance can be further improved by taking the following constraint into consideration.

Let L be the mean number of all jobs in the multi-class queue. Then, $L=\rho/(1-\rho)$ where $\rho=\sum_{i=1}^{N}\lambda_i/\mu$, as the queue, in this case, behaves as an M/M/1 queue (cf. Theorem 1). Furthermore, let L_k denote the mean queue length of class k jobs in the queue, i.e. $L_k = \sum_{n_k=0}^{\infty} n_k P_k(n_k)$, k=1,...,N. Then, the following equality holds:

(**cons. 2**) $$\sum_{k=1}^{N} L_k = L$$

If constraint 2 is not met after obtaining the marginal queue length distributions of each class using Algorithm 1, then, the difference $L-\sum_{k=1}^{N} L_k$ is distributed to each queue using L_k as the respective weights. However, once the mean queue length of each class is modified, the marginal probabilities needs to be updated accordingly to produce the new L_k. The following algorithm summarizes this procedure.

Algorithm 2:

Step 0:Let $\rho=\sum_{i=1}^{N}\lambda_i/\mu$ and calculate $L=\rho/(1-\rho)$. Furthermore, let $sum=\sum_{k=1}^{N}L_k$,

$dif=L-sum$; $norm=\sum_{k=1}^{N}(sum-L_k)$;

Step 1: for k=1 to N do

$$L(k)=dif*(sum-L_k)/norm$$

$$P_k(n_k)=P_k(n_k)+P_k(n_k)L(k)/L_k, \qquad n_k>0$$

$$P_k(0)=1-\sum_{n_k=1}^{\infty}P_k(n_k)$$

The complete procedure of obtaining the marginal queue length probabilities of all classes can be summarized as follows:

Algorithm 3:

Step1: For each class k do
Obtain the marginal queue length probabilities of class k jobs using Algorithm 1.

Step 2: Apply Algorithm 2 to re-normalize the marginal queue length probabilities.

We note that the two-class model of Algorithm 1 can be solved numerically by going through the four steps presented in section 1 to solve multi-class open queueing networks under consideration. However, seeing that this is the most time consuming part of the algorithm, we will present a more efficient way of solving the above two-class model. In particular, let (n_k,n_A) denote the state of this queue, where n_k and n_A are the number of class k and aggregated class jobs, respectively. Then, we have $0\le n_k\le\infty$ and $0\le n_A\le w_A+1$. The resulting continuous time irreducible

Markov Process has an infitesimal generator Q of the following form after the states are ordered lexicographically, where the submatrices are defined as follows.

$A_{01}(i,j) = \lambda_A$, if $i=j$, $i=0,\ldots,w_A+1$

$A_{00}(i,j) = -(\lambda_k+\lambda_A)$, if $i,j=0$

$= -(\lambda_k+\lambda_A+\mu)$, if $i=j$; $i=1,\ldots,w_A$

$= -(\lambda_k+\mu)$, if $i=j=w_A+1$

$= \lambda_A$, $j=i+1$; $i=0,\ldots,w_A$

$= \mu$, $j=i-1$; $i=1,\ldots,w_A+1$

Let $k^*=\min(k,w_k)$; $j^*=\min(j,w_A)$. For $1\le k\le w_k$:

$A_{kk}(i,j) = -(\lambda_k+\lambda_A+\mu)$, if $i=j$; $i=0,\ldots,w_A$

$= -(\lambda_k+\mu)$, if $i=j=w_A+1$

$= \lambda_A$, if $j=i+1$; $i=0,\ldots,w_A$

$= \{j^*/(k^*+j^*)\}\mu$, $j=i-1$; $i=1,\ldots,w_A+1$

$A_{k;k-1}(i,j) = \{k^*/(k^*+j^*)\}\mu$, $j=i$; $i=0,\ldots,w_A+1$

$$
\begin{array}{cccccccc}
A_{00} & A_{01} & & & & & & \\
A_{10} & A_{11} & A_{01} & & & & & \\
 & A_{21} & A_{22} & A_{01} & & & & \\
 & & * & * & * & & & \\
 & & & A_{w_kw_{k-1}} & A_{w_kw_k} & A_{01} & & \\
 & & & & * \ * & * & & \\
 & & & & & A_{w_kw_{k-1}} & A_{w_kw_k} & A_{01} \\
 & & & & & & * \qquad * & \qquad *
\end{array}
$$

We note that, all matrices are of order w_A+2. The stationary probability vector x, if it exists, is the solution to the system $xQ=0$ and $xe=1$, where

the column vector e has all elements equal to 1. Let the vector x be partitioned as $x=(x_0,x_1,x_2,...)$ where x_i is a w_A+2 vector. Then we have:

(i) $x_0A_{00}+x_1A_{10}=0$

(ii) $x_{i-1}A_{01}+x_iA_{ii}+x_{i+1}A_{i+1;i}=0, \quad i=1,...,w_k-1$

(iii) $x_{i-1}A_{01}+x_iA_{w_kw_k}+x_{i+1}A_{w_kw_{k-1}}=0, \quad i=w_k, w_k+1,.....$

(iv) $\sum_{i=0}^{\infty}x_ie = 1$

This system of equations can be solved relatively easily using Neuts' procedure (cf. Neuts[5]). In particular, let $A=A_{01}+A_{w_kw_k}+A_{w_kw_{k-1}}$, and π be such that $\pi A=0$, $\pi e=1$. If $\pi A_{w_kw_{k-1}}e>\pi A_{01}e$ (which, in this case, is equivalent to $\sum_{k=1}^{N}\lambda_k<\mu$), then there exists a non-negative matrix R such that:

$$x_i=x_{w_{k-1}}R^{i-w_{k-1}}, \qquad i\geq w_k$$

Assuming for a moment that R is known, the above system of equations can be written as follows:

(i) $x_0A_{00}+x_1A_{10}=0$

(ii) $x_{i-1}A_{01}+x_iA_{ii}+x_{i+1}A_{i+1;i}=0, \quad i=1,...,w_{k-2}$

(iii) $x_{w_{k-2}}A_{01}+x_{w_{k-1}}(A_{w_kw_k}+RA_{w_kw_{k-1}})=0,$

(iv) $x_i=x_{w_k}R^{i-w_k}, \quad i\geq w_k$

(v) $\{\sum_{i=0}^{w_{k-2}}x_i\}e+x_{w_{k-1}}(I-R)^{-1}e=1$

The matrix R is the unique non-negative solution of the matrix quadratic equation $A_{01}+RA_{w_kw_k}+R^2A_{w_kw_{k-1}}=0$. It is obtained as the limit

of the monotonically increasing sequence of matrices $\{R_n, n>0\}$; $R_0=0$, and $R_{n+1}=A_{01}(-A_{w_kw_k})^{-1}+R_n^2A_{w_kw_{k-1}}(A_{w_kw_k})^{-1}$.

Once, the matrix R is obtained, the unknowns $x_0,\ldots,x_{w_k}$ are obtained solving the system of equations (i), (ii), and (iii) with the normalizing equation (v). The resulting matrix is of order $\{w_k(w_A+2)\}$.

The algorithm is applied to a variety of multi-class queues with different number of classes, window capacities, arrival rates and service rates. A representative set of examples were given in Tables 2-8. Exact values for the validation purposes are obtained numerically for the three class model and using a simulation program (written in QNAP) for four and five classes. Relative error percentages are calculated as 100*(exact value-approximate value)/exact value.

3. Network of Queues

In this section, we discuss the applicability of the algorithm developed in section 2 to network of queues in which each queue is a multi-class queue with class dependent window-flow control as studied above.

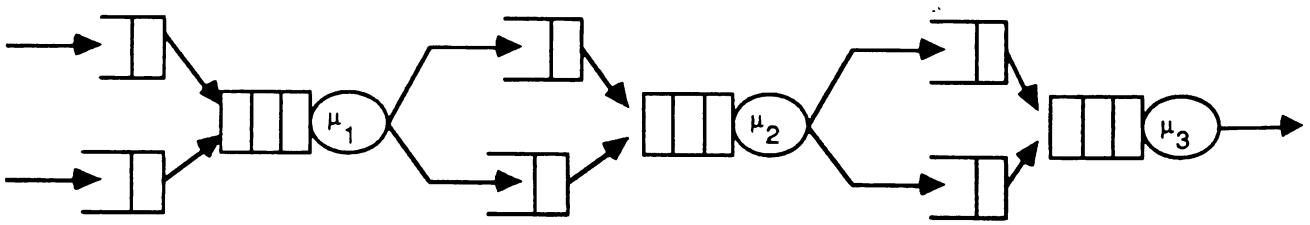

Figure 2: A Network of Queues in Series

There are N classes of jobs arriving at queue 1 in a poisson fashion with a class dependent rate λ_k. Each class k has its own window size at queue i, w_{ik}, i=1,...,K; k=1,...,N, where K is the number of queues in the network. Each queue in the network is of the type considered in this paper, as illustrated in Figure 1. A job, completing its service at queue i, joins queue j with probability p_{ij}, which is independent of the class of the job. The state of this network is a vector of vectors $(S_1,...,S_K)$, where $S_i=(s_{i1},...,s_{iN})$. s_{ik} denotes the number of class k jobs at queue i. As each queue in isolation do not have a product form solution, we should expect that the network of queues do not have a product form solution as well.

Let us now consider the network after aggregating all classes into a single class. The arrival rate of this single class is the sum of the arrival rates of all classes, i.e.$\lambda_A = \sum_{k=1}^{N}\lambda_k$. The state of the network is $(n_1,...,n_K)$, where n_i is the total number of jobs (all classes) at node i. In this case, from theorem 1, the network reduces to K independent M/M/1 queues. Let P(.) be the joint steady state queue length distribution of the total number of jobs at each queue and $P_i(n_i)$ be the marginal probability of having n_i jobs at queue i, which can be obtained by analyzing the queue in isolation. Then:

$$P(n_1,...,n_K)=P_1(n_1)P_2(n_2)....P_K(n_K)$$

where $P_i(n_i)=(1-\rho)\rho^{n_i}$, $\rho=\sum_{i=1}^{N}\lambda_i/\mu_i$. Hence, the performance measures of the network can be easily calculated, if we do not distinguish between classes. The product form breakdowns when we distinguish between

classes. In particular, the departure process of each class is not exponential, hence, the arrival distribution of class k jobs at queue i, $i \neq 1$, is not poisson. Considering the fact that the state space of this network is very large, it may not be possible to solve it numerically, and simulating the network is expected to be very time consuming.

As an approximate solution technique, we decompose the network into individual queues and analyze them in isolation. By doing so, only the arrival processes need to be characterized, as the other parameters of the queue in isolation, i.e. window-sizes, service rate, should be the same as in the original network. Let us first assume that arrival of class k jobs to each queue occurs in a poisson fashion. The total arrival stream of class k jobs into queue i is formed by class k departures from other queues that are routed to queue i. Hence, the arrival rate of class k jobs into queue i, γ_{ik}, is the solution of the traffic equations:

$$\gamma_{ik}=\lambda_i+\sum_{j=1}^{K}\gamma_{jk}p_{ji}, \quad i=1,\ldots,K$$

With these parameters, the algorithm developed in section 2 can be used to solve these individual queues in isolation approximately to obtain the marginal queue length probabilities of all classes at each queue. We note that the only assumption to analyze the network of queues was made on the arrival distribution of classes at each queue, i.e. class dependent poisson arrivals.

4. Conclusions

We presented a model for the case where multiple connections use the same resource, depicted by a single server queue. As the numerical approach becomes infeasible even with a few classes, it is necessary to develop approximation algorithms for the analysis of such queues. An approximation algorithm is developed based on decomposing the queue into N-two-class queues where N is the number of classes. These two-class models are then solved numerically using Neuts' matrix geometric procedure. Validation tests show that the algorithm is very accurate. The applicability of the algorithm to the network of queues is immediate after decomposing the network by node. In this case, the only assumption required is poisson arrivals to each queue.

In real situations, more than one node may be required to represent a route. We are currently working on this problem as an extension of the ideas presented in this paper.

REFERENCES

[1] S. Fdida, H.G. Perros, and A. Wilks, "Semaphore Queues: Modelling multilayered Window Flow Control Mechanisms", Tech. Rep. Computer Science, 1988, North Carolina State University

[2] A. Krzeinski and P. Teunissen, "Multiclass Queueing Networks with Population Constrained Subnetworks", Proc. ACM Sigmetrics Conf. on Measurement and Modelling Computer Systems, Austin, TX, 1985, 128-139

[3] S.S. Lam, "Queueing Networks with Population Size Constraints", IBM J. Res. Dev., July 1977, 370-388

[4] E.D. Lazowska and J. Zahorjan, "Multiple Class Memory Constrained Queueing Networks", Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Seattle WA, 1982, 130-140

[5] M.F. Neuts, Matrix Geometric Solutions in Stochastic Models-An algorithmic approach, The John Hopkins University Press, Baltimore, 1981

[6] R. Onvural, H.G. Perros, and U. Koerner, "On a Multi-Class Queue with Class Dependent Window-Flow Control", 4th Int. Conf. on Modelling Tech. and Tools for Comp. Perf. Evaluation

[7] M. Reiser, "A Queueing Network Analysis of Computer Communications Network with Window Flow Control", IEEE Trans. Comm., vol. COM 27, 1199-1209, 1979

[8] M. Reiser, "Performance Evaluation of Data Communications Systems", Proc. IEEE, vol. 70, 1982, 171-196

[9] A. Thomasian and P. Bay, "Analysis of Queueing Network Models with Population Constraints and Delayed Blocked Customers", Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Cambridge, 1984, 202-216

Table 1: Data for the examples in tables 2 to 8

Example No	arrival rates	service rate	window capacities
1	{1,1,3}	9	{2,3,2}
2	{2,3.2,2.4,1.5}	11	{2,3,2,3}
3	{2,1.2,1.5,3}	12	{4,3,2,4}
4	{2,3.2,2.4,1.5}	15	{2,3,2,3}
5	{1,3,2,4,2}	16	{2,4,3,3,2}
6	{1,2.5,2,2,3}	14	{2,3,2,3,2}
7	{2,3.2,2.4,1.5,3}	18	{4,3.4,3.2}

Table 2: Example 1 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.8118	0.8082	0.45
P(1)	0.1493	0.156	-4.29
P(2)	0.0299	0.0284	5.28
P(3)	0.0068	0.0057	19.30
P(4)	0.0016	0.0012	33.33
P(5)	0.0004	0.0003	33.33
Mean Queue L.	0.2387	0.2367	0.84

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.8113	0.809	0.28
P(1)	0.1499	0.1563	-4.09
P(2)	0.0308	0.0286	7.69
P(3)	0.0063	0.0051	23.53
P(4)	0.0014	0.0009	55.56
P(5)	0.0003	0.0002	50.00
Mean Queue L.	0.23755	0.23315	1.89

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.5744	0.5705	0.68
P(1)	0.2402	0.2427	-1.03
P(2)	0.1009	0.1018	-0.88
P(3)	0.0446	0.0449	-0.67
P(4)	0.0206	0.0207	-0.48
P(5)	0.0098	0.0098	0.00
Mean Queue L.	0.77374	0.78015	-0.82

Table 3: Example 2 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.4921	0.478	2.95
P(1)	0.2566	0.25	2.64
P(2)	0.1154	0.128	-9.84
P(3)	0.0591	0.064	-7.66
P(4)	0.0324	0.036	-10.00
P(5)	0.0185	0.019	-2.63
Mean Queue L.	1.078	1.085	-0.65

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.3801	0.37	2.73
P(1)	0.2459	0.238	3.32
P(2)	0.1453	0.148	-1.82
P(3)	0.0827	0.091	-9.12
P(4)	0.0505	0.057	-11.40
P(5)	0.0321	0.036	-10.83
Mean Queue L.	1.654	1.7009	-2.76

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.4427	0.407	8.77
P(1)	0.2486	0.232	7.16
P(2)	0.1231	0.134	-8.13
P(3)	0.0691	0.076	-9.08
P(4)	0.0417	0.047	-11.28
P(5)	0.0262	0.031	-15.48
Mean Queue L.	1.3824	1.4334	-3.56

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.578	0.583	-0.86
P(1)	0.2645	0.261	1.34
P(2)	0.1003	0.101	-0.69
P(3)	0.0359	0.037	-2.97
P(4)	0.0133	0.012	10.83
P(5)	0.0049	0.004	22.50
Mean Queue L.	0.6703	0.646	3.76

Table 4: Example 3 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.6864	0.685	0.20
P(1)	0.2145	0.216	-0.69
P(2)	0.0677	0.069	-1.88
P(3)	0.0213	0.021	1.43
P(4)	0.0066	0.007	-5.71
P(5)	0.0022	0.002	10.00
Mean Queue L.	0.4592	0.455	0.92

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.7863	0.784	0.29
P(1)	0.1672	0.17	-1.65
P(2)	0.0362	0.036	0.56
P(3)	0.0078	0.008	-2.50
P(4)	0.0019	0.0017	11.76
P(5)	0.0005	0.0003	66.67
Mean Queue L.	0.27384	0.2743	-0.17

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.7437	0.735	1.18
P(1)	0.1876	0.192	-2.29
P(2)	0.0473	0.053	-10.75
P(3)	0.0139	0.015	-7.33
P(4)	0.0046	0.004	15.00
P(5)	0.0017	0.001	70.00
Mean Queue L.	0.35763	0.364	-1.75

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.5913	0.589	0.39
P(1)	0.2407	0.242	-0.54
P(2)	0.0986	0.099	-0.40
P(3)	0.0402	0.041	-1.95
P(4)	0.0162	0.018	-10.00
P(5)	0.0069	0.007	-1.43
Mean Queue L.	0.69999	0.695	0.72

Table 5: Example 4 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.7522	0.746	0.83
P(1)	0.1832	0.187	-2.03
P(2)	0.0454	0.049	-7.35
P(3)	0.0129	0.013	-0.77
P(4)	0.0041	0.004	2.50
P(5)	0.0014	0.001	40.00
Mean Queue L.	0.34074	0.345	-1.23

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.653	0.652	0.15
P(1)	0.2247	0.226	-0.58
P(2)	0.0783	0.078	0.38
P(3)	0.027	0.027	0.00
P(4)	0.0101	0.01	1.00
P(5)	0.004	0.004	0.00
Mean Queue L.	0.5421	0.538	0.76

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.7151	0.711	0.58
P(1)	0.2	0.202	-0.99
P(2)	0.0564	0.06	-6.00
P(3)	0.0181	0.018	0.56
P(4)	0.0064	0.006	6.67
P(5)	0.0024	0.002	20.00
Mean Queue L.	0.41499	0.416	-0.24

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.8052	0.801	0.52
P(1)	0.1554	0.16	-2.88
P(2)	0.0314	0.031	1.29
P(3)	0.0063	0.006	5.00
P(4)	0.0014	0.001	40.00
P(5)	0.0003	0.001	-70.00
Mean Queue L.	0.24455	0.249	-1.79

Table 6: Example 5 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.8176	0.804	1.69
P(1)	0.1456	0.16	-9.00
P(2)	0.0279	0.03	-7.00
P(3)	0.0065	0.005	30.00
P(4)	0.0017	0.001	70.00
P(5)	0.0005	0.001	-50.00
Mean Queue L.	0.23095	0.243	-4.96

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.5857	0.574	2.04
P(1)	0.2403	0.248	-3.10
P(2)	0.1011	0.105	-3.71
P(3)	0.0419	0.044	-4.77
P(4)	0.017	0.017	0.00
P(5)	0.0074	0.007	5.71
Mean Queue L.	0.71773	0.723	-0.73

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.6831	0.673	1.50
P(1)	0.2135	0.223	-4.26
P(2)	0.069	0.071	-2.82
P(3)	0.0218	0.022	-0.91
P(4)	0.0077	0.007	10.00
P(5)	0.0029	0.002	45.00
Mean Queue L.	0.47496	0.475	-0.01

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.506	0.496	2.02
P(1)	0.2446	0.25	-2.16
P(2)	0.1196	0.123	-2.76
P(3)	0.0571	0.067	-14.78
P(4)	0.0299	0.031	-3.55
P(5)	0.0167	0.017	-1.76
Mean Queue L.	1.0593	1.071	-1.09

Class No=5	Approximate Values	Exact Values	Relative error %
P(0)	0.6773	0.665	1.85
P(1)	0.2109	0.221	-4.57
P(2)	0.0667	0.075	-11.07
P(3)	0.0251	0.026	-3.46
P(4)	0.0106	0.009	17.78
P(5)	0.0048	0.003	60.00
Mean Queue L.	0.51711	0.508	1.79

Table 7: Example 6 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.8059	0.789	2.14
P(1)	0.1513	0.167	-9.40
P(2)	0.0318	0.034	-6.47
P(3)	0.0079	0.007	12.86
P(4)	0.0022	0.002	10.00
P(5)	0.0006	0.001	-40.00
Mean Queue L.	0.25177	0.266	-5.35

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.6046	0.596	1.44
P(1)	0.2339	0.246	-4.92
P(2)	0.0943	0.097	-2.78
P(3)	0.037	0.037	0.00
P(4)	0.0159	0.015	6.00
P(5)	0.0073	0.005	46.00
Mean Queue L.	0.6818	0.673	1.31

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.6542	0.645	1.43
P(1)	0.2166	0.227	-4.58
P(2)	0.0741	0.081	-8.52
P(3)	0.0297	0.029	2.41
P(4)	0.0132	0.011	20.00
P(5)	0.0062	0.004	55.00
Mean Queue L.	0.57963	0.565	2.59

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.6624	0.654	1.28
P(1)	0.219	0.228	-3.95
P(2)	0.0764	0.079	-3.29
P(3)	0.0259	0.027	-4.07
P(4)	0.0097	0.008	21.25
P(5)	0.0038	0.003	26.67
Mean Queue L.	0.52498	0.52	0.96

Class No=5	Approximate Values	Exact Values	Relative error %
P(0)	0.5462	0.534	2.28
P(1)	0.2349	0.24	-2.13
P(2)	0.1016	0.112	-9.29
P(3)	0.0501	0.054	-7.22
P(4)	0.0271	0.027	0.37
P(5)	0.0155	0.014	10.71
Mean Queue L.	0.96184	0.966	-0.43

Table 8: Example 7 in Table 1

Class No=1	Approximate Values	Exact Values	Relative error %
P(0)	0.7554	0.751	0.59
P(1)	0.183	0.188	-2.66
P(2)	0.046	0.046	0.00
P(3)	0.0115	0.011	4.55
P(4)	0.0029	0.003	-3.33
P(5)	0.0008	0.001	-20.00
Mean Queue L.	0.32706	0.329	-0.59

Class No=2	Approximate Values	Exact Values	Relative error %
P(0)	0.6538	0.648	0.90
P(1)	0.2235	0.228	-1.97
P(2)	0.0779	0.08	-2.63
P(3)	0.0267	0.03	-11.00
P(4)	0.0102	0.009	13.33
P(5)	0.0042	0.003	40.00
Mean Queue L.	0.54597	0.542	0.73

Class No=3	Approximate Values	Exact Values	Relative error %
P(0)	0.7187	0.714	0.66
P(1)	0.2003	0.207	-3.24
P(2)	0.0577	0.058	-0.52
P(3)	0.0165	0.016	3.13
P(4)	0.0047	0.005	-6.00
P(5)	0.0014	0.001	40.00
Mean Queue L.	0.39535	0.396	-0.16

Class No=4	Approximate Values	Exact Values	Relative error %
P(0)	0.8064	0.801	0.67
P(1)	0.1544	0.16	-3.50
P(2)	0.0311	0.031	0.32
P(3)	0.0062	0.007	-11.43
P(4)	0.0014	0.001	40.00
P(5)	0.0004	0.001	-60.00
Mean Queue L.	0.24328	0.247	-1.51

Class No=5	Approximate Values	Exact Values	Relative error %
P(0)	0.6661	0.658	1.23
P(1)	0.2162	0.222	-2.61
P(2)	0.0704	0.075	-6.13
P(3)	0.0264	0.028	-5.71
P(4)	0.0111	0.01	11.00
P(5)	0.005	0.004	25.00
Mean Queue L.	0.53921	0.535	0.79