Approximate Analysis of a Multi-Class Open Queueing Network with Class Blocking and Push-Out

Tülin Atmaca

Harry G. Perros

Yves Dallery

Center for Communications and Signal Processing
Department Electrical and Computer Engineering
North Carolina State University

TR-91/2
January 1991

Approximate Analysis of a Multi-Class Open Queueing Network with Class Blocking and Push-out[1]

by

Tülin Atmaca
Laboratoire MASI,
Université Pierre et Marie Curie
4 Place Jussieu
75252 Paris Cedex 05, France

Harry G. Perros[2]
Computer Science Department, and
Center for Communication and Signal Processing
North Carolina State University
Raleigh, NC 27695-8206, USA

Yves Dallery
Laboratoire MASI,
Université Pierre et Marie Curie
4 Place Jussieu
75252 Paris Cedex 05, France

January 1991

[1]Supported in part by NATO under grant no CRG 900580.
[2]Supported in part by the National Science Foundation under grant no CCR-87-02258.

Abstract:

We study a multi-class queueing network which consists of a finite capacity node (node 0) linked to M parallel finite capacity nodes (nodes 1 to M). M classes of customers are assumed. All customers first join node 0. A class i customer after completion of its service at node 0 always joins the ith node. All service times and inter-arrival times are assumed to be exponentially distributed. The service priority at node 0 is head-of-line with pre-emption. When node i (i=1,2,...,M) is full, node 0 cannot process class i customers. In addition to the service priorirty at node 0, push-out is employed. That is, a customer that arrives at node 0 when the node is full, takes the space of a customer which has the lowest priority among the customers already in the node. If all customers in the node have a higher or equal priority, then the arriving customer is lost. This queueing network is analyzed approximately by decomposing it into individual nodes, and then analyzing each node separately. Node 0 is analyzed using a class by class decomposition. The approximation algorithm has been validated using simulation, and the approximate results have a good error.

1. Introduction

Queueing networks with blocking have recently received a lot of attention (see [1], and [8]). Blocking arises as a consequence of the finiteness of the buffers. In many real systems, and especially in computer networks and in manufacturing systems, the finiteness of buffers has a significant effect on the performance. Thus, it is important to be able to analyze queueing network models with finite capacity queues.

So far, most of the work on queueing networks with blocking has been devoted to single-class queueing networks. Exact solutions of queueing networks with blocking are in general not obtainable. As a result, much effort has been devoted to obtaining approximate solutions. Several approximation methods have been proposed for the analysis of open tandem queueing networks with finite buffers. These approximation algorithms are based on the notion of decomposing the original system into a set of smaller subsystems. These methods are efficient and usually provide a fairly accurate estimation of the performance measures, such as throughput and queue-length distributions. Some of these methods have been extended to handle open queueing networks with a general topology. A survey of these approximation algorithms is given in [6]. Approximation methods for analyzing closed queueing networks with finite buffers have also been proposed. For a survey of relevant results see [5].

Despite the numerous contributions in the area of single-class queueing networks with blocking, very few papers have addressed the problem of analyzing open multi-class queueing networks with blocking (see [5]). One of the main difficulties encountered when analyzing approximately such networks, is the analysis of a single node. Due to multiple classes and blocking, this is not a simple task as it was in the case of single-class open queueing networks with blocking.

In this paper, we consider an open multi-class queueing network consisting of a finite capacity node (node 0) linked to M parallel finite capacity nodes (nodes 1 to M). M classes of customers are assumed. This queueing network is analyzed approximately using single-node decomposition. Of interest in this paper, is the analysis of node 0 for which we have assumed M classes of customers, head-of-line with pre-emption service priority, blocking, and a mechanism for managing the space in node 0 known as *push-out*. The computational complexity of analyzing node 0 numerically increases rapidly as the number of classes increases. In order to avoid this

problem, we propose a class aggregation technique which avoids the complexities of the numerical technique.

The paper is organized as follows. The multi-class queueing network with blocking is described in section 2. In section 3, we describe the approximation method for analyzing this queueing network, and in section 4, we present the class aggregation technique. Comparisons between approximate and simulation results are given in section 5. Finally, the conclusions are given in section 6.

2. The multi-class queueing network under study

The multi-class queueing network under study is a tree-like configuration consisting of a node linked to M parallel nodes as shown in figure 1. We shall refer to the first node as node 0, and the M parallel nodes are numbered from 1 to M. Each node is represented by a single server queue with a finite capacity b_i (i=0,1,...,M) including the space in front of the server. There are M classes of customers. Class i customers arrive at node 0 in a Poisson fashion at a rate λ_i and they receive an exponentially distributed service with parameter μ_{0i} (i=1,...,M). Each class of customers has a unique destination node. That is, upon service completion at node 0, class i customers always join node i (i=1,...,M). In view of this, node 0 is shared by all classes, while nodes 1 to M is used only by a single class. The service times at node i (i=1,...,M) are exponentially distributed with parameter μ_i.

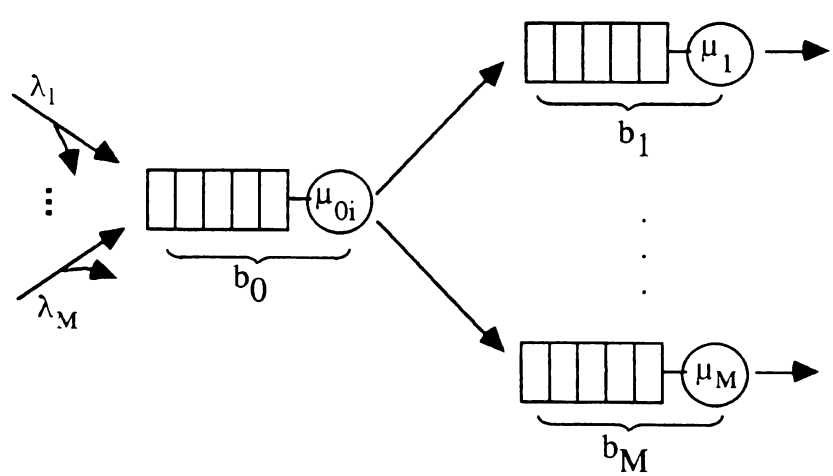

Figure 1: The multi-class queueing network under study

We assume that class i (i=1,2,....,M-1) has a higher priority than class i+1. The service discipline at node 0 is head-of-line with pre-emption. Note that since the service times are exponentially distributed and because of the memoryless property of the exponential distribution, it is not important to specify whether the discipline is pre-emptive resume or pre-emptive repeat.

Due to the fact that the destination nodes 1 to M are finite, the flow of customers through node 0 may be blocked. Blocking-before-service is assumed (see [6]). That is, a customer cannot begin his service unless there is space in the destination node. If node i becomes full upon service completion at node 0, the whole class i of customers becomes blocked. However, the server at node 0 is not blocked, and it can proceed to serve customers from lower classes that may be present at node 0. The server will become blocked if all classes present at node 0 are blocked. Class i becomes unblocked at the moment when a departure takes place from node i (i=1,2,...,M). Due to service pre-emption, a class i customer can start service immediately, if the server is busy serving a lower class customer. We shall refer to this type of blocking as *class blocking*. We note that in view of this type of blocking, it is possible to have a lower class customer in service while higher class customers are present at node 0 despite the service pre-emption discipline. This happens when all the higher class customers are blocked.

Blocking-before-service has been used extensively in single-class queueing networks with finite capacity queues. The difference between this blocking mechanism and the class blocking mechanism described above, is that in the former case the server gets blocked each time a customer becomes blocked. Intuitively speaking, one can easily see that under the class blocking mechanism, the utilization of the server at node 0 is higher than in the case of blocking before service, and therefore the system's throughput is also higher.

The overall capacity of node 0, b_0, is shared by all classes of customers. Unlike other finite capacity nodes, a customer that arrives to node 0 when the node is full, is not always lost. In particular, let us assume that a class i customer arrives when node 0 is full. Then, the customer is lost if the node at that moment only contains customers of class i or higher. However, if the node contains lower class customers, then a customer of the lowest class currently present at node 0 is forced out of the node so that the arriving customer may enter the node. As an example, let us consider the case where we have four classes (i.e. M = 4). Let us assume that node 0 is full and that a class 2 customer arrives. The customer will get lost if node 0 at that moment only contains customers belonging to class 1 or 2. The customer will not get lost, however, if there is a class 3

or class 4 customer present. In particular, the arriving customer will first attempt to force out a class 4 customer. If there is no class 4 customer present, then he will force out a class 3 customer. In general, when the arriving class i customer forces a class j customer out of node 0 (where $j > i$), the forced out customer within class j is the one that arrived last. Due to the service pre-emption discipline, this customer will be forced out even if at that moment it so happens that he is receiving service. This mechanism for managing the space in a finite capacity queue is known as *push-out*. This mechanism and other similar mechanisms have been proposed within the context of high-speed computer networks. However, only two classes have been considered (see [3]).

3. The approximation algorithm

In this section, we present an approximation algorithm for analyzing the multi-class queueing network with blocking described above. We analyze this queueing network using a single-node decomposition algorithm. The principle of this algorithm is similar to that proposed in [7] for single-class queueing networks with blocking-after-service. The multiclass queueing network described in section 2 is analyzed by decomposing it to its individual nodes. In particular, each node i is analyzed by approximating it by another node, hereafter referred to as Q_i (i=0,1....,M).

Q_i (i=1,...,M) is a single-class node with the same buffer capacity and service rate as node i but with a revised arrival process. This arrival process approximates the actual arrival process to node i from node 0 in the original network. This arrival process is assumed to be Poisson with rate $\bar{\lambda}_i$. Thus, Q_i is simply an M/M/1/N queue with arrival rate $\bar{\lambda}_i$, service rate μ_i, and buffer capacity b_i. Let $p_i(k_i)$ to be the steady-state probability that there are k_i customers in Q_i (k_i=0,1,....,b_i) and let X_i be the throughput of Q_i. For a given arrival rate $\bar{\lambda}_i$, these quantities can be easily calculated.

Q_0 is a multiclass node that is identical to node 0 except that the blocking effect due to the downstream nodes (nodes 1 to M) is taken into account in an approximate way. The blocking effect is modelled as follows. Upon service completion of a class i customer at Q_0, class i is blocked with a probability r_i and is not blocked with a probability $1-r_i$. If it is blocked, the server is prevented from servicing class i customers. This blocking condition will last for a period of time that is exponentially distributed with rate μ_i. As soon as this period of time ends, the server is again allowed to serve customers of class i. Due to service pre-emption, a class i customer can start service immediately, if the server is busy serving a lower class customer. The behaviour of Q_0 is

otherwise identical to that of node 0, i.e. it has the same service priority and space priority mechanisms. Let $p_0(n_{0i})$ denote the probability that there are n_{0i} class i customers in Q_0. Also, let X_{0i} be the throughput of class i customers at Q_0. Note that X_{0i} is less than λ_i due to the finite capacity of Q_0 and also due to push-out.

The behavior of Q_0 can be described by a continuous-time Markov chain (CTMC). The state of this CTMC is defined by the vector $s = (n_1,j_1,...,n_i,j_i,...,n_M,j_M)$. n_i is the number of class i customers currently present at Q_0 (n_i=0,1,...,b_0) and j_i is a parameter that indicates whether class i is blocked or not. $j_i = 0$ if class i is not blocked and $j_i = 1$ if class i is blocked. Let **Q** be the infinitesimal generator of the CTMC of Q_0. Let **p** be the stationary probability vector of all probabilities of state. Then, **p** can be obtained by solving numerically the system of linear equations $\mathbf{pQ} = \mathbf{0}$ and $\mathbf{pe} = 1$.

In order to analyze Q_0 we need to know the completion instant blocking probabilities r_i (i=1,...,M). Each probability r_i is approximated by the probability that a customer arriving to Q_i will occupy the last position (see [2]). Let π_i be this probability. Then, we have

$$\pi_i = \frac{p_i(b_i-1)}{1 - p_i(b_i)} \ .$$

Also, the analysis of Q_i (i=1,...,M) requires knowledge of the arrival rate $\bar{\lambda}_i$. This arrival rate is determined in such a way that the throughput of Q_i is equal to the throughput of class i customers in Q_0, that is $X_i = X_{0i}$.

The following iterative algorithm can be used to determine the unknown parameters:

Algorithm 1.

Step 0. Initialization.

Set $X_{0i} = \lambda_i$, for all i = 1,2,...,M.

Step 1. Analysis of each single-class queue Q_i.

For i = 1,2,...,M :

Step 1.a. Determine $\bar{\lambda}_i$ such that $X_i = X_{0i}$.

Step 1.b. Calculate the probability π_i.

Step 2. Analysis of the multi-class queue Q_0.

Step 2.a. For i = 1,...., M, set $r_i = \pi_i$, .
Step 2.b. Calculate the steady-state probability vector **p**.
Step 2.c. For i = 1,...., M, derive the throughputs X_{0i}.

Step 3. Go to Step 1 until convergence of the unknown parameters.

Step 4. Calculate all performance measures of interest.

Algorithm 1 is an iterative procedure that analyzes the set of single-class nodes Q_i (i=1,2,...,M) and the multi-class node Q_0 in turn. On the one hand, for given values of the throughputs, Step 1 calculates new values of the blocking probabilities π_i. In Step 1.a, the value of $\bar{\lambda}_i$ is obtained as the solution of the fixed-point iteration: $\bar{\lambda}_i = X_i /(1-p_i(b_i))$ (see [2], and [7] for details). On the other hand, for given values of the blocking probabilities, Step 2 calculates new values of the throughputs X_{0i}. In Step 2.b, the vector **p** is the solution of the system of equations $\mathbf{pQ} = 0$ and $\mathbf{pe} = 1$. In our implementation of this algorithm, we used the power method to solve the above system of equations. Obviously, more efficient methods can be used (see [10]). In Step 3, the iterative procedure is stopped as soon as $(\pi_i^{(k)}-\pi_i^{(k-1)})/\pi_i^{(k)} < \varepsilon$, for all i=1,...,M, where $\pi_i^{(k)}$ is the k-th estimate of π_i. In our implementation of this algorithm, we used $\varepsilon = 10^{-5}$.

In terms of complexity, the main difficulty of algorithm 1 lies in the numerical solution of the CTMC associated with Q_0. The cpu and memory complexity of the solution depends on the size of the state space of the CTMC. The number of states of the CTMC is a function of the buffer capacity b_0 of Q_0 and the number of classes of customers M. Thus, the size of the CTMC is very large when b_0 and M are large. Moreover, the CTMC has to be solved as many times as the number of iterations required for the convergence of algorithm 1. Thus, this numerical solution can only be used when the CTMC associated with Q_0 is of moderate size. In the next section, we present an approximation method for analyzing Q_0 which avoids the complexity issues of the above numerical solution.

4. Class aggregation

As stated above, the size of the CTMC associated with Q_0 can be very large for networks with an arbitrary number of classes. However, for networks with only two classes, the number of states will in general be acceptable. Indeed, in that case, the number of states is equal to $2(b_0+1)(b_0+2)$. For instance, for b_0=10 the number of states is 264. Thus, except for very large values of b_0, the

solution of the CTMC associated with a network consisting of only two classes is obtainable in a reasonable amount of time.

Based on this observation, we propose a *class aggregation* technique which reduces the analysis of Q_0 to analyzing this node M-1 times, but each time with two classes of customers. Due to the head-of-line pre-emptive service priority and the push-out mechanism, the behaviour of class i is not affected by the lower priority classes. Thus, it can be analyzed without considering the lower priority classes. The behavior of class i, however, is affected by the higher priority classes. In view of this, we analyze the behavior of class i by ignoring classes i+1 to M, and representing all higher priority classes i-1 to 1 by a single aggregate class. Let $A_{1,i-1}$ denote this aggregate class.

The class aggregation technique works as follows. We first consider classes 1 and 2. Using algorithm 1, we analyze nodes Q_0, Q_1 and Q_2 assuming that Q_0 consists of only two classes, i.e. classes 1 and 2. Based on the results obtained, we aggregate classes 1 and 2 into the single class $A_{1,2}$. We then use algorithm 1 to analyze nodes Q_0 and Q_3, assuming that Q_0 consists of classes $A_{1,2}$ and 3. Class A_{12} and class 3 are then aggregated into the single class $A_{1,3}$. Using algorithm 1 we analyze class nodes Q_0 and Q_4, assuming that Q_0 consists of classes $A_{1,3}$ and 4. We continue in this fashion until we analyze all classes.

We now proceed to describe how two classes are aggregated into one. We first consider the aggregation of classes 1 and 2. After algorithm 1 has been applied to nodes Q_0, Q_1 and Q_2, the quantities r_1 and r_2 as well as the steady-state probabilities of the CTMC associated with Q_0 are known. Let E be the set of states of the CTMC, where each state is given by the vector (n_1, j_1, n_2, j_2). E is partitioned into several subsets of states. Let $E(n_A, j_A)$ be a subset of E, where $n_A=0,1,...,b_0$, and $j_A=0,1$. The set of states that belong to $E(n_A, j_A)$ is defined as follows:

$$(n_1, j_1, n_2, j_2) \in E(n_A, 0) \quad \text{if} \quad \{n_1+n_2 = n_A\}$$
$$\text{and} \quad \{\{n_1 > 0\} \text{ and } \{j_1 = 0\}\} \text{ or } \{\{n_2 > 0\} \text{ and } \{j_2 = 0\}\}$$

$$(n_1, j_1, n_2, j_2) \in E(n_A, 1) \quad \text{if} \quad \{n_1+n_2 = n_A\}$$
$$\text{and} \quad \{\{n_1 = 0\} \text{ or } \{j_1 = 1\}\} \text{ and } \{\{n_2 = 0\} \text{ or } \{j_2 = 1\}\}.$$

That is, a state belongs to $E(n_A, 0)$ if the total number of customers of classes 1 and 2 is equal to n_A and one class of customers (either class 1 or 2) is currently receiving service. On the other

hand, a state belongs to $E(n_A, 1)$ if the total number of customers of classes 1 and 2 is equal to n_A and neither class 1 nor class 2 is receiving service. This happens when for each class of customers, either there is no customer ($n_i = 0$) or the class is blocked ($j_i = 1$). It is easy to check that the total number of subsets is equal to $2b_0+1$ (subset $E(0, 0)$ is empty).

The reason we partition the states of the CTMC in this way is because the behavior of class 3 in Q_0 with respect to higher priority classes 1 and 2, is only affected by two factors: 1) the total number of class 1 and 2 customers currently present in Q_0, and 2) whether the server is busy working on a class 1 or 2 customer. The former factor is represented by variable n_A and the latter by j_A.

According to this partition, we can define an aggregate CTMC. Each state of this aggregate CTMC is given by the vector (n_A, j_A), and the total number of states is equal to the number of subsets of states of the original CTMC. The transitions rates are obtained using standard Markov chain aggregation. Let us briefly recall how this works. Consider a CTMC partitionned into several subsets of states $\{E_1,...,E_r,...,E_R\}$. Let j be any state of the CTMC and let p(j) be the probability of being in state j. Let $\gamma(j, k)$ be the transition rate between states j and k. Then, the transition rate between states E_r and E_s in the aggregate CTMC, say $\delta(r, s)$, is as follows

$$\delta(r, s) = \frac{\sum_{j \in E_r} \left(p(j) \sum_{k \in E_s} \gamma(j, k) \right)}{\sum_{j \in E_r} p(j)} .$$

The aggregate CTMC describes the behaviour of the aggregate class $A_{1,2}$. We note however that the behaviour of the aggregate CTMC is only an approximation to the behavior of the original CTMC. This is because the transitions between subsets of states in the original CTMC are not Markovian.

Having aggregated classes 1 and 2 to class $A_{1,2}$, we can proceed to analyze class 3. This is achieved by analyzing nodes Q_0 and Q_3 using algorithm 1 and assuming that Q_0 has two classes, namely, aggregate class $A_{1,2}$ and class 3. The unknown parameters are the completion instant blocking probabilities r_3 of class 3 customers at Q_0, and the arrival rate $\bar{\lambda}_3$ at Q_3. For a given value of r_3, Q_0 can again be analyzed by constructing a CTMC describing its behavior in a similar way

as what was done in section 3. The state of this CTMC is defined by the vector $s = (n_A, j_A, n_3, j_3)$. The transition rates of this CTMC are obtained as follows. For any transition related to the aggregate class, i.e. any transition that involves a modification of either n_A or j_A or both, the transition rate is obtained from the aggregate CTMC described above. For any transition related to class 3 customers, the transition rate is simply obtained according to the Markovian behavior of class 3.

The unknown parameters, r_3 and $\bar{\lambda}_3$, can then be determined using a fixed-point procedure similar to that described in algorithm 1. Having analyzed class 3, classes $A_{1,2}$ and 3 can be aggregated into the single class $A_{1,3}$ in a similar way as describe above for classes 1 and 2. The behaviour of class $A_{1,3}$ is thus described by an aggregate CTMC whose transition rates are obtained from the steady-state probabilities of Q_0. Class 4 can be analyzed using algorithm 1 for the nodes Q_0 and Q_4, and assuming that Q_0 has two classes, namely, class $A_{1,3}$ and 4. This procedure is repeated until all classes have been analyzed.

The main steps of this approximation algorithm are summarized below.

Algorithm 2.

Set i = 1.

Step 0. Initialization

Set $X_{0i} = \lambda_i$.

Step 1. Analysis of Q_i.

Step 1.a. Determine $\bar{\lambda}_i$ such that $X_i = X_{0i}$.

Step 1.b. Calculate the probability π_i.

Step 2. Analysis of Q_0 consisting of class $A_{1,i-1}$ and class i.

Step 2.a. Set $r_i = \pi_i$.

Step 2.b. Calculate the steady-state probability vector **p**.

Step 2.c. Derive the throughput X_{0i}.

Step 3. Go to Step 1 until convergence of the unknown parameters.

Step 4. Derive the transition rates of the aggregate class $A_{1,i}$.

Step 5. If i < M, set i = i+1 and go to step 0. Otherwise stop.

5. Validation

The approximation algorithm described above, including the class aggregation approach, was implemented on a vax workstation. Different configurations were analyzed and the approximate results were compared against simulation data. The simulation model was developed using QNAP2 [9]. Six representative comparisons between the approximate and the simulation data are presented in tables 2 to 7 assuming 4 classes of customers. Each table gives the mean queue length for each class in node 0, the global mean queue length for node 0, the mean queue length for nodes 1 to 4, and the throughput of each class (departure rate from queue 1, 2, 3, and 4). The relative error calculated as (simul-approx)/ simul is also given. In general, the accuracy of this type of approximation algorithm depends upon the amount of traffic carried by the network. Therefore, in order to put the results in perspective, each table gives the utilization, as measured by simulation, of the server at node 0 (per class and global), and the utilization of the servers at nodes 1 to 4. The parameters of the six examples are given in table 1.

Example	λ_i, i=1,2,3,4	μ_{0i}, i=1,2,3,4	μ_i, i=1,2,3,4	b_0	b_i, i=1,2,3,4
1	0.6,0.5,0.05,0.05	2,2,2,2	3,3,3,3	5	3,3,3,3
2	0.8,0.6,0.05,0.05	2,2,2,2	3,3,3,3	5	3,3,3,3
3	0.05,0.1,0.8,1	2,2,2,2	2,2,2,2	5	5,3,3,3
4	0.2,0.4,0.8,0.4	1,2,2,3	2,2,4,4	5	3,3,3,3
5	0.1,0.2,0.4,0.2	1,2,2,3	2,2,4,4	5	3,3,3,3
6	0.05,0.1,0.2,0.1	1,2,2,3	2,2,4,4	5	3,3,3,3

Table 1: Parameters of the examples given in tables 2 to 7

Tables 2 and 3 give results where the server at node 0 is utilized considerably more by classes 1 and 2 than by classes 3 and 4. The utilization of the server at node 0 is about 58% in table 2 and about 70% in table 3. Table 4 gives an example where the utilization of classes 3 and 4 is higher than the utilization of classes 1 and 2, and in the example given in table 5 the utilization of class 3 is higher. The server's utilization at node 0 is about 81% in table 4 and 79% in table 5. The examples given in tables 6 and 7 are for two cases where the utilization of the server at node 0 is lower than in the four previous examples, i.e. about 45% and 23% respectively. We note that the approximation algorithm performs well and, in general, the relative errors are quite low.

6. Conclusions

One of the main difficulties encountered when analyzing approximately open queueing networks with blocking, is the analysis of a single node. Due to multiple classes and blocking, this is not a simple task as it was in the case of single-class open queueing networks with blocking. In this paper, we considered an open multi-class queueing network consisting of a finite capacity node (node 0) linked to M parallel finite capacity nodes (nodes 1 to M). Of interest in this paper, was the analysis of node 0 for which we assumed M classes of customers, head-of-line with pre-emption service priority, blocking, and a mechanism for managing the space in node 0 known as *push-out*. This queueing network is analyzed approximately using single-node decomposition. The computational complexity of analyzing node 0 numerically increases rapidly as the number of classes increases. In order to avoid this problem, we proposed a class aggregation technique which avoids the complexities of the numerical technique. The approximation algorithm has been validated using simulation, and the approximate results have a good error.

Mean queue-length at node 0				
	Approx	Simul	Rel. error	Util
Class 1	0.4298	0.4160	-0.033	0.304
Class 2	0.6292	0.6049	-0.040	0.236
Class 3	0.0764	0.0753	-0.015	0.022
Class 4	0.0781	0.0803	0.027	0.022
Global	1.2135	1.1765	-0.031	0.584
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.2450	0.2464	0.006	0.200
Node 2	0.1864	0.1954	0.046	0.159
Node 3	0.0148	0.0149	0.007	0.015
Node 4	0.0144	0.0142	-0.014	0.014
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.5989	0.5982	-0.001	
Class 2	0.4751	0.4696	-0.012	
Class 3	0.0437	0.0425	-0.028	
Class 4	0.0427	0.0422	-0.012	

Table 2: Example 1.

Mean queue-length				
	Approx	Simul	Rel. error	Util
Class 1	0.6611	0.5958	-0.109	0.398
Class 2	0.8900	0.8470	-0.051	0.262
Class 3	0.0826	0.0824	-0.002	0.018
Class 4	0.0829	0.0806	-0.028	0.017
Global	1.7166	1.6058	-0.070	0.695
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.3465	0.3486	0.006	0.265
Node 2	0.2089	0.2244	0.069	0.176
Node 3	0.0123	0.0121	0.017	0.012
Node 4	0.0119	0.0119	0.0	0.012
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.7946	0.7888	-0.008	
Class 2	0.5238	0.5222	-0.003	
Class 3	0.0365	0.0362	-0.028	
Class 4	0.0351	0.0351	-0.023	

Table 3: Example 2.

Mean queue-length				
	Approx	Simul	Rel. error	Util
Class 1	0.0256	0.0258	0.008	0.025
Class 2	0.0555	0.0555	0.0	0.050
Class 3	0.8250	0.8384	0.016	0.394
Class 4	1.6119	1.6480	0.022	0.344
Global	2.5180	2.5677	0.019	0.813
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.0256	0.0264	0.030	0.026
Node 2	0.0526	0.0527	0.002	0.050
Node 3	0.5779	0.5898	0.020	0.391
Node 4	0.4878	0.5344	0.088	0.344
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.0500	0.0502	-0.004	
Class 2	0.1000	0.0992	-0.008	
Class 3	0.7849	0.7799	-0.007	
Class 4	0.6913	0.6817	-0.014	

Table 4: Example 3.

Mean queue-length				
	Approx	Simul	Rel. error	Util
Class 1	0.2500	0.2525	0.010	0.200
Class 2	0.4524	0.4525	0.0	0.196
Class 3	1.2352	1.2263	-0.007	0.323
Class 4	0.5038	0.5022	-0.003	0.075
Global	2.4414	2.4335	-0.003	0.794
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.1108	0.1131	0.020	0.100
Node 2	0.2399	0.2507	0.044	0.196
Node 3	0.1919	0.2064	0.073	0.163
Node 4	0.0600	0.0659	0.090	0.056
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.1999	0.2000	0.001	
Class 2	0.3923	0.3911	-0.003	
Class 3	0.6495	0.6415	-0.012	
Class 4	0.2265	0.2236	-0.013	

Table 5: Example 4.

Mean queue-length				
	Approx	Simul	Rel. error	Util
Class 1	0.1111	0.1071	-0.037	0.097
Class 2	0.1520	0.1481	-0.026	0.098
Class 3	0.4124	0.3939	-0.048	0.195
Class 4	0.2032	0.1972	-0.030	0.063
Global	0.8787	0.8463	-0.038	0.453
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.0526	0.0510	-0.031	0.048
Node 2	0.1107	0.1112	0.004	0.099
Node 3	0.1086	0.1098	0.011	0.097
Node 4	0.0493	0.0516	0.045	0.048
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.1999	0.2000	-0.032	
Class 2	0.3923	0.3911	-0.017	
Class 3	0.6495	0.6415	-0.016	
Class 4	0.2265	0.2236	0.001	

Table 6: Example 5.

Mean queue-length				
	Approx	Simul	Rel. error	Util
Class 1	0.0527	0.0522	-0.009	0.050
Class 2	0.0614	0.0618	0.006	0.050
Class 3	0.1437	0.1437	0.0	0.100
Class 4	0.0617	0.0623	0.010	0.033
Global	0.3195	0.3200	0.002	0.233
Mean queue-length at node i, i=1,2,3,4				
	Approx	Simul	Rel. error	Util
Node 1	0.0256	0.0261	0.019	0.025
Node 2	0.0526	0.0519	-0.013	0.049
Node 3	0.0526	0.0535	0.017	0.050
Node 4	0.0255	0.0256	0.004	0.025
Throughput for class i, i=1,2,3,4				
	Approx	Simul	Rel. error	
Class 1	0.0500	0.0494	-0.012	
Class 2	0.1000	0.0989	-0.011	
Class 3	0.1998	0.1975	-0.012	
Class 4	0.0996	0.0989	-0.007	

Table 7: Example 6.

References

[1] I. Akyildiz and H. G. Perros (Eds.), *Special Issue on Queueing Networks with Finite Buffers, Performance Evaluation,* Vol. 10, pp. 197-210, 1989.

[2] Y. Dallery and Y. Frein, On decomposition methods for tandem queueing networks with blocking, Tech. Rep. MASI No. 293, Université Pierre et Marie Curie, July 1989.

[3] A.A. Nilsson, F. Lai, and H.G. Perros, A queueing model of a bufferless synchronous Clos ATM switch with head-of-line priority and push-out, Tech. Rep. 90-18, Computer Science Dept., North Carolina State University, 1990.

[4] R. O. Onvural, A survey of closed queueing networks with finite buffers, *ACM Computing Surveys*, Vol. 22. No 2, pp. 83-121, 1990.

[5] R. O. Onvural and H. G. Perros, Approximate analysis of multi-class tandem open queueing networks with Coxian parameters and finite buffers, King, Mitrani, Pooley (Eds.) *Performance '90* (North-Holland, 1990) 131-141.

[6] H. G. Perros, Open queueing networks with blocking, *Stochastic Analysis of Computer and Communication Systems,* Takagi (Ed.), North Holland, 1989.

[7] H. G. Perros and T. Altiok, Approximate analysis of open queueing networks with blocking : tandem configurations, *IEEE Transactions on Software Engineering*, Vol SE-12, N°3, pp. 450-461, 1986.

[8] H. G. Perros and T. Altiok (Eds.), *Proc. of the First Int. Workshop on Queueing Networks with Blocking,* North Holland, 1989.

[9] QNAP2, Reference manual, version 4.0, Simulog, Av. du Centre, 78182 St. Quentin en Yvelines, France.

[10] W. J. Stewart, A comparison of numerical techniques in Markov modeling, *Communications of the ACM,* Vol. 21, N°2, pp. 144-152, 1978.