An Approximation Analysis of a Shared Buffer ATM Switch under Bursty Arrivals

H. Yamashita
H. G. Perros
S. W. Hong

Center for Communications and Signal Procesing
Department of Computer Science
North Carolina State University

TR-90/8
August 1990

An Approximation Analysis of a Shared Buffer ATM Switch Architecture under Bursty Arrivals[1]

H. Yamashita[2], H.G. Perros, S.-W. Hong

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

Abstract: We present an approximation algorithm for the performance analysis of a shared buffer ATM switch architecture. The arrival process to each input port of the ATM switch is assumed to be bursty and it is modelled by an Interrupted Poisson Process. Comparisons against simulation showed that the approximation algorithm has a good error-level.

Keyword: ATM, shared buffer switch, bursty arrivals, Interrupted Poisson Process, queueing models, aggregation, approximation

1. Introduction

One of the most promising solutions for Broadband ISDN is the Asynchronous Transfer Mode (ATM). Many ATM switch designs have been proposed, which provide a high throughput and a low cell loss probability. The most common design is based on multistage interconnection networks. In this type of design, dedicated buffers may be at either the input ports or the output ports, or at both input and output ports. The input buffer switch has a simple architecture, but it achieves a very low throughput. On the other hand the output buffer switch is known to achieve the optimal throughput-delay performance[4].

Another switch architecture is the shared buffer switch, where all the output ports share the same buffer. This switch is based on the "Prelude" switch proposed by CNET, France[3]. This architecture is considered to achieve the optimal throughput-delay performance and requires less buffer memory than the output buffer switch mentioned above[4]. Few studies, however, have been carried out on the performance of this switch architecture. These studies are restricted to single queue analysis[1,5,6].

---

[1]Supported in part by DARPA under grant no. DAEA18-90-C-0039 and by the National Science Foundation under grant no. CCR-87-02258.

[2]On a sabbatical leave of absence from Sophia Univ., Tokyo, Japan.

In this paper, we introduce an approximation algorithm for the performance analysis of the shared buffer switch. The approximation algorithm was developed assuming that the stream of arrivals to each input port is bursty and it is modelled by an Interrupted Poisson Process. In the following section, we describe the queueing model for the switch, and in section 3 we give the approximation algorithm. 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

In the shared buffer switch architecture, the buffer memory is shared by all the switch output ports. An incoming cell destined for the output port i is stored in the shared buffer, and its address is stored in the address buffer. The mechanism to route the cell in the shared buffer to its output port can be implemented in various ways. The cells which have the same output port can be linked by the address chain pointer[5], or their addresses can be stored into a FIFO buffer which is dedicated to the specific output port[6]. A cell will be lost if it arrives to find the shared buffer full or the address buffer 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 queue).

We model this system by the continuous time queueing model shown in figure 1. The queueing model consists of n single server queues, where n is the number of the input (or output) ports of the switch. Each server represents an output port. An incoming cell destined to output port i joins the ith queue. The total number of cells in all queues can not exceed M, the buffer size of the switch. A cell gets lost if it arrives at a time when the switch is full. 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 study, we assume that M_i=M. However, the case where $M_i < M$ can be easily implemented.

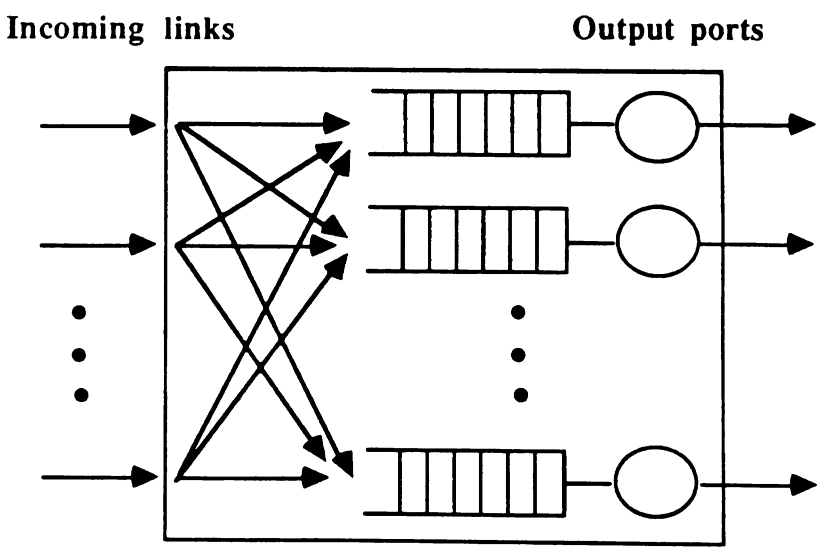

Figure 1: The queueing model

In the real system, the transmission time at each output port is constant. In view of this, we model the service time at a server by an Erlang distribution with r stages. All servers have the same service time distribution. Let μ be the service rate at one exponential stage. The arrival process to each input queue is assumed to be bursty and it is modelled by an Interrupted Poisson Process(IPP). That is, two separate exponentially distributed periods, an active period and a silence period, occur alternatively. During the active period, arrivals occur in a Poisson fashion. It can be shown that the time between two successive arrivals has a hyperexponential distribution with two stages, which in turn is equivalent to a Coxian distribution with two stages (hereafter referred to as C_2)[7]. In view of this, we will assume that the arrival process to the ith input port is described by a C_2 distribution with parameters $\lambda_{i,1}$,$\lambda_{i,2}$, and a_i for i=1,2,...,n, where a_i is the probability of going from stage 1 to stage 2. When a cell arrives at the ith input port, it chooses queue j with probability $p_{i,j}$, where $\sum_j p_{i,j} = 1$, i=1,2,...n.

3. The Approximation Algorithm

The approximation algorithm described in this paper is based on the notion of aggregation. For other applications of the aggregation technique to queueing systems with dedicated buffers or with shared buffers, the reader is refered to [2], [8], [9], [10], and [11].

The state of our model is described by the vector ($\underline{w}$,$\underline{m}$)=$(w_1, ..., w_n; m_1, ..., m_n)$, where w_i, i=1,2,...,n, is the stages of the arrival process to the ith input port (w_i=1 or 2) and m_j, j=1,2,...,n, is the total number of stages to be completed by the cells in the jth queue. Let K be the total number of cells in the whole switch, and let k_j, j=1,...,n, be the number of cells in the jth queue. Then, we have

$$K = \sum_{j=1}^{n} k_j = \sum_{j=1}^{n} \lceil \frac{m_i}{r} \rceil,$$

where $\lceil a \rceil$ is the least integer which is greater than or equal to a.

We now proceed to obtain the global balance equations for the above model. In order to simplify the notation, let us consider the following pseudo-arrival rates and pseudo-service rates:

$$\lambda_{i,1}(K) \equiv \begin{cases} \lambda_{i,1}, & 0 \le K \le M-1 \\ 0, & K = M \end{cases}$$

$$\lambda_{i,2}(K) \equiv \begin{cases} \lambda_{i,2}, & K = M \\ 0, & 0 \le K \le M-1 \end{cases}$$

$$\mu_j(m_j) \equiv \begin{cases} \mu, & 1 \le m_j \le rM \\ 0, & m_j = 0. \end{cases}$$

Let P($\underline{w}$,$\underline{m}$) be the steady state probability of the state ($\underline{w}$,$\underline{m}$). Furthermore, for given $\underline{w}$ let W_1 and W_2 be the set of input ports which are in phase 1 and 2 respectively. The global balance equations for the set ($\underline{w}$, $\underline{m}$) are as follows:

$$\Big[\sum_{i\in W_1} \{a_i\lambda_{i,1} + (1-a_i)\lambda_{i,1}(K)\} + \sum_{i\in W_2} \lambda_{i,2} + \sum_{j=1}^{n} \mu_j(m_j)\Big] P(\underline{w},\underline{m})$$
$$= \sum_{i\in W_2} a_i\lambda_{i,1} P(\underline{w} - \underline{e}_i^n, \underline{m})$$

$$+ \sum_{i \in W_1} \sum_{j=1}^{n} p_{i,j}(1-a_i)\lambda_{i,1} P(\underline{w}, \underline{m} - r\underline{e_j^n}) \quad (1)$$
$$+ \sum_{i \in W_1} \sum_{j=1}^{n} p_{i,j}\lambda_{i,2} P(\underline{w} + \underline{e_i^n}, \underline{m} - r\underline{e_j^n})$$
$$+ \sum_{i \in W_1} \lambda_{i,2}(K) P(\underline{w} + \underline{e_i^n}, \underline{m})$$
$$+ \sum_{j=1}^{n} \mu P(\underline{w}, \underline{m} + \underline{e_j^n}),$$

where $\underline{e_i^n}$ is the n-dimensional unit vector whose ith component equals to 1.

Note that with the exception of very simple cases, the solution of the system of linear equations (1) becomes intractable due to the very large number of states. In view of this, we analyze the queueing model under study approximately by analyzing each queue in isolation from the remaining queues. Let us focus on a particular queue j. The set of the remaining queues will be referred to by the symbol n-{j}. In order to analyze queue j in isolation it is necessary to keep track of the number of stages left to be completed in the queue. Also, it is necessary to keep track of the total number of customers K in the whole system, so that one can decide whether an arriving cell can be admitted to the switch or not. However, in order to keep track of how K changes, it is necessary to know the rate at which cells depart from n-{j}. This departure rate is calculated using function $f_j(K - k_j)$ of the mean number of busy queues in n-{j}, where k_j is the number of cells in queue j. This function is not known, and it is approximated iteratively as it will be described below.

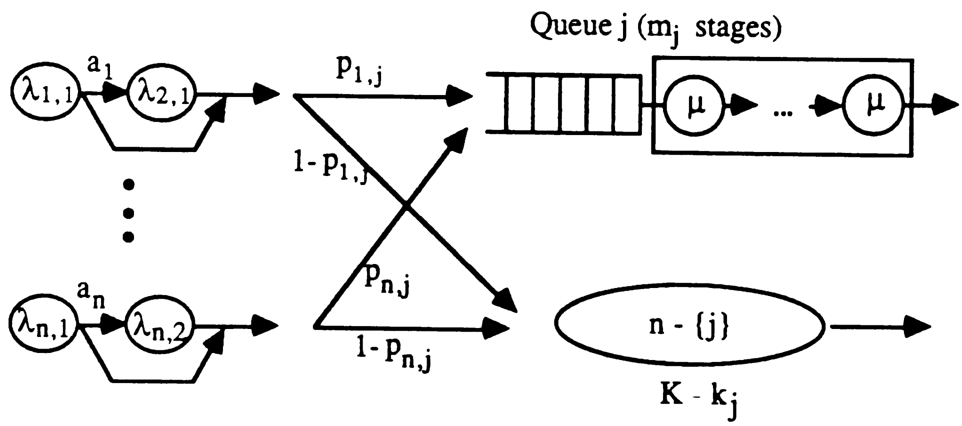

Figure 2: Queueing structure of sub-system for queue j (before aggregating the n arrival processes)

Thus, the switch is decomposed into n sub-systems, one per queue, as shown in figure 2. Each sub-system is analyzed numerically by setting up its rate matrix and subsequently obtaining its stationary probability vector using equation solving techniques, such as the successive over-relaxation method(SOR). The state space of each sub-system is very large due to the fact that each of the n arrival processes is represented by a C_2. In order to reduce the dimensionality of each sub-system, we aggregate the n arrival processes into a single variable x which simply gives the number of the arrival processes which are in stage 2. As was mentioned above, the state of the n arrival processes is described by the vector $(w_1, w_2, ..., w_n)$. Due to the fact that each arrival process is independent from the remaining processes, we have that $p(\underline{w})=p(w_1)p(w_2)\cdots p(w_n)$, where $p(w_i)$ is simply the probability that the ith arrival process is in stage w_i. In view of this, $p(\underline{w})$ can be easily obtained. Now, using the stationary probability vector $p(\underline{w})$, we can easily aggregate the Markov chain of the n arrival processes to a Markov chain that gives all the transitions of variable x. In this way, we can significantly reduce the dimensionality of each sub-system shown in figure 2. Below, we present this approximation method in a more formal way.

Let S_x be the set of states $\underline{w}=(w_1, ..., w_n)$, where for each state there are exactly x arrival processes which are in phase 2. As discussed above, we analyze queue j approximately by keeping track of the number of input arrival processes which are in phase 2 (x), the number of stages to be completed by the cells in the jth queue (m_j), and the total number of cells in the whole switch (K). Denote by $P_j(x, m_j, K)$ the stationary joint probability distribution of the state (x, m_j, K). $P_j(x, m_j, K)$ is a marginal probability derived from $P(\underline{w}, \underline{m})$ by

$$P_j(x, m_j, K) = \sum_{\underline{w}\in S_x} \sum_{\sum_{l=1}^{n} \lceil \frac{m_l}{r} \rceil = K} P(\underline{w}, \underline{m}).$$

Performing this summation on equation (1), we get the aggregate balance equation as follows:

$$\begin{aligned}
&[\sum_{\underline{w}\in S_x} Prob\{\underline{w}|x, m_j, K\}\{\sum_{i\in W_1}(a_i\lambda_{i,1} + (1-a_i)\lambda_{i,1}(K)) + \sum_{i\in W_2}\lambda_{i,2}\} \\
+ \ & \mu(m_j) + \mu \sum_{l=1, l\neq j}^{n} \sum_{t=1}^{K-\lceil \frac{m_j}{r} \rceil - 1} Prob\{m_l = rt+1|x, m_j, k\}]P_j(x, m_j, K) \\
= \ & \sum_{\underline{w}\in S_{x-1}} Prob\{\underline{w}|x-1, m_j, K\} \sum_{i\in W_1} a_i\lambda_{i,1} P_j(x-1, m_j, K) \\
+ \ & \sum_{\underline{w}\in S_x} Prob\{\underline{w}|x, m_j - r, K-1\} \sum_{i\in W_1} P_{i,j}(1-a_i)\lambda_{i,1} P_j(x, m_j - r, K-1) \\
+ \ & \sum_{\underline{w}\in S_x} Prob\{\underline{w}|x, m_j, K-1\} \sum_{i\in W_1}\sum_{l=1, l\neq j}^{n} P_{i,l}(1-a_i)\lambda_{i,1} P_j(x, m_j, K-1) \\
+ \ & \sum_{\underline{w}\in S_{x+1}} Prob\{\underline{w}|x+1, m_j - r, K-1\} \sum_{i\in W_2} P_{i,j}\lambda_{i,2} P_j(x+1, m_j - r, K-1) \\
+ \ & \sum_{\underline{w}\in S_{x+1}} Prob\{\underline{w}|x+1, m_j, K-1\} \sum_{i\in W_2}\sum_{l=1, l\neq j}^{n} P_{i,l}\lambda_{i,2} P_j(x+1, m_j, K-1) \\
+ \ & \sum_{\underline{w}\in S_{x+1}} Prob\{\underline{w}|x+1, m_j, K\} \sum_{i\in W_2} \lambda_{i,2}(K) P_j(x+1, m_j, K) \\
+ \ & \mu P_j(x, m_j + 1, K + \nu_{m_j+1}) \\
+ \ & \mu \sum_{l=1, l\neq j}^{n} \sum_{t=1}^{K-\lceil \frac{m_j}{r} \rceil} Prob(m_\ell = rt+1|x, m_j, K+1) P_j(x, m_j, K+1),
\end{aligned} \tag{2}$$

for x=0,1,...,n, m_j=0,1,...,rM, and K=0,1,...,M, where

$$\nu_{m_j+1} = \begin{cases} 1, & \lceil \frac{m_j}{r} \rceil = \frac{m_j}{r} \\ 0, & \lceil \frac{m_j}{r} \rceil > \frac{m_j}{r}. \end{cases}$$

Equation (2) is a set of linear equations with (n+1)(M+1)(Mr+2)/2 unknown probabilities $P_j(x, m_j, K)$. We note that these equations are exact if the correct values of the aggregate transition rates are used. However these aggregate transition rates contain two types of unknown conditional probabilities, namely $Prob\{\underline{w}|x, m_j, K\}$ and $Prob\{m_l = rt + 1|x, m_j, K\}$. These probabilities are approximated as follows.

To begin with, we assume that the states of the input arrival processes do not depend on the total number of stages to be completed by a cell in service in the jth queue or on the total number of cells in the whole switch, so that

$$\begin{aligned} Prob\{\underline{w}|x, m_j, k\} &\simeq Prob\{\underline{w}|x\} \\ &= \frac{P(\underline{w})}{P(S_x)}, \end{aligned} \tag{3}$$

where

$$P(S_x) = \sum_{\underline{w} \in S_x} P(\underline{w})$$

and as reported in [7],

$$P(\underline{w}) = [\prod_{i \in W_1} \frac{\lambda_{i,2}}{a_i \lambda_{i,1} + \lambda i, 2}][\prod_{i \in W_2} \frac{\lambda_{i,1}}{a_i \lambda_{i,1} + \lambda i, 2}].$$

In order to estimate the other type of conditional probabilities, we assume that a cell in queue $l(l \neq$ j) which is in service has the same probability to be in any of the r Erlang stages. Moreover, the probability that a queue $l(l \neq$ j) is empty is assumed to depend on the total number of cells in n-{j}, i.e,

$$\begin{aligned} &\sum_{l=1, l \neq j}^{n} \sum_{t=1}^{K - \lceil \frac{m_j}{r} \rceil - 1} Prob\{m_l = rt + 1|x, m_j, K\} \\ \simeq &\sum_{l=1, l \neq j}^{n} \frac{1 - Prob\{k_l = 0|x, m_j, K\}}{r} \\ \simeq &\sum_{j=1, l \neq j}^{n} \frac{1 - Prob\{k_l = 0|K - k_j\}}{r} \\ \equiv &f_j(K - k_j)/r \end{aligned} \tag{4}$$

Here $f_j(K - k_j)$ is the mean number of non-empty queues in n-{j}, given that the number of cells in n-{j} is $K - k_j$, where k_j is the number of cells in the ith queue. $f_j(\alpha)$ satisfies the following conditions.

1. $f_j(1) = 1$
2. $f_j(\alpha + 1) > f_j(\alpha)$, α=1,2,...,M

3. $f_j(\alpha+1) - f_j(\alpha) > f_j(\alpha+2) - f_j(\alpha+1)$

4. $f_j(\alpha) < min[n-1, \alpha]$

$f_j(\alpha)$ is not known and it is calculated as follows. Let $g_j(\alpha)$ be the mean number of non-empty queues in n-{j} when α cells arrive at the switch assuming that the service time at each queue is infinite. Then we have

$$g_j(\alpha) = \sum_{l=1, l\neq j}^{n} [1 - (\frac{\sum_{t=1,t\neq j,l}^{n} \lambda_t}{\sum_{t=1,t\neq j}^{n} \lambda_t})^{\alpha}]. \tag{5}$$

Clearly $1 \leq f_j(\alpha) \leq g_j(\alpha)$, and therefore $f_j(\alpha)$ can be approximated as

$$f_j(\alpha,\beta) = 1 + \beta[g_j(\alpha) - 1], \tag{6}$$

where

$$0 \leq \beta \leq 1.$$

Note that $g_j(\alpha)$ satisfies the above four conditions, and so does $f_j(\alpha,\beta)$. We estimate $f_i(\alpha)$ iteratively by adjusting β up and down accordingly in order to meet the convergence criterion. Several different convergence criteria which must always hold can be condidered. In this study, the mean number of cells in the switch is used because it is computationally simple. The mean number of cells in the switch can be computed in the following two different ways:

$$E_1 = \sum_{j=1}^{n} \sum_{k_j=1}^{M} k_j P_j(k_j) \tag{7}$$

$$E_2 = \frac{1}{n} \sum_{j=1}^{n} \sum_{K=1}^{M} K P_j(K) \tag{8}$$

where

$$P_j(k_j) = \sum_{x=0}^{n} \sum_{m_j=rk_j-r+1}^{rk_j} \sum_{K=k_j}^{M} P_j(x, m_j, K),$$

and

$$P_j(K) = \sum_{x=0}^{n} K \sum_{m_j=0}^{Kr} P_j(x, m_j, K).$$

E_1 must be equal to E_2. Moreover, $E_1 - E_2$ is a monotonically increasing function of β as well as of $f_j(\alpha)$. We can, thus, decide whether the $f_j(\alpha)$ is overestimated or underestimated depending on which mean is larger, and accordingly change $f_j(\alpha)$, in order to reduce the difference E_1 and E_2. Below, we summarize the proposed approximation algorithm. The superscript s is used in order to denote an iteration number.

Algorithm

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

step 1: s=s+1, $\beta_M^s = \frac{\beta_H^{s-1}+\beta_L^{s-1}}{2}$ and calculate $f_j^s(\alpha,\beta_M^s)$ for α=1,...,M and j=1,2,...,n by using (5) and (6).
step 2: Calculate conditonal probabilities by using (3) and (4), and then obtain $P_j^s(x,m_j,K)$ for j=1,2...,n by solving numerically (2). This numerical solution is obtained by first setting up the underlying rate matrix and subsequently calculating the stationary vector of $P_j^s(x,m_j,K)$ using the SOR method.
step 3: Calculate E_1^s and E_2^s by using (7) and (8) respectively.
step 4: 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.

4. Validation

The approximation algorithm was implemented on a Cray super computer and it was used to analyze an 8×8 shared buffer switch. The approximation results were compared against data obtained from a simulation model. Representative results are summarized in tables 1 to 5. Each table gives approximate and simulation results for the global queue-length distribution P(K), K=0,1,...,M, and for the queue-length distribution $P_1(k_1)$,k_1=0,1...,M of queue 1, which is the most heavily utilized queue. Approximate and simulation results for the mean number of cells, m.q.l., in the switch are also given. The absolute errors for P(K) and $P_1(k_1)$ given in tables 1 to 5 have been plotted in figures 3 to 7 respectively. The results have been obtained, assuming the following values for the 8 arrival processes.

Arrival process	$\lambda_{i,1}$	$\lambda_{i,2}$	a	C^2
1	1.0025	0.0025	0.0025	200
2	1.0051	0.0050	0.0051	100
3	1.0064	0.0063	0.0063	80
4	1.0103	0.0101	0.0102	50
5	1.0130	0.0127	0.0128	40
6	1.0270	0.0256	0.0263	20
7	1.0586	0.0525	0.0554	10
8	1.7071	0.2929	0.4142	2

These values were obtained by assuming that each arrival process has an average arrival rate of 0.5, a peak arrival rate of 1, and a squared coefficient of variation of the interarrival time C^2 as shown in the table above. The branching probabilities were as follows: $p_{i,1}$=0.25, $p_{i,2}$=0.2, $p_{i,3}$=0.18, $p_{i,4}$ = 0.12, $p_{i,5}$=0.1, $p_{i,6}$=0.08, $p_{i,7}$=0.05, $p_{i,8}$=0.02, i=1,2,...,8. The service time distribution was assumed to be an E_3. The results in tables 1 to 5 were obtained by varying the total buffer space, M, and the service rate.

In general, the approximation algorithm gives good results. The approximate results for the queue-length distribution $P_i(k_i)$ for each queue i seem to be slightly more accurate than the global queue-length distribution P(K). The approximate results are not as good when the utilization of the output ports is very high (for instance, when the utilization of the output port 1, which is the most heavily utilized, is more than 85%). However, such high utilization is not likely to be encountered in real- life systems.

5. Conclusion

In this paper we presented an approximation algorithm for the analysis of the shared buffer ATM switch architecture under the assumption of bursty arrivals. Comparisons with simulation results showed that the approximation algorithm has good accuracy. This algorithm can be extended to the case where the bursty arrivals are also correlated. Another possible extension is the analysis of the discrete-time version of the queueing model studied in this paper. These two extensions will be considered in our future work.

References

[1] P. Boyer, M. Lehnert and P. Kuhn, Queueing in an ATM basic switch element, Technical Report, CNET-123-030-CD-CC, CNET (1988).

[2] A. Brandwajn and Y.-L.Jow, An Approximation Method for Tandem Queues with Blocking, Oper. Res. 36 (1988) 73-83.

[3] M. Devault, J. Cochennec, and M. Servel, The "Prelude" ATD Experiment: Assessments and Future Prospects, IEEE J. SAC, 6 (1988) 1528-1537.

[4] M. Hluchyj and M. Karol, Queueing in High-Performance Packet Switching, IEEE J. SAC, 6 (1988) 1587-1597.

[5] H. Kuwahara, N. Endo, M. Ogino and T. Kozaki, A Shared Buffer Memory Switch for an ATM Exchange, Proc. ICC'89, (1989), 118-122.

[6] H. Lee, K. Kook, C. Rim, K. Jun, and S. Lim, A Limited Shared Output Buffer Switch for ATM, Fourth Int. Conf. on Data Communication Systems and Their Performance, Barcelona, June (1990), 163-179.

[7] 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, Barcelona, June (1990), 341-357.

[8] P.J. Schweitzer and T. Altiok, Aggregate Modelling of Tandem Queues without Intermediate Buffers, in: Perros and Altiok, eds., *Queueing Networks with Blocking* (North Holland, 1989) 47-72.

[9] Y. Takahashi, A New Type Aggregation Method for Large Markov Chains and Its Application to Queueing Networks, in: Akiyama, ed., *Teletraffic Issues in an Advanced Information Society, ITC-11* (Elsevier,1985) 490-494.

[10] Y. Takahashi, Aggregate Approximation for Acyclic Queueing Networks with Communication Blocking, in: Perros and Altiok, eds., *Queueing Networks with Blocking*(North Holland, 1989) 33-46.

[11] H.Yamashita, An Approximation Method for Queueing Network with a Common Buffer and Blocking, Technical Report, Mechanical Engineering Dept., Sophia Univ., Tokyo, Japan, (1989).

K	P(K) approx.	P(K) simul.	k_1	$P_1(k_1)$ approx.	$P_1(k_1)$ simul.
0	.1363	.1252±.0032	0	.5202	.5116±.0042
1	.1983	.1922±.0024	1	.2748	.2724±.0013
2	.1983	.1949±.0013	2	.1219	.1233±.0015
3	.1634	.1626±.0011	3	.0520	.0548±.0011
4	.1195	.1219±.0013	4	.0210	.0239±.0007
5	.0808	.0859±.0014	5	.0075	.0098±.0004
6	.0517	.0579±.0012	6	.0021	.0033±.0002
7	.0320	.0372±.0010	7	.0004	.0009±.0001
8	.0199	.0221±.0007	8	.0	.0001±.0
m.q.l	2.68	2.77			

Table 1: Approximate and simulation results for P(K) and $P_1(k_1)$ buffer size=8, service rate=2.0

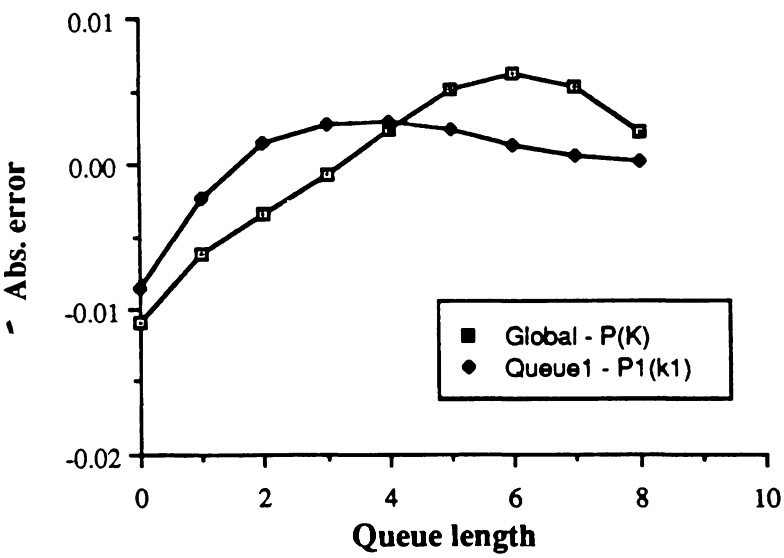

Figure 3: Absolute errors for the results in table 1

K	P(K)		k_1	$P_1(k_1)$	
	approx.	simul.		approx.	simul.
0	.1299	.1273±.0029	0	.5004	.5010±.0039
1	.1860	.1903±.0024	1	.2641	.2623±.0009
2	.1874	.1888±.0012	2	.1206	.1184±.0011
3	.1572	.1539±.0008	3	.0564	.0552±.0009
4	.1178	.1128±.0009	4	.0278	.0275±.0007
5	.0818	.0776±.0010	5	.0143	.0147±.0006
6	.0539	.0517±.0009	6	.0076	.0084±.0005
7	.0341	.0338±.0008	7	.0042	.0050±.0003
8	.0211	.0219±.0007	8	.0023	.0031±.0003
9	.0128	.0144±.0006	9	.0012	.0020±.0002
10	.0076	.0096±.0004	10	.0006	.0012±.0001
11	.0045	.0064±.0003	11	.0003	.0007±.0001
12	.0026	.0044±.0003	12	.0001	.0004±.0001
13	.0015	.0030±.0002	13	.0001	.0002±.0
14	.0009	.0021±.0001	14	.0	.0001±.0
15	.0005	.0014±.0001	15	.0	.0 ±.0
16	.0003	.0009±.0001	16	.0	.0 ±.0
m.q.l	2.99	3.04			

Table 2: Approximate and simulation results for P(K) and $P_1(k_1)$ buffer size=16, service rate=2.0

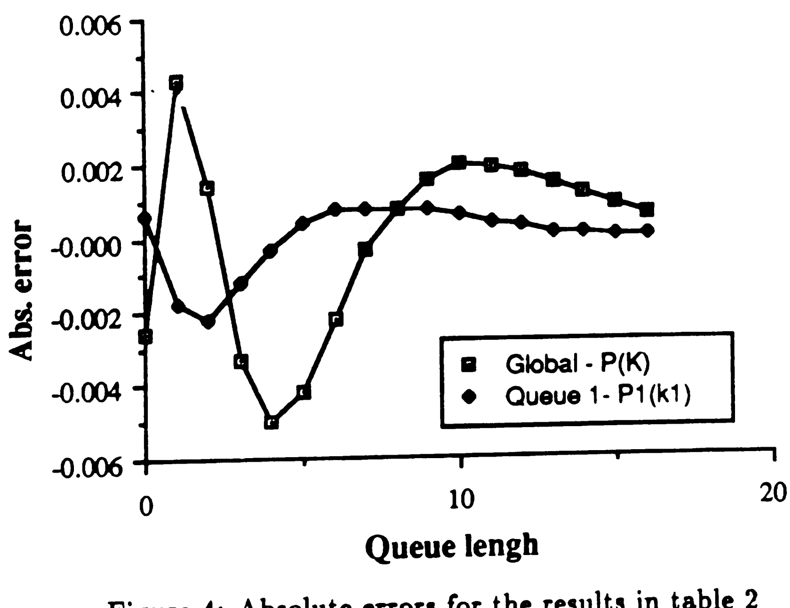

Figure 4: Absolute errors for the results in table 2

K	P(K)		k_1	$P_1(k_1)$	
	approx.	simul.		approx.	simul.
0	.0445	.0478±.0025	0	.2719	.2719±.0055
1	.0704	.0792±.0026	1	.2033	.1954±.0021
2	.0879	.0944±.0020	2	.1389	.1291±.0011
3	.0956	.0955±.0013	3	.0995	.0905±.0013
4	.0956	.0892±.0008	4	.0752	.0677±.0012
5	.0906	.0799±.0008	5	.0587	.0538±.0012
6	.0827	.0705±.0009	6	.0461	.0444±.0011
7	.0737	.0620±.0009	7	.0357	.0372±.0008
8	.0647	.0548±.0008	8	.0267	.0308±.0009
9	.0563	.0490±.0008	9	.0188	.0256±.0009
10	.0488	.0447±.0007	10	.0123	.0201±.0009
11	.0423	.0418±.0008	11	.0071	.0149±.0008
12	.0368	.0403±.0007	12	.0036	.0097±.0007
13	.0322	.0396±.0008	13	.0015	.0055±.0004
14	.0284	.0393±.0010	14	.0005	.0025±.0002
15	.0256	.0380±.0011	15	.0001	.0008±.0001
16	.0238	.0338±.0012	16	.0	.0001±.0
m.q.l	6.42	6.62			

Table 3: Approximate and simulation results for P(K) and $P_1(k_1)$ buffer size=16, service rate=1.3

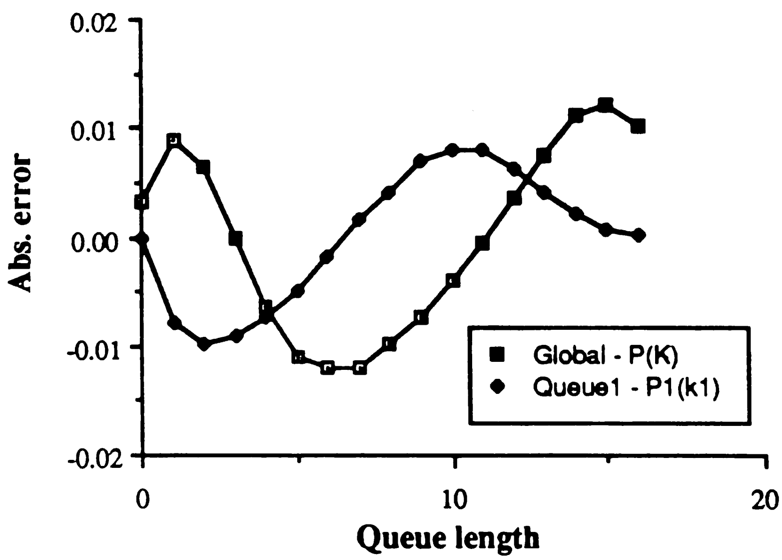

Figure 5: Absolute errors for the results in table 3

K	P(K) approx.	P(K) simul.	k_1	$P_1(k_1)$ approx.	$P_1(k_1)$ simul.
0	.0638	.0638±.0020	0	.3453	.3442±.0044
1	.1030	.1078±.0021	1	.2378	.2304±.0014
2	.1239	.1263±.0017	2	.1458	.1377±.0011
3	.1271	.1233±.0013	3	.0920	.0860±.0011
4	.1182	.1094±.0009	4	.0607	.0573±.0010
5	.1028	.0920±.0007	5	.0411	.0407±.0008
6	.0854	.0754±.0007	6	.0282	.0297±.0008
7	.0686	.0610±.0008	7	.0191	.0224±.0007
8	.0539	.0494±.0008	8	.0127	.0170±.0007
9	.0416	.0405±.0008	9	.0081	.0124±.0006
10	.0318	.0334±.0008	10	.0048	.0089±.0005
11	.0240	.0281±.0007	11	.0026	.0061±.0004
12	.0181	.0239±.0007	12	.0012	.0039±.0003
13	.0136	.0205±.0006	13	.0005	.0021±.0002
14	.0102	.0179±.0006	14	.0002	.0010±.0001
15	.0778	.0152±.0006	15	.0	.0003±.0001
16	.0061	.0120±.0005	16	.0	.0 ±.0
m.q.l	4.92	5.14			

Table 4: Approximate and simulation results for P(K) and $P_1(k_1)$
buffer size=16, service rate=1.5

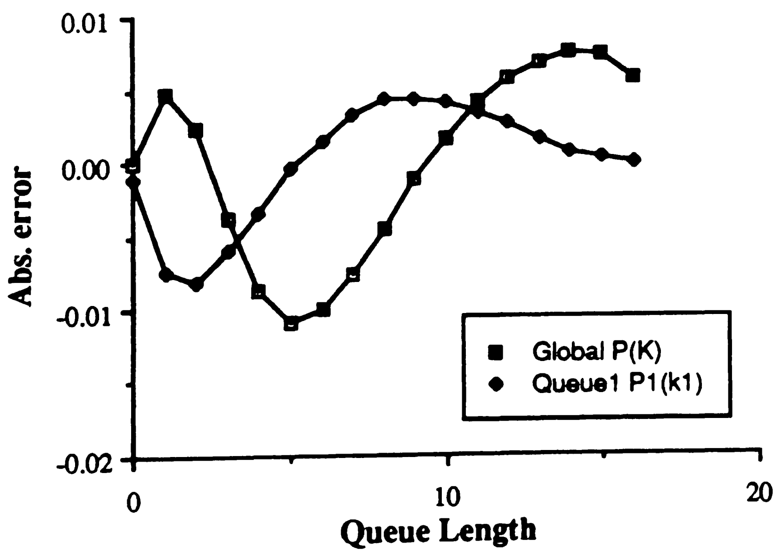

Figure 6: Absolute errors for the results in table 4

K	P(K) approx.	P(K) simul.	k_1	$P_1(k_1)$ approx.	$P_1(k_1)$ simul.
0	.0871	.0864±.0028	0	.4157	.4129±.0056
1	.1337	.1416±.0028	1	.2556	.2513±.0016
2	.1504	.1558±.0020	2	.1383	.1341±.0011
3	.1435	.1421±.0013	3	.0769	.0742±.0011
4	.1236	.1168±.0007	4	.0449	.0445±.0010
5	.0994	.0911±.0009	5	.0272	.0279±.0009
6	.0761	.0689±.0010	6	.0168	.0185±.0008
7	.0563	.0514±.0011	7	.0104	.0126±.0007
8	.0406	.0383±.0011	8	.0064	.0087±.0005
9	.0288	.0285±.0009	9	.0038	.0060±.0004
10	.0202	.0216±.0008	10	.0021	.0040±.0004
11	.0140	.0165±.0007	11	.0011	.0026±.0003
12	.0096	.0127±.0006	12	.0005	.0015±.0002
13	.0066	.0100±.0006	13	.0002	.0007±.0001
14	.0046	.0079±.0005	14	.0001	.0003±.0001
15	.0031	.0061±.0004	15	.0	.0001±.0
16	.0022	.0043±.0003	16	.0	.0 ±.0
m.q.l	4.00	4.09			

Table 5: Approximate and simulation results for P(K) and $P_1(k_1)$ buffer size=16, service rate=1.7

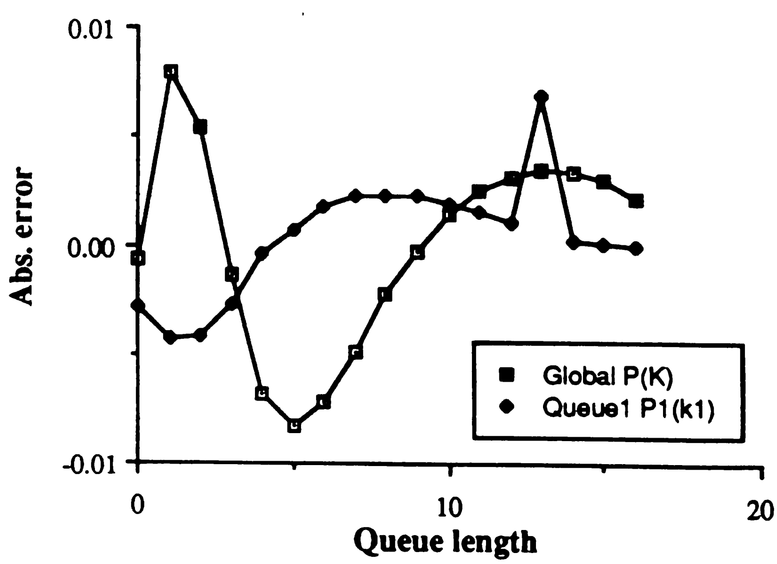

Figure 7: Absolute errors for the results in table 5