A discrete-time queueing model of the shared buffer ATM switch with bursty arrivals [1]

S. Hong and H.G. Perros

Computer Science Department and
Center for Communication and Signal Processing
North Carolina State University
Raleigh, NC 27695-8206, U.S.A.

and

H. Yamashita[2]

Mechanical Engineering Dept.
Sophia University
Kioi-cho 7-1, Chiyoda-ku
Tokyo 102, Japan

[1]Supported in part by DARPA under grant no. DAEA18-90-C-0039.
[2]Work done while on a sabbatical leave of absence at the Computer Science Dept. of N. C. state Univ.

Abstract

We model the shared buffer ATM switch as a discrete-time queueing system. The arrival process to each port of the ATM switch is assumed to be bursty and it is modelled by an Interrupted Bernoulli Process. The discrete-time queueing system is analyzed approximately. It is first decomposed into subsystems, and then each subsystem is analyzed separately. The results from the subsystems are combined together through an iterative scheme. The analysis of each subsystem involves the contruction of the superposition of all the arrival processes to the switch. Comparisons against simulation data showed that the approximate results have a good accuracy.

Keywords: ATM, shared buffer switch, bursty arrivals, interrupted Bernoulli process, discrete-time queueing, superposition, aggregation, approximations.

1 Introduction

Many ATM switch architectures have been proposed so far. These architectures can be classified into the following three categories: space-division, medium sharing, and buffer sharing (see Tobagi [22]). In this paper, we are concerned with modelling the shared buffer switch architecture. In this type of switch architecture, there is a single memory which is shared by all input and output ports. All incoming and outgoing cells are kept in the same memory. The size of the memory is fixed so that it corresponds to a specific cell loss. An example of this type of switch architecture is the Prelude architecture (see Devault, Cochennec, and Servel [5]). Also see Kuwahara et al [12], and Lee et al [13].

Analytic performance models of the shared buffer switch architecture have been proposed under the assumption of Bernoulli arrivals. These models are based on the analysis of a single Geo/D/1 infinite capacity queue. The results obtained from this queue are used to analyze the whole switch which is represented by N identical Geo/D/1 queues. In particular, Bruneel and Steyaert [4] obtained the N-fold convolution of the queue-length distribution of a Geo/D/1 queue. Using this convolution, they derived the queue-length distribution of N Geo/D/1 queues. Eckberg and Hou [6] proposed a heuristic method which involved the following two steps. First, derive the mean and variance of the number of cells in N Geo/D/1 queues. Then, choose a well known distribution, characterized by its first two moments, which closely approximates the real queue-length distribution in a shared buffer switch. For further details see Petit and Desmet [18]. See also Boyer et al [3], and Petit et al [17].

In this paper, we model the shared buffer switch by a discrete-time queueing system which explicitly represents the input and output ports and the queueing of cells in the shared buffer. The arrival process to each input port is assumed to be an *Interrupted Bernoulli Process* (IBP). We analyze the queueing system approximately by decomposing it into subsystems. Each subsystem is analyzed separately. The results from the subsystems are combined together through the means of an iterative scheme.

The analysis of a subsystem is quite complex due to the large number of arrival processes. One way of analyzing a subsystem is to first obtain the superposition of all the arrival processes, and then analyze it assuming a single arrival process. This approach reduces the dimensionality of the problem. However, it requires the construction of the superposition process which is a fairly complex problem in itself. This problem has received a lot of attention in the open literature. One approach for obtaining the superposition process focuses on approximating it by a renewal process, see Albin [1], Whitt [23], Sriram and Whitt [21], Iscoe, McDonald, and Qian [9], and also Perros and Onvural [16]. Heffes and Lucantoni [8] proposed a method for superposing approximately voice sources using a *Markov Modulated Poisson Process* (MMPP). An alternative method for constructing an MMPP approximation to the superposition of identical bursty sources was proposed by Baiocchi et al [2]. Finally, Heffes [7] obtained an MMPP approximation to the superposition of different MMPP arrival processes using a set of simple expressions.

An alternative way to solving the superpostion problems is to analyze approximately all the arrival processes together with the queue where the arrivals wait. Such a system, typically, represents an ATM multiplexer. Various solutions have been proposed for the analysis of an ATM mul-

tiplexer, see for instance, Norros et al [15], Sengupta [20], and Nagarajan, Kurose, and Towsley [14]. For further reference, see Pujolle and Perros [19]. We note that most of the methodologies reported in the literature for the analysis of an ATM multiplexer have been obtained assuming identical arrival processes. However, this assumption is unrealistic since in a real ATM network it is highly unlikely that all patterns of arriving cells will be identical.

The remaining of the paper is organized as follows. In the following section, we describe in detail the queueing model of the shared buffer switch. In section 3, we present the approximation algorithm, and in section 4, we validate the approximation algorithm by comparing it against simulation data. Finally, the conclusions are given in section 5.

2 Model Description

An ATM switch routes incoming cells from input ports to their destination output ports. The mechanism to route incoming cells to their requested output ports can be implemented in various ways. In the shared buffer switch architecture, all incoming cells are stored in a shared buffer and their memory addresses are stored in address queues. There is one address queue per output port. All memory addresses in an address queue point to cells in the shared buffer which are destined for the same output port. The address queue can be implemented in different ways . For instance, an address queue can be implemented as a linked list (see Kuwahara et al [12]) or it can be stored in a FIFO buffer which is dedicated to a specific output port (see Lee et al [13]). The shared memory has a finite capacity. Also, limitations can be imposed on the size of each address queue. A cell will be lost if it arrives to find the shared buffer full. It will also be lost if the shared buffer is not full, but its address queue is full. The switch architecture is synchronized. Between two synchronization points any incoming cells that are in process of arriving at the input ports are written to the memory, and each output port transmits a cell (if there is one in its address queue).

An important factor that affects the performance of an ATM switch is the burstiness of the arrival process to each port. In this paper, each arrival process is assumed to be bursty and it is modelled by an *Interrupted Bernoulli Process* (IBP). The burstiness of an arrival process is measured by the squared coefficient of variation of the interarrival time. An IBP process may find itself either in the busy state or in the idle state. Arrivals occur in a Bernoulli fashion only when the process is in the busy state. No arrivals occur if the process is in the idle state. Let us assume that at the end of slot t the process is in the idle (or busy) state. Then, in the next slot $t+1$ it will remain in the idle (or busy) state with probability q (or p), or it will change to busy (or idle) with probability $1-q$ (or $1-p$). The transitions between the busy and idle states are shown in figure 1. If during a slot the process is in the busy state, then with probability λ a cell will arrive during the slot, and with probability $1-\lambda$ no arrival will occur. In this paper, we assume that $\lambda = 1$. That is, at every busy slot there is an arrival.

Let $\tilde{t}$ be the interarrival time of a cell. Then, it can be shown that the probability ρ that any slot contains a cell and the squared coefficient of variation of the interarrival time C^2 are as

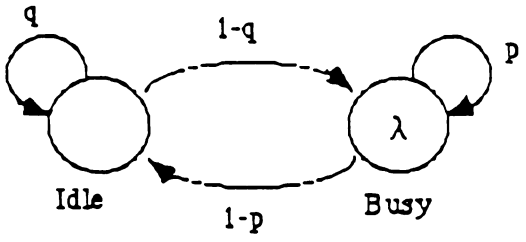

Figure 1: Transition between busy and idle states

follows:

$$\rho = \frac{\lambda(1-q)}{2-p-q},$$

and

$$\begin{aligned} C^2 &= \frac{Var(\bar{t})}{E[\bar{t}]^2} \\ &= 1 + \lambda(\frac{(1-p)(p+q)}{(2-p-q)^2} - 1). \end{aligned}$$

For $\lambda = 1$, we have

$$\rho = \frac{1-q}{2-p-q}, \tag{1}$$

and

$$C^2 = \frac{(p+q)(1-p)}{(2-(p+q))^2}. \tag{2}$$

We model the shared buffer switch by the queueing system shown in figure 2. The queueing model consists of N single server queues. Each of these queues represents an address queue in the shared buffer switch. The server of each queue represents an output port. There are N arrival streams, one per input port. Each arrival stream is assumed to be bursty and it is modelled as an IBP. We assume that the N arrival processes are heterogeneous. A cell upon arrival at the ith input port joins the jth queue with probability b_{ij}, where $\sum_{j=1}^{N} b_{ij} = 1$, $i = 1, 2, \cdots, N$. The total number of cells in all queues cannot exceed M, the buffer size of the swtich. Also, it is possible that the total number of cells in each queue i may be limited to M_i, where $M_i < M$. In this paper, we will assume that $M_i = M$. The case where $M_i < M$ can be easily incorporated.

All servers are synchronized so that they start and end service at the same time. The service time is assumed to be equal to one slot. At the end of each slot, a cell from each queue (provided that the queue is not empty) departs. Obviously, at most N cells can depart. Each

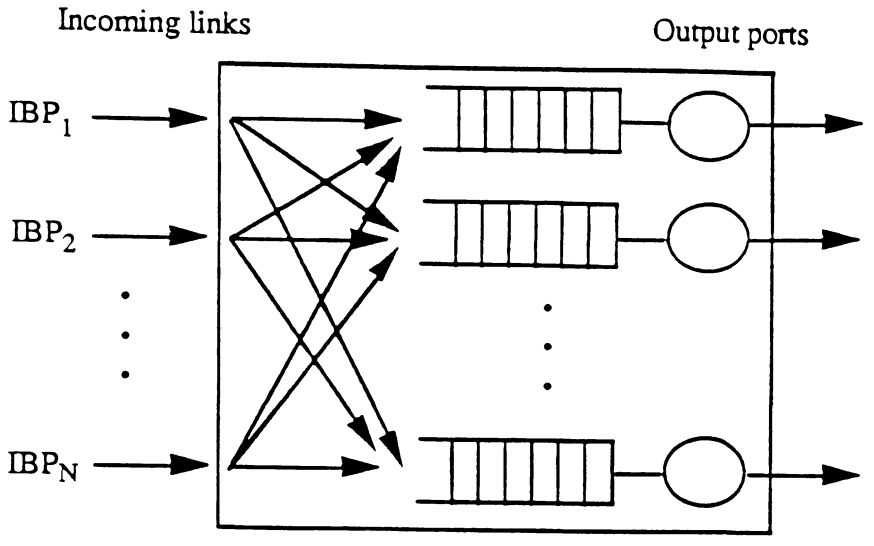

Figure 2: The queueing model of the shared buffer switch

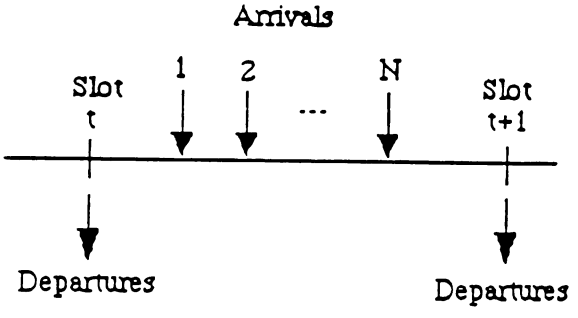

Figure 3: Relation between arrivals and departures

arrival process is also slotted, with a slot size equal to the service time. (That is, the speed of an incoming link is assumed to be equal to the speed of an outgoing link.) We assume that the slot boundaries of the arrival processes lie somewhere between the service time slot boundaries as shown in figure 3. An arriving cell to an empty queue cannot start its service until the begining of the next service slot.

A continuous-time version of this queueing model was analyzed by Yamashita, Perros, and Hong [24]. The analysis presented here is similar in spirit to the analysis presented in [24]. Also, we note that under the assumption of exponential service times and exponential interarrival times, this queueing system is a truncated process of a reversible queueing system and, therefore, it has a product-form solution (see Kelly [10]).

3 The Approximation Algorithm

The state of our model is described immediately after the beginning of each slot by the vector $(\underline{w}; \underline{n}) = (w_1, ..., w_N; n_1, ..., n_N)$, where w_i is the state of the *ith* IBP arrival process, w_i=0, 1, and n_i is the number of cells in the *ith* queue, $n_i = 0, 1, 2, ..., M$, such that $\sum_{i=1}^{N} n_i \leq M$. If $w_i = 0$, then the IBP arrival process is in the idle state, and if $w_i = 1$, then the process is in the busy state. Based on this state description, we can compute the transition probability P_{ij} from state i to state j, and subsequently the matrix of transition probabilities $P = [P_{ij}]$. Finally, solving the steady-state equation $\underline{\pi}P = \underline{\pi}$, where $\underline{\pi}$ is the steady-state probability vector, we can obtain the exact probabilities of all states numerically.

This numerical method can only be used to solve small systems. It becomes intractable as the state space increases. In this paper, we analyze this queueing system using the notion of decomposition. The queueing system is decomposed into subsystems, and then each subsystem is analyzed separately. The results from the subsystems are combined together through the means of an iterative scheme. Furthermore, in order ot reduce the complexity of each subsystem, we aggregate the N arrival processes to a single superposition. The main steps of the approximation algorithm are as follows.

We first consider in isolation the transition matrix of the N heterogeneous arrival processes. The states of this matrix are described by the vector $\underline{w} = (w_1, ..., w_N)$, where $w_i = 1, 2$ and $i = 1, 2, ..., N$. This transition matrix may be very large, seeing that it consists of 2^N states. We aggregate this transition matrix to a considerably smaller transition matrix involving $N + 1$ states, where each aggregate state k gives the number of the arrival processes which are in the busy state, $k = 0, 1, ..., N$. This aggregate matrix describes the superposition of the N arrival processes. If the superposition process is in state k, then there are k active arrival processes and consequent k cells will arrive (since each arrival is an on-off process).

We now consider the queueing system assuming a single arrival stream of cells, described by the above superposition process. The queueing system is analyzed by descomposing it into N subsystems. Each subsystem comprises one queue of the queueing system. The remaining queues are represented in the subsystem indirectly by simply keeping count of the total number of cells. As it will be shown below, the state space of a subsystem is not very big. Consequently, each subsystem is analyzed numerically. The N subsystems are combined together using an iterative procedure. We now proceed to describe the approximation algorithm in detail.

3.1 Aggregation of the N Arrival Processes

Let us consider the N heterogeneous arrival processes in isolation. The state of this system is described by the vector $\underline{w} = (w_1, w_2, ..., w_N)$. Let $r(\underline{w} \rightarrow \underline{w'})$ be the transition probability from state $\underline{w}$ to state $\underline{w'}$. Define S_i to be the set of all states $\underline{w}$ in which there are exactly i arrival processes in the busy state. Using this definition we can lump the state space into $N + 1$ sets of states S_i, $i = 0, 1, 2, ..., N$. The transition matrix with the lumped states is shown in figure 4.

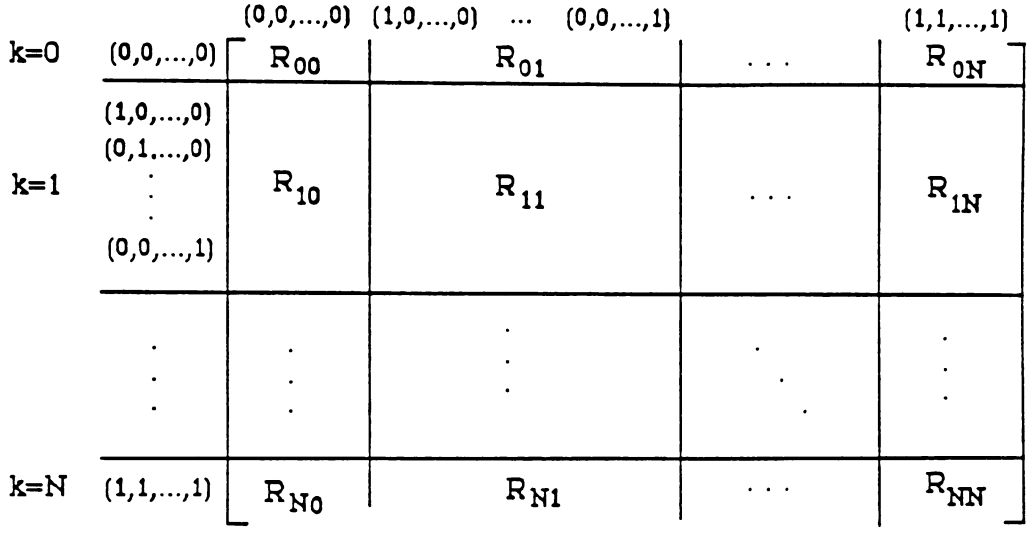

Figure 4: The transition matrix with the lumped states

Let R_{ij} be the transition probability from lump i to lump j. Then, we have

$$R_{ij} = \sum_{\underline{w} \in S_i} Pr[\underline{w}|S_i][\sum_{\underline{w'} \in S_j} r(\underline{w} \rightarrow \underline{w'})], \tag{3}$$

where

$$Pr[\underline{w}|S_i] = \frac{Pr[\underline{w}]}{Pr[S_i]}$$

and

$$Pr[S_i] = \sum_{\underline{w} \in S_i} Pr[\underline{w}].$$

Seeing that the N arrival processes are independent from each other, we have that $Pr[\underline{w}] = Pr[w_1] \cdot Pr[w_2] \cdots Pr[w_N]$, where

$$Pr[w_i = 0] = \frac{1 - p_i}{2 - (p_i + q_i)},$$

and

$$Pr[w_i = 1] = \frac{1 - q_i}{2 - (p_i + q_i)}.$$

Thus, we have

$$Pr[\underline{w}] = [\prod_{i \in W_0} \frac{1-p_i}{2-(p_i+q_i)}][\prod_{i \in W_1} \frac{1-q_i}{2-(p_i+q_i)}],$$

where for a given state $\underline{w}$, W_0 and W_1 are the sets of the arrival processes which are in the idle state and the busy state respectively.

The generation of the aggregate transition matrix (hereafter denoted as A) is a time consuming operation because we first have to generate the transition matrix associated with the N arrival processes and subsequently calculate the transition probabilities R_{ij}, $i, j = 0, 1, 2, ..., N$.

The computational effort for the generation of the aggregate matrix A can be significantly reduced by observing that the parameters p and q of a bursty IBP process are very likely to be close to 1. For instance, in the case of voice, the squared coefficient of variation of the inter-arrival time C^2 is 18.1(see Heffes and Lucantoni [8]). Assuming a link utilization of 0.1, and using equation(1) and (2), we find that $p = 0.922$ and $q = 0.991$. The higher the value of C^2, the closer to 1 p and q get. It is highly unlikely, therefore, that a bursty source will change its current state at the next slot, seeing that $1 - p \simeq 0$ and $1 - q \simeq 0$. Based on this argument, we neglect all the transitions where more than two arrival processes change their states in the next slot. This results in great savings when it comes to generate the transition matrix of the N arrival processes. As it will be seen in the validation section, neglecting these transtions does not affect the accuracy of the results. The structure of the non-zero elements of the resulting aggregate matrix A is shown in figure 5. Finally, we note that further computational savings can be achieved using some of the combinatorial expressions given in Iscoe, McDonald, and Qian [9].

3.2 Analysis of a subsystem

As it was mentioned earlier on, we analyze the queueing system under study by decomposing it into N subsystems, one per queue. In this section, we describe how a subsystem is analyzed. The subsystems are combined together using the iterative procedure described in section 3.3.

Let us consider the *ith* queue. The remaining queues will be denoted by the symbol $N - \{i\}$. In order to study the *ith* queue in isolation, we need to know how many cells are in $N - \{i\}$. Consequently, we analyze queue i by studying a subsystem of the original queueing system which consists of the following: (a) the arrival stream characterized by the superposition process, (b) queue i, and (c) the number of cells in $N - \{i\}$. The queueing structure of a subsystem i is shown in figure 6. The state of a subsystem i is given by the vector (k, n_i, n), where k is the state of the superposition process, n_i is the number of cells in the *ith* queue, and n is the number of cells in the entire system, i.e., $n - n_i$ is the number of cells in the remaining queues $N - \{i\}$. The total number of states in the subsystem is $\frac{(N+1)(M+1)(M+2)}{2}$. The synchronization of the departures and arrivals in a subsystem is the same as in the original queueing system. This is shown in figure 3. For presentation purposes, we shall refer to time t as being immediately after the end of slot t.

Each subsystem is analyzed numerically by first generating its undelying transition matrix and then solving the resulting system of linear equations to obtain its stationary probability vector.

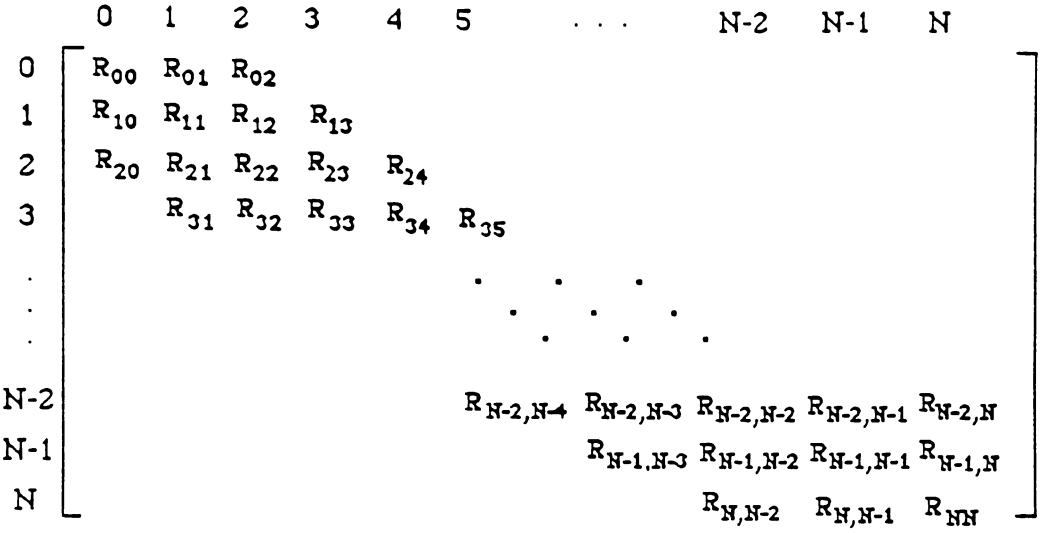

Figure 5: Aggregate transition matrix A

The automatic generation of the states of a subsystem is not discussed here since it is a straightforward procedure. Below, we describe how the one-step transition probabilities are calculated. A one-step transition from state (k, n_i, n) to state (k', n_i', n') takes place in subsystem i if one or more of the following events occur:

1. state change of the arrival process
2. cell arrivals to queue i
3. cell arrivals to $N - \{i\}$
4. cell departures from queue i
5. cell departures from $N - \{i\}$

In case (1), the state of the arrival process is described by the number k of the arrival processes which are in busy states. The transition probability from k to k' is $R_{kk'}$ and it is given by expression (3). Let us consider now cases (2) and (3). Since there are k arrival processes which are in busy state at time t, k cells will arrive to queue i or to the remaining queues $N - \{i\}$ between time t and time $t + 1$. If there is not enough space in the shared buffer to accomadate k cells, some of them will be lost. The probability that a cell joins queue i or $N - \{i\}$ is computed using the branching probabilities b_{ji} $j, i = 1, ..., N$, i.e., the probability that a cell arriving at input port j is

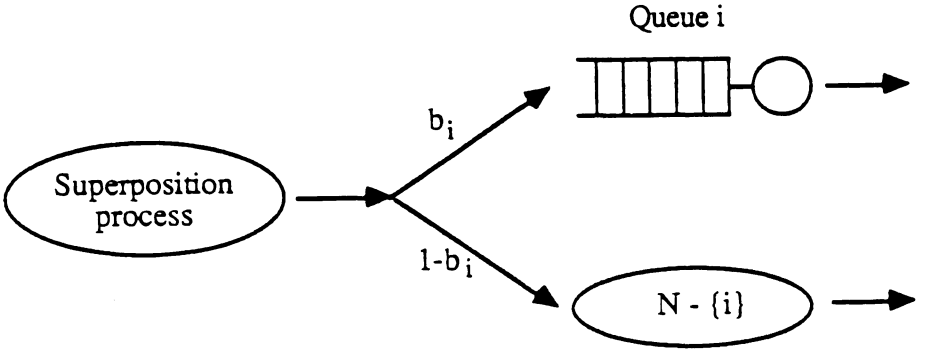

Figure 6: Queueing structure of subsystem i

routed to output port i. We assume that every cell has the same branching probabilities regardless of the input port at which it arrived. Thus, with probability b_i, ,where $b_i = \sum_{j=1}^{N} b_{ji}$, an arriving cell joins the ith queue. In case (4), a cell always depart at the end of a slot if queue i is not empty with probability 1.

In case (5), the number of cells that depart from the remaining queues $N - \{i\}$ depends on how many of these queues are busy. From the state description (k, n_i, n), we only know that there are $n - n_i$ cells in $N - \{i\}$. We do not know how these cells are distributed over the $N - \{i\}$ queues. For instance, all the cells could be in the same queue. In this case, only one cell will depart at the end of the next slot. On the other hand, all $N - 1$ queues may contain at least one cell (provided that $n - n_i > N - 1$). In this case, there will be $N - 1$ departures. In order to calculate the transition probabilities out of state (k, n_i, n) due to cell departures from $N - \{i\}$, we use the probability distribution $P_{\alpha_i}(l)$ that there are l busy queues in $N - \{i\}$, given that there are $\alpha_i = n - n_i$ cells in $N - \{i\}$. This probability is obtained approximately as follows.

As it will be described below in section 3.3, at each iteration we will have an estimate of the mean number of non-empty queues, $f_i(\alpha_i)$, in $N - \{i\}$ for subsystem i, given that there are α_i cells in $N - \{i\}$. For each α_i, $\alpha_i = 1, 2, \cdots, M$, the unknown probability distribution $P_{\alpha_i}(l)$ satisfies the following constraints:

$$\sum_{l=1}^{N-1} P_{\alpha_i}(l) = 1 \tag{4}$$

$$\sum_{l=1}^{N-1} l P_{\alpha_i}(l) = f_i(\alpha_i), \tag{5}$$

where we assume that $P_{\alpha_i}(l) = 0$ if $l > \alpha_i$.

Since only the first moment of the probability distribution is given, there is an infinite number of probability distributions that satisfy constraint (5). We choose a distribution whose first moment matches $f_i(\alpha_i)$ using the notion of maximum entropy (see Kouvatsos and Xenios [11] and the references within). The maximum entropy distribution is the distribution that maximizes the entropy function $\sum_{l=1}^{N-1} P_{\alpha_i}(l) ln(P_{\alpha_i}(l))$ subject to constraints (4) and (5). Using the method of the Lagrange's undetermined multipliers we obtain:

$$P_{\alpha_i}(l) = \frac{1}{Z} e^{-l\beta}, \tag{6}$$

where Z is the normalizing constant given by

$$Z = e^{-\beta_0} = e^{-\beta} + e^{-2\beta} + \cdots + e^{-N\beta}, \tag{7}$$

and β_0 is the Lagrange multiplier associated with the normalizing constraint (4). It can be easily shown that the Lagrangian multiplyer β satisfies the following relation:

$$-\frac{\partial \beta_0}{\partial \beta} = f_i(\alpha_i). \tag{8}$$

β is determined from equation (8) numerically. Therefore, the probability distribution, $P_{\alpha_i}(l)$, can be obtained using eqations (6), (7) and (8).

The algorithm that computes the transition probability from (k, n_i, n) to (k', n_i', n'), $Pr\{(k, n_i, n) \rightarrow (k', n_i', n')\}$, is summarized below. We use the following notation: a_i and a_R are the number of cells that arrive at queue i and $N - \{i\}$ respectively, a_N is the number of cells that enter the whole system without blocking, n_R is the number of cells in $N - \{i\}$, d_i and d_R are the number of cells to depart from queue i and $N - \{i\}$ respectively. All random variables are considered at time t except n_R', and a_R' which are considered at the time $t + 1$.

Algorithm

step 1: $a_i = n_i' + d_i - n_i$, where d_i is 1 if queue is non-empty, 0 otherwise.
step 2: *For* $d_R = 1$ *to* Min(N-1,n_R)
begin
$a_R' = n_R' + d_R - n_R$
if $a_i + a_R = k$ (if cells are not blocked) *then*

$$Pr\{(k, n_i, n) \rightarrow (k', n_i', n')\} = R_{kk'} Pr\{d_R\} [\binom{k}{a_i} b_i^{a_i} (1 - b_i)^{k - a_i}].$$

if $a_i + a_R < k$ (if cells are blocked) *then*
$a_N = a_i + a_R$

$$Pr\{(k, n_i, n) \rightarrow (k', n_i', n')\} = R_{kk'} Pr\{d_R\} [\binom{k}{a_i} b_i^{a_i} (1 - b_i)^{k - a_i} \frac{\binom{a_i}{a_i}\binom{k - a_i}{a_R}}{\binom{k}{a_N}}$$

$$+\binom{k}{a_i+1} b_i^{a_i+1}(1-b_i)^{k-a_i-1}\frac{\binom{a_i+1}{a_i}\binom{k-a_i-1}{a_R}}{\binom{k}{a_N}}$$

$$\vdots$$

$$+\binom{k}{k-a_R} b_i^{k-a_R}(1-b_i)^{a_R}\frac{\binom{k-a_R}{a_i}\binom{a_R}{a_R}}{\binom{k}{a_N}}]$$

end;

Once the transition probability matrix is set up, the stationary probability vector is obtained numerically.

3.3 The Iterative Procedure

Each subsystem i is analyzed in isolation if we know the mean number of non-empty queues $f_i(\alpha_i)$ in $N-\{i\}$, given that there are α_i cells in $N-\{i\}$. The results obtained from all the subsystems are combined through an iterative procedure. Each iteration involves the solution of the N subsystems in isolation. At the end of each iteration a convergence test is carried out. If the procedure has not converged the values of $f_i(\alpha_i)$, $\alpha_i = 1,2,\cdots,M$, $i = 1,2,\cdots,N$, the values of $f_i(\alpha_i)$ are adjusted and another iteration is carried out. For each subsystem i, $f_i(\alpha_i)$ satisfies the following conditions:

1. $f_i(1) = 1$
2. $f_i(\alpha_i + 1) > f_i(\alpha_i)$, α_i=1,2,...,M
3. $f_i(\alpha_i + 1) - f_i(\alpha_i) > f_i(\alpha_i + 2) - f_i(\alpha_i + 1)$
4. $f_i(\alpha_i) < min[n-1, \alpha_i]$

$f_i(\alpha_i)$ is calculated as follows. Let $g_i(\alpha_i)$ be the mean number of non-empty queues in queue $N-\{i\}$ when α_i cells arrive at the entire queueing system assuming that the service time at each queue is infinite. $g_i(\alpha_i)$ is an upper bound of $f_i(\alpha_i)$, and it is given by the following expression.

$$g_i(\alpha_i) = \sum_{l=1,l\neq i}^{n} [1 - (\frac{\sum_{j=1,j\neq i,l}^{n} \rho_j}{\sum_{j=1,j\neq i}^{n} \rho_j})^{\alpha_i}]. \tag{9}$$

where ρ_j is the average arrival rate of the jth arrival process (given by expression (1)). Clearly $1 \leq f_i(\alpha_i) \leq g_i(\alpha_i)$. $f_i(\alpha_i)$ is approximated as follows:

$$f_i(\alpha_i, \beta) = 1 + \beta[g_i(\alpha_i) - 1], \tag{10}$$

where $0 \leq \beta \leq 1$. Note that $g_i(\alpha_i)$ satisfies the above four conditions, and so does $f_i(\alpha_i, \beta)$. We estimate $f_i(\alpha_i)$ iteratively by adjusting β up and down accordingly in order to meet a convergence criterion.

In this study, we use the mean number of cells in the entire queueing system as a convergence criterion because it is computationally simple. Since we obtain the stationary probabilities $P_i(k, n_i, n)$ for each subsystem i, the mean number of cells in the entire queueing system can be computed in the following two different ways:

$$E_1 = \sum_{i=1}^{N} \sum_{n_i=1}^{M} n_i P_i(n_i) \tag{11}$$

$$E_2 = \frac{1}{N} \sum_{i=1}^{N} \sum_{n=1}^{M} n P_i(n) \tag{12}$$

where $P_i(n_i)$ is the marginal probability that there are n_i cells in the ith queue, and $P_i(n)$ is the marginal probability that there are n cells in the entire queueing system as calculated from subsystem i. These two probabilities are obtained as follows:

$$P_i(n_i) = \sum_{k=0}^{N} \sum_{n=n_i}^{M} P_i(k, n_i, n),$$

and

$$P_i(n) = \sum_{k=0}^{N} \sum_{n_i=0}^{n} P_i(k, n_i, n).$$

Convergence occurs if $|E_1 - E_2| \leq \epsilon$. If convergence has not occured, we adjust β and do another iteration. In particular, we note that $E_1 - E_2$ is a monotonically increasing function of $f_i(\alpha_i)$. Thus, we can decide whether the $f_i(\alpha_i)$ is overestimated or underestimated depending on which mean value is larger, and accordingly change $f_i(\alpha_i)$ in order to reduce the difference E_1 and E_2. Below, we summarize the approximation algorithm. The superscript s denotes the iteration number.

Algorithm

step 0: Set $\beta_L^0 = 0$ and $\beta_H^0 = 1$ and $s = 0$

step 1: $s = s + 1$, $\beta_M^S = (\beta_H^{S-1} + \beta_L^{S-1})/2$ and calculate $f_i^S(\alpha, \beta_M^S)$ for α=1,...,M and i=1,2,...,N using (9) and (10).

step 2: Calculate the probability distribution of nonempty queues, $P_{\alpha_i}(l)$ using (6), (7), and (8) for $l = 1, 2, \cdots, N-1$.

step 3: Set-up the transition probability matrix for the each subsystem using the algorithm described in section 3.2, and calculate the stationary vector of $P_i(k, n_i, n)$

using the SOR method for $i = 1, 2, ..., N$.

step 4: Calculate E_1^S and E_2^S using (11) and (12) respectively.

step 5: If $|E_1 - E_2| \leq \epsilon$ then stop.
if $E_1 - E_2 > \epsilon$ then set $\beta_H^S = \beta_M^S$, $\beta_L^S = \beta_L^{S-1}$ and go to step 1.
if $E_1 - E_2 < -\epsilon$ then $\beta_H^S = \beta_H^{S-1}$, $\beta_L^S = \beta_M^S$ and go to step 1.

Due to the nature of this iterative procedure it is possible that the quantities $f_i(\alpha_i)$ may converge before $|E_1 - E_2| < \epsilon$. In this case, no further improvement on the approximation results can be achieved. Consequently, for convergence it is important to also check if $f_i(\alpha_i)$ have converged. In order to simplify the convergence criteria, it was found emprically that it suffices to check if the absolute value of the relative error between E_1 and E_2 is less than ε, i.e. $|\frac{E_1 - E_2}{E_1}| \leq \varepsilon$.

4 Validation

The approximation algorithm was validated in two parts. In the first part, we checked the accuracy of the approximation method described in section 3.1 for constructing the superposition process. In the second part, we checked the accuracy of the entire approximation algorithm for analyzing the queueing model of the shared buffer switch shown in figure 2. All validations were carried out by comparing the approximate results against simulation data.

4.1 The Superposition Process

The accuracy of the superposition process was tested by analyzing an ATM multiplexer. Specifically, we considered a single finite capacity queue with N arrival process as shown in figure 7. Let B be the maximum capacity of the queue. The server is assumed to be slotted, and the service time is equal to 1 slot. Service begins and ends at slot boundaries. Each arrival stream is slotted, with a slot equal to a service time, and it is modelled by an IBP with $\lambda = 1$ (i.e. each busy slot contains a cell). The synchronization of the arrivals and the departures is shown in figure 3. If a cell arrives when the system is empty, the cell waits until the beginning of the next service slot, and then begins its service. The cell departs one slot later.

The ATM multiplexer is analyzed using the techniques described in sections 3.1 and 3.2. In particular, the N arrival processes are aggregated to a single superposition process with $N + 1$ states. The finite capacity queue is then analyzed assuming that all arrivals are generated by the superposition process. The state of this system is given by the vector (k, n), where k is the state of the superposition process, and n is the number of cells in the queue. (We note that the state space is a subset of the state space of the subsystem described in section 3.2) The system is observed immediately after the beginning of each slot. If it is found in state (k, n), then during the subsequent slot k arrivals will occur and at the end of the slot one cell will depart (given that $n > 0$). The transition probabilities $Pr\{(k, n) \rightarrow (k', n')\}$ between two states obtained using the transition probabilities R_{ij} (see expression 3) of the aggregate matrix A. We have

$$Pr\{(k, n) \rightarrow (k', n')\} = Pr\{(k, n) \rightarrow (k', min(max(0, n - 1) + k, B))\}$$

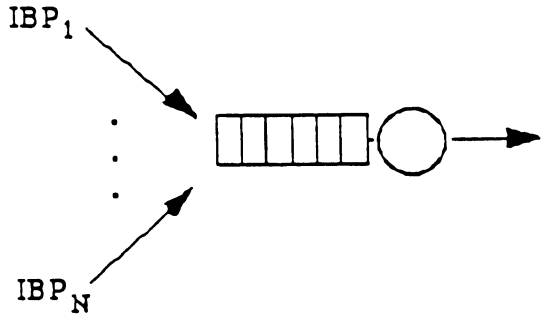

Figure 7: The ATM multiplexer

$$= R_{kk'}$$

We can now solve numerically the system of linear equations $\underline{\pi}P = \underline{\pi}$ where $P = Pr\{(k,n) \rightarrow (k',n')\}$, in order to obtain the steady state probabilities $\pi(k,n)$. In our experiments, we used the successive overrelaxation method to obtain $\pi(k,n)$. Once we have calculated the stationary vector $\pi(k,n)$, we can calculate the probability distribution of the number of cells in the queue $P(n)$, and the probability $P(r)$ that a cell will wait in the queue for r slots before it is transmitted. $P(r)$ is calculated as follows:

If we exclude the transmission time, every cell waits for a minimum of 0 slots and a maximum of $B-1$ slots. Let C_r denote the expected number of cells which wait exactly r slots and C_E denote the expected number of cells that entered the queue. If the current state at slot i is (k,n), then k cells would arrive in the next slot $i+1$. The number of cells in slot $i+1$ is equal to the sum of the remaining cells and the new arriving cells, i.e., $(n-1)+k$. If this sum is less than the buffer size B, then all k cells can enter the queue. However, if it is greater than B, only the cells which the buffer can accomodate, i.e., $B-(n-1)$, can enter the queue, and the others are lost. Thus, C_E is expressed as

$$C_E = \sum_{n=0}^{B} \sum_{k=1}^{N} \tilde{k}\pi(k,n)$$

where $\tilde{k} = min(k, B-(n-1))$.

Now, let us consider C_r. If the queue is empty or it contains one cell and the number of arriving cells at the next slot is more than r $(k > r)$, then exactly one cell among the arriving cells will wait for r slots in the queue before transmission. Likewise, if there are already two cells in the queue, one cell among arriving cells at the next slot will wait for r slots only when more than $r-1$ cells arrive. If there are already $r+1$ cells in the queue, one cell among the arriving cells will always wait for r slots regardless of the number of arriving cells k. Thus, C_r can be obtained as

follows:

$$C_r = \sum_{n=0}^{r+1} \sum_{k=r+1-\bar{n}}^{N} \pi(k,n)$$

where $\bar{n} = max(0, n-1)$. The probability $P(r)$ that a cell waits exactly r slots before transmission can now be obtained by dividing C_r by C_E. (Note that in this calculation we do not include the offset of the arrival boundary to the next service boundary.)

The results obtained for the ATM multiplexer using the above approximation method were validated against simulation. In figures 8 and 10, we give the approximate and simulation results for $P(n)$ and $P(r)$ respectively, assuming 16 arrival processes ($N = 16$), and a buffer capacity of 32 ($B = 32$). (The confidence intervals were not plotted because they were too small.) The absolute errors (simulation-approximation) are given in figure 9 and 11. The arrival processes were set as follows: $\rho_i = 0.03125$, i=1,2,...,16, and $C_1^2 = C_2^2 = C_3^2 = C_4^2 = 50$, $C_5^2 = C_6^2 = C_7^2 = C_8^2 = 100$, $C_9^2 = C_{10}^2 = C_{11}^2 = C_{12}^2 = 200$, $C_{13}^2 = C_{14}^2 = C_{15}^2 = C_{16}^2 = 500$. We observe that there is a good agreement between the approximate and simulation data. In general, it has been observed that the approximate results have a good accuracy. The accuracy seems to be affected by the variability in the C^2 values of the arrival processes. (The squared coefficient of variation of the interarrival time C^2 is used in this paper as a measure of burstiness of an arrival process). For example, the accuracy of the approximate results will not be as good if the C^2 values of the arrival processes range from 1 to 500, rather than from 50 to 500. Depending on the number of arrival processes, the construction of the aggregate matrix may be CPU intensive. This is because it requires the generation of the Markov chain of the N arrival processes, which may involve a large number of states. CPU complexity problems were encountered when $N > 20$. The case of $N = 8$ can run efficiently on a workstation. The case of $N = 16$ required a mainframe system.

The approximation method for constructing the superposition process can be easily adapted to the case where each individual arrival processes is defined in continuous-time as an *Interrupted Poisson Process* (IPP). In this case, the construction of the aggregate matrix A is not time-consuming as in the discrete-time case. Heffes [7] considered the case of superposing heterogeneous arrival processes. In particular, Heffes constructs an MMPP approximation to the superposition of any number of different MMPP sources using a set of very simple expressions. In order to compare our method against Heffes' method, we considered a continuous-time version of the ATM multiplexer where each arrival stream is represented by a different IPP, and the server provides an exponentially distributed service time. We assumed 8 arrival processes and a buffer capacity of 16. Each arrival process had an average arrival rate of 0.5, and a peak arrival rate of 1. The squared coefficient of variation of the interarrival time of the arrival processes was set as follows: $C_1^2 = 200$, $C_2^2 = 100$, $C_3^2 = 80$, $C_4^2 = 50$, $C_5^2 = 40$, $C_6^2 = 20$, $C_7^2 = 10$, $C_8^2 = 2$. The total offered average traffic was $\sum_{i=1}^{8} \lambda_i = 4$. The service rate μ was obtained from the ratio $\nu = \sum_{i=1}^{8} \lambda_i/\mu$, which in our experiments was varied from 0.1 to 0.96.

Using Heffes' approximation, we obtained an MMPP approximation to the superposition of the 8 arrival processes. Subsequently, we analyzed the multiplexer numerically as an MMPP/M/1 finite capacity queue. The results for the grade-of-service(GOS), expressed as $100(1-Pr\{cell\ loss\})$, and the mean queue-length as a function of ν are given in figures 12 and 13 respectively. In the same figures, we give results obtained using our method, but adapted for the continuous-time case,

and we also give simulation results. We note that our approach gives better estimates for GOS as ν increases. This is probably due to the fact that the aggregate matrix A contains more information regarding the superposition process than Heffes' MMPP. However, our approximation method is computationally more intensive.

4.2 The Shared Buffer Switch

In this section, we discuss the validation of the approximation algorithm described in section 3 for the analysis of the shared buffer switch. The approximation algorithm was used to analyze an 8×8 and a 16×16 shared buffer switch. The approximation results were compared against simulation data. Representative results are summarized in tables 2 to 5. Each table gives approximate and simulation results for the global queue-length distribution $P(n)$, $n = 0, 1, 2, \cdots, M$, and for the queue-length distribution $P_1(n_1)$, $n_1 = 0, 1, 2, \cdots, M$, of queue 1 which is the most heavily utilized queue (hot spot). Approximate and simulation results for the mean number of cells, m.q.l., in the switch is also given. The absolute errors for $P(n)$ and $P_1(n_1)$ given in tables 2 to 5 have been plotted in figures 14 to 17 respectively. The parameters of the arrival processes are given in table 1.

In general, the approximation algorithm gives good results. The approximation results for the queue-length distribution $P_i(n_i)$ of each queue i seem to be slightly more accurate than the global queue-length distribution $P(n)$. CPU complexity problems were encountered when $N > 20$. The case of $N = 8$ can be done efficiently on a workstation. The case $N = 16$ required a mainframe system.

5 Conclusions

In this paper, we presented an approximation algorithm for analyzing a discrete-time queueing model of the shared buffer ATM switch. The arrival process to each port of the switch was assumed to be bursty, and it was modelled by an IBP. The discrete-time queueing model was analyzed approximately using the notion of decomposition. The queueing model was decomposed into subsystems, and each subsystem was analyzed separately. The results from the subsystems were combined together through the means of an iterative scheme. In order to reduce the complexity of each subsystem, the arrival processes to the switch were aggregated to a single superposition process. Comparisions against simulation data showed that the approximation algorithm has a good accuracy.

table 2	$\rho_i = 0.2, i = 1, 2, \cdots, 8;$ $C_1^2 = C_2^2 = 500, C_3^2 = C_4^2 = 200, C_5^2 = C_6^2 = 100, C_7^2 = C_8^2 = 50;$ $b_{i1} = 0.5, b_{i2} = \cdots = b_{i5} = 0.1, b_{i6} = 0.05, b_{i7} = b_{i8} = 0.025$
table 3	same as in table 2
table 4	$\rho_i = 0.2, i = 1, 2, \cdots, 16;$ $C_1^2 = \cdots = C_4^2 = 500, C_5^2 = \cdots = C_8^2 = 200, C_9^2 = \cdots = C_{12}^2 = 100,$ $C_{13}^2 = \cdots = C_{16}^2 = 50;$ $b_{i1} = 0.21, b_{i2} = \cdots = b_{i5} = 0.082, b_{i6} = \cdots = C_{i9} = 0.06,$ $b_{i10} = \cdots = b_{i13} = 0.041, b_{i14} = \cdots = b_{i16} = 0.020$
table 5	$\rho_i i$ and $C_i, i = 1, 2, \cdots, 16$, same as in table 4; $b_{i1} = 0.339, b_{i2} = \cdots = b_{i5} = 0.068, b_{i6} = \cdots = b_{i9} = 0.051,$ $b_{i10} = \cdots = b_{i13} = 0.034, b_{i14} = \cdots = b_{i16} = 0.017$

Table 1: Parameters of the arrival processes

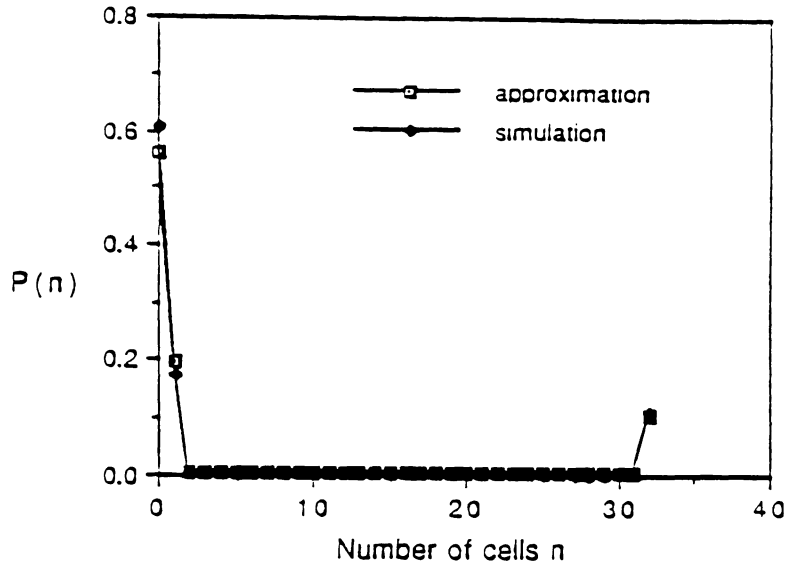

Figure 8: Queue length distribution

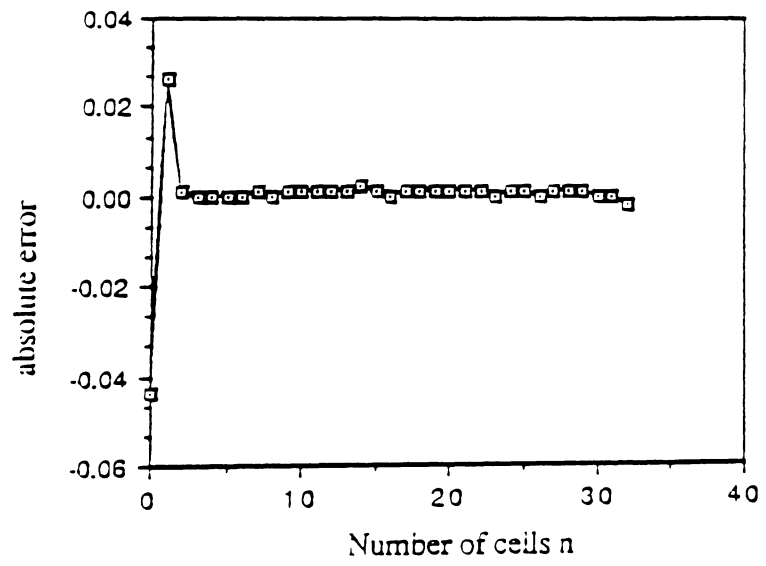

Figure 9: Absolute errors for the results in figure 8

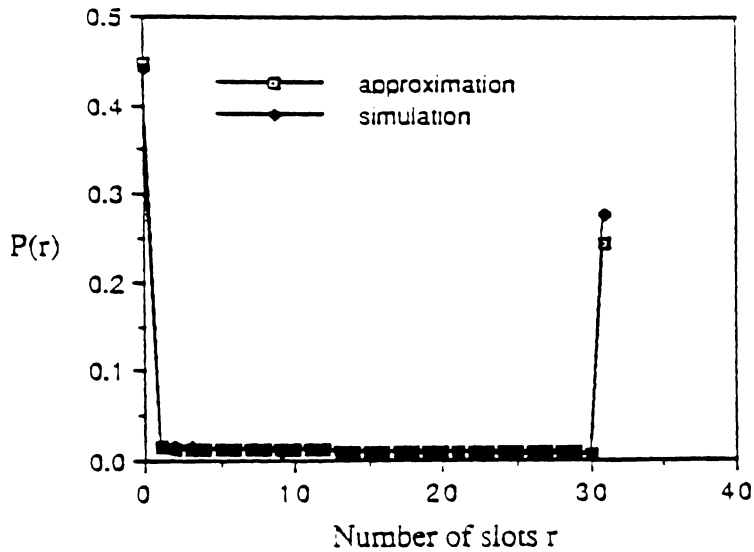

Figure 10: Cell delay distribution

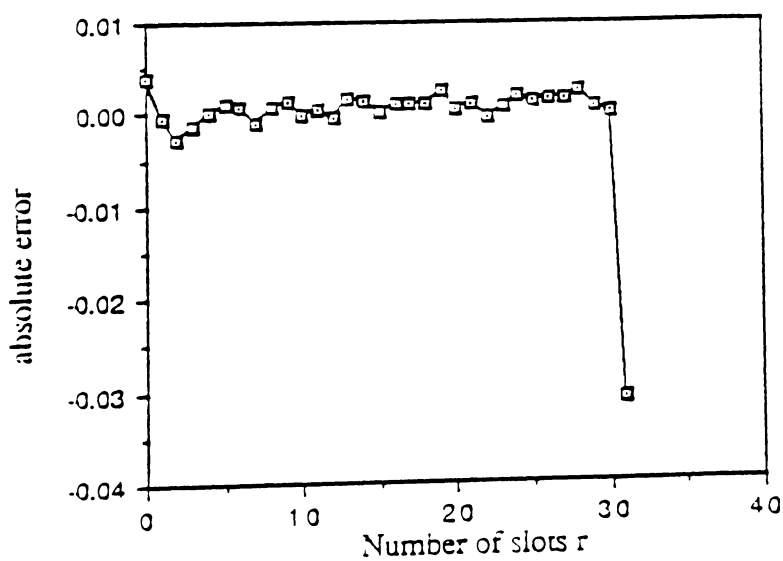

Figure 11: Absolute errors for the results in figure 10

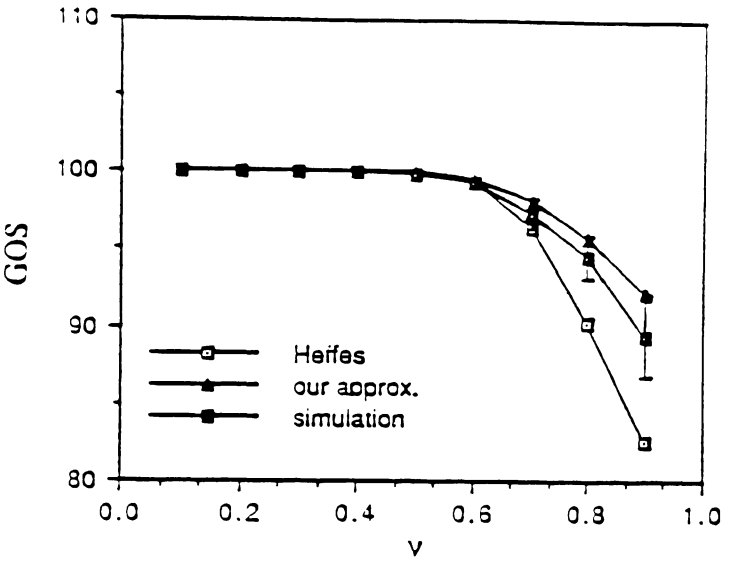

Figure 12: Comparisons for grade-of-service(GOS)

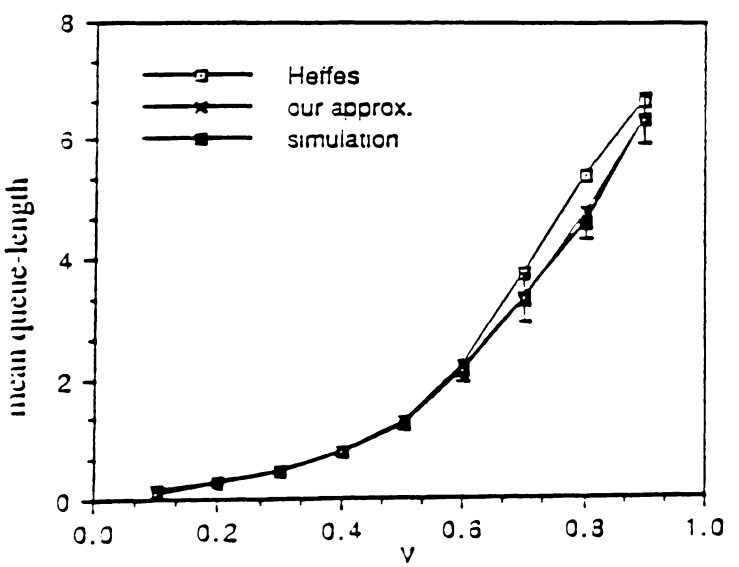

Figure 13: Comparisons for the mean queue-length

n	P(n)		n_1	$P_1(n_1)$	
	approx.	simul.		approx.	simul.
0	.1666	.1608±.0054	0	.3329	.3264±.0065
1	.3004	.2999±.0061	1	.1993	.1971±.0037
2	.0469	.0642±.0022	2	.0595	.0572±.0011
3	.0640	.0572±.0013	3	.0509	.0480±.0007
4	.0501	.0452±.0007	4	.0501	.0445±.0010
5	.0419	.0408±.0011	5	.0599	.0524±.0011
6	.0406	.0410±.0009	6	.0830	.0821±.0024
7	.0462	.0476±.0009	7	.1067	.1225±.0051
8	.2432	.2433±.0105	8	.0576	.0699±.0025
cell loss	.16	.20±.01			
m.q.l	3.38	3.51±.0719			

Table 2: Approximate and simulation results for P(n) and $P_1(n_1)$ switch size: 8×8, buffer size=8.

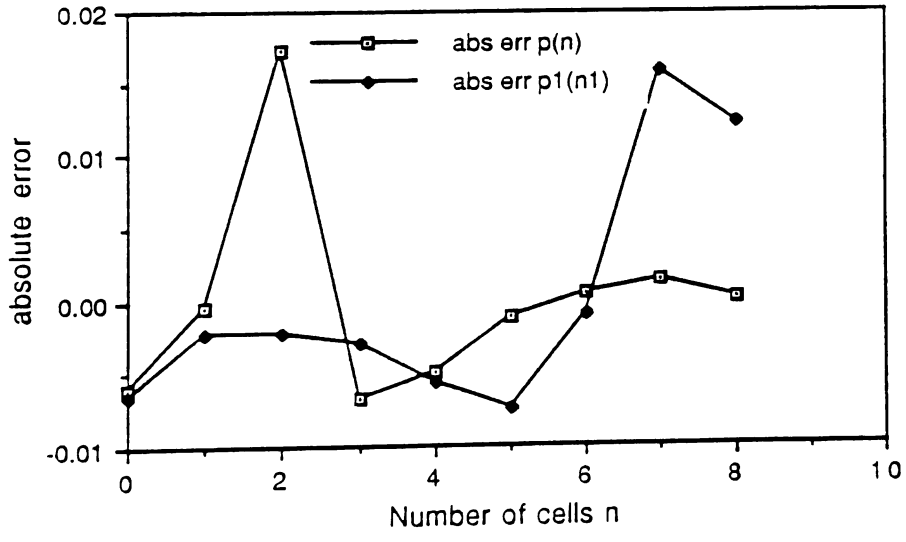

Figure 14: Absolute errors for the results in table 2

n	P(n) approx.	simul.	n_1	$P_1(n_1)$ approx.	simul.
0	.1640	.1591±.0054	0	.3143	.3092±.0066
1	.2741	.2750±.0064	1	.1774	.1754±.0034
2	.0376	.0528±.0016	2	.0473	.0455±.0011
3	.0525	.0460±.0012	3	.0366	.0355±.0010
4	.0401	.0341±.0010	4	.0294	.0280±.0009
5	.0298	.0272±.0007	5	.0246	.0234±.0007
6	.0243	.0228±.0006	6	.0214	.0203±.0006
7	.0210	.0201±.0006	7	.0195	.0188±.0008
8	.0190	.0182±.0007	8	.0185	.0175±.0005
9	.0178	.0172±.0006	9	.0184	.0172±.0007
10	.0173	.0165±.0007	10	.0193	.0176±.0007
11	.0173	.0172±.0007	11	.0220	.0190±.0007
12	.0181	.0182±.0005	12	.0281	.0229±.0008
13	.0197	.0202±.0008	13	.0402	.0331±.0010
14	.0230	.0240±.0009	14	.0605	.0625±.0035
15	.0299	.0320±.0012	15	.0795	.0977±.0057
16	.1945	.1993±.0121	16	.0433	.0564±.0032
cell loss	.139	.148±.008			
m.q.l	5.97	6.21±.16			

Table 3: Approximate and simulation results for P(n) and $P_1(n_1)$ switch size: 8×8, buffer size=16.

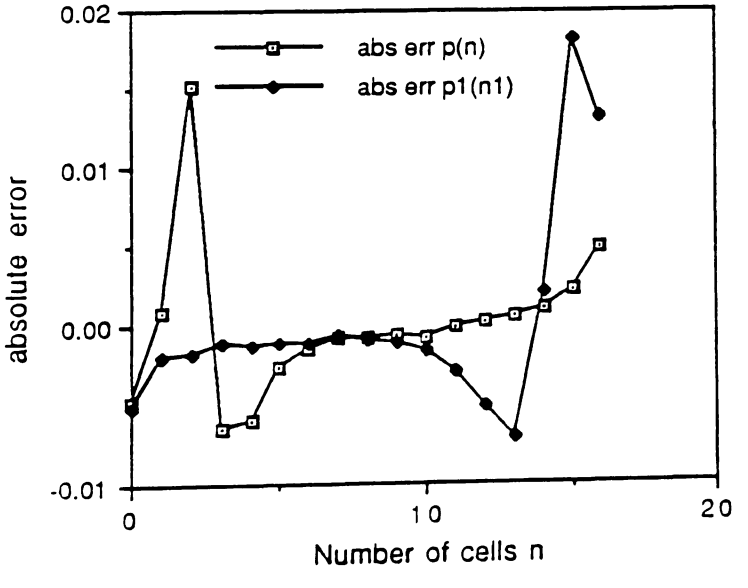

Figure 15: Absolute errors for the results in table 3

n	P(n)		n_1	$P_1(n_1)$	
	approx.	simul.		approx.	simul.
0	.1852	.1978±.0087	0	.6738	.6861±.0083
1	.3264	.3364±.0088	1	.2319	.2265±.0041
2	.2025	.2341±.0053	2	.0541	.0508±.0024
3	.1399	.1143±.0048	3	.0170	.0156±.0014
4	.0673	.0548±.0034	4	.0078	.0071±.0008
5	.0332	.0254±.0022	5	.0044	.0038±.0004
6	.0162	.0128±.0015	6	.0029	.0024±.0003
7	.0089	.0073±.0010	7	.0020	.0017±.0003
8	.0055	.0045±.0005	8	.0015	.0014±.0003
9	.0037	.0029±.0003	9	.0012	.0011±.0002
10	.0026	.0020±.0003	10	.0011	.0010±.0001
11	.0020	.0015±.0002	11	.0009	.0009±.0002
12	.0015	.0012±.0002	12	.0007	.0008±.0002
13	.0012	.0011±.0002	13	.0002	.0005±.0001
14	.0010	.0009±.0002	14	.0	.0002±.0
15	.0009	.0008±.0002	15	.0	.0 ±.0
16	.0021	.0022±.0006	16	.0	.0 ±.0
cell loss	.00181	.00165 ±.00062			
m.q.l	1.89	1.81 ±.06			

Table 4: Approximate and simulation results for P(n) and $P_1(n_1)$ switch size: 16×16, buffer size=16.

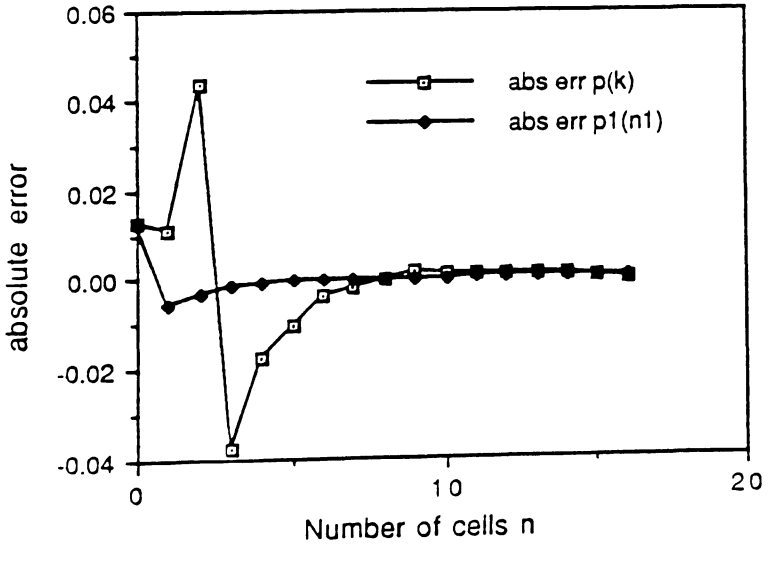

Figure 16: Absolute errors for the results in table 4

n	P(n) approx.	P(n) simul.	n_1	$P_1(n_1)$ approx.	$P_1(n_1)$ simul.
0	.0281	.0293±.0023	0	.3684	.3719±.0132
1	.1083	.1125±.0078	1	.2487	.2474±.0040
2	.1204	.1850±.0083	2	.1110	.1082±.0017
3	.1450	.1649±.0049	3	.0571	.0553±.0015
4	.1219	.1187±.0026	4	.0374	.0351±.0016
5	.0979	.0815±.0017	5	.0286	.0261±.0014
6	.0721	.0546±.0014	6	.0242	.0215±.0012
7	.0534	.0382±.0017	7	.0222	.0193±.0016
8	.0403	.0291±.0015	8	.0215	.0183±.0018
9	.0316	.0234±.0014	9	.0213	.0188±.0020
10	.0258	.0199±.0014	10	.0204	.0196±.0018
11	.0221	.0172±.0014	11	.0174	.0203±.0021
12	.0197	.0156±.0014	12	.0122	.0180±.0018
13	.0183	.0146±.0015	13	.0065	.0127±.0014
14	.0174	.0146±.0013	14	.0023	.0059±.0008
15	.0166	.0158±.0013	15	.0005	.0015±.0002
16	.0602	.0651±.0071	16	.0	.0002±.0
cell loss	.039	.031±.03			
m.q.l	5.	5.15±.20			

Table 5: Approximate and simulation results for P(n) and $P_1(n_1)$ switch size: 16×16, buffer size=16.

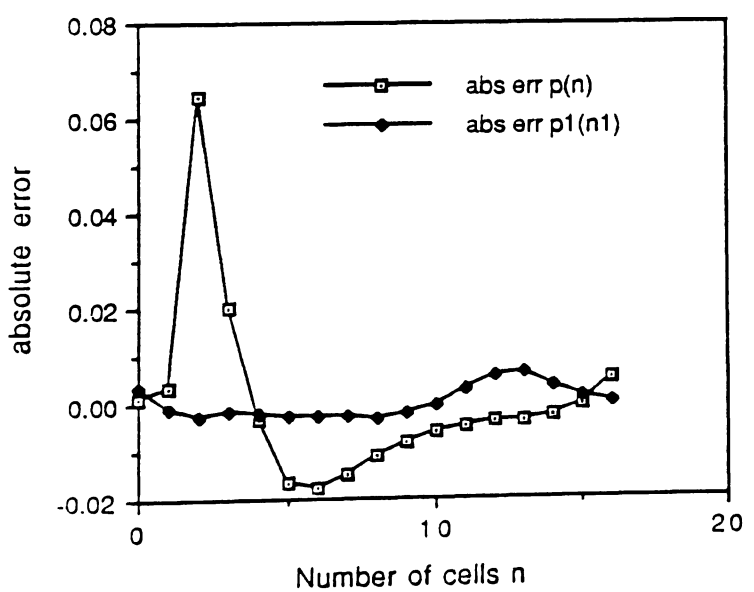

Figure 17: Absolute errors for the results in table 5

References

[1] S. L. Albin, "Approximating a point process by a renewal process, 2: Superposition arrival processes to queues," *Oper. Res.*, 32(5):1133–1162, 1984.

[2] A. Baiocchi, N. Melazzi, M. Listanti, A. Roveri, and R. Winkler, "Loss performance analysis of an ATM multiplexer loaded with high-speed on-off sources," *IEEE J. SAC*, 9:388–393, 1991.

[3] P. Boyer, M. R. Lehnert, and P. J. Kuehn. Queueing in an ATM basic switch element. Technical Report CNET-123-030-CD-CC, CNET, France, 1988.

[4] H. Bruneel and B. Steyaert. Tail distribution of shared buffer queue contents. Technical report, RUG Internal Research Report, Nr. 4, 900522.

[5] M. Devault, J.-Y. Cochennec, and M. Servel, "The "Prelude" ATD experiment: Assessments and future prospects," *IEEE J. SAC*, 6(9):1528–1537, December 1988.

[6] A. E. Eckberg and T.-C. Hou, "Effects of output buffer sharing on buffer requirements in an ATDM packet switch," *INFOCOM '88*, pp. 459–466, March 1988.

[7] H. Heffes, "A class of data traffic processes-covariance function characterization and related queueing results," *Bell Sys. Tech. J.*, 59:897–929, July-August 1980.

[8] H. Heffes and D. M. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE J. SAC*, 4:856–868, September 1986.

[9] I. Iscoe, D. McDonald, and K. Qian. Capacity of ATM switch. Technical report, Dept. Math., Univ. of Ottawa.

[10] F.P. Kelly. *Reversibility and Stochastic Netwroks.* John Wiley & Sons, Inc., 1979.

[11] D.D. Kouvatsos and N.P. Xenios, "MEM for arbitrary queueing networks with multiple general services and repetitive-service blockings," *Perform. Eval.*, 10:169–195, 1989.

[12] H. Kuwahara, N. Endo, M. Ogino, and T. Kozaki, "A shared buffer memory switch for an ATM exchange," *Int. Conf. on Communications*, pp. 4.4.1–4.4.5, Boston, MA, June 1989.

[13] H. Lee, K. Kook, C. S. Rim, K. Jun, and S.-K. Lim, "A limited shared output buffer switch for ATM," *Fourth Int. Conf. on Data Communication Systems and Their Performance*, pp. 163–179, Barcelona, 1990.

[14] R. Nagarajan, J. F. Kurose, and D. Towsley, "Approximation techniques for computing packet loss in finite-buffered voice multiplexers," *IEEE J. SAC*, 9:368–377, 1991.

[15] I. Norros, J. W. Roberts, A. Simmonian, and J. T. Virtamo, "Loss performance analysis of an ATM multiplexer loaded with high-speed on-off sources," *IEEE J. SAC*, 9:378–387, 1991.

[16] H. G. Perros and R. Onvural, "On the superposition of arrival processes for voice and data," *Fourth Int. Conf. on Data Communication Systems and Their Performance*, pp. 341–357, Barcelona, June 1990.

[17] G.H. Petit, A. Buchheister, A. Guerrero, and P. Parmentier, "Performance evaluation methods applicable to an ATM multi-path self-routing switching network," Jensen and Iversen (Ed.), *Teletraffic and datatraffic in a period of change*, pp. 917–922 North Holland, 1991.

[18] G.H. Petit and E. M. Desmet, "Performance evaluation of shared buffer multiserver output queue switches used in ATM," *7th ITC Specialist Seminar*, pp. paper 7.1, New Jersey, 1990.

[19] G. Pujolle and H.G.Perros, "Queueing systems for modelling ATM networks," *Int'l conf. on the Performance of Distributed Systems and Integrated Comm. Networks*, Koyto, September 1991.

[20] B. Sengupta, "A queue with superposition of arrival streams with and application to packet voice technology," King, Mitrani, and Pooley (Ed.), *Performance '90*, pp. 53–59 North Holland, 1990.

[21] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE J. SAC*, 4:833–846, September 1986.

[22] F.A. Tobagi, "Fast packet switch architectures for broadband integrated services networks," *Proc. IEEE*, 78:1133–1167, January 1990.

[23] W. Whitt, "Approximation a point process by a renewal process, 1: Two basic methods," *Oper. Res.*, 30(1):125–147, 1982.

[24] H. Yamashita, H. G. Perros, and S. Hong, "Performance modelling of a shared buffer ATM switch architecture," Jensen and Iversen (Ed.), *Teletraffic and datatraffic in a period of change* North-Holland, 1991.