14

An integrated approach to evaluating the loss performance of ATM switches

S. Montagna
Italtel - A STET and Siemens Company, Milan, Italy
tel. 39 2 4388 8098, fax. 39 2 4388 7989
e.mail Montagna@settimo.italtel.it

R. Paglino
Italtel - A STET and Siemens Company, Milan, Italy

J. F. Meyer
Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan, USA.

Abstract
This work addresses model-based evaluation of cell loss probabilities for an ATM switching element with a shared output buffer. The incoming traffic to the switch is represented by the superposition of N bursty input sources, each of which is modeled as a two-state (On/Off) Markov chain. For such systems, we consider an integrated approach to their evaluation that employs both exact and approximate solutions. The exact method is based on a reduced Markov model obtained by lumping the states according to certain symmetries of the traffic model. However, even with such reduction, numerical solutions are feasible only if the switch dimensions involved, particularly the number of output ports, are reasonably small. We then introduce a new approximate solution algorithm that can be applied to larger switches. By comparing the results obtained with those of the exact method, we find that the errors of approximation are relatively small. Moreover, due to the iterative nature of the approximate solution algorithm, the two methods can be integrated so as to yield even more accurate results with less execution time.

Keywords
ATM switch, shared buffers, On/Off sources

1 INTRODUCTION

Services both realized and planned for broadband, ATM-based, ISDNs impose extremely severe constraints on the performance of ATM switching elements. In particular, admissible cell loss probabilities as small as 10^{-9} (or even less) call for switch buffers that are sufficiently large to guarantee this quality of service. In this regard, it has been shown (see [1,2], for example) that the best utilization of buffer capacity is obtained by dynamically sharing cell storage among all the output ports of the switch. This permits a reduction of required capacity (for a specified admissible cell loss probability) relative to switches which employ dedicated, fixed-capacity queues at either the input or the output. However, the problem of evaluating the loss performance of a shared-buffer switch is difficult, due primarily to the large number of internal states that must be accounted for in the process, even when the switch dimensions are relatively small. Therefore, various studies have proposed approximate solutions to this problem, assuming further (see [3,4,5], for example) that traffic sources for the input ports are represented by independent Bernoulli arrival processes, thus precluding any correlation between cell arrivals. Among such investigations, perhaps the most widely cited is [3] which presumes an infinite buffer and approximates its steady-state occupancy distribution with a Gamma function. The parameters of the Gamma distribution are obtained analytically by computing the mean and variance of the shared-buffer occupancy distribution. This method provides a practical means of quickly estimating the required buffer capacity of a switch. However, since it matches only the first two moments of the actual distribution, it often fails to accurately estimate the distribution's "tail", i.e., the probabilities of large occupancies which have very low values.

Another simple way to estimate the buffer occupancy distribution of a shared buffer with uncorrelated traffic is by convolving the individual distributions of a number of Geo/D/1 queues. Since Geo/D/1 models are relatively easy to solve (as discussed in [6], for example), this method is also attractive. Other studies, such as those of [7,4], suggest more complex heuristic algorithms that typically lead to more precise solutions.

Although Bernoulli sources are convenient by virtue of their simplicity, a more realistic arrival process should capture correlation that exists between successive arrivals at an input port. This is done in [8], for example, by employing a continuous-time model where each input source is modeled by an interrupted Poisson process. The investigation that follows considers two discrete-time models of a shared-buffer switch subjected to bursty (and hence correlated) traffic. They support exact and approximate solution algorithms, respectively; moreover, we find that these methods can be usefully integrated to achieve both greater accuracy and reduced execution time (when compared with exclusive use of the approximate method).

The first method, referred to as *Algorithm 1*, provides an exact solution of the steady-state distribution of shared-buffer occupancy for switches of limited (but not trivial) size. This solution is based on an efficient representation of the state space that derives from certain symmetries implied by the underlying assumptions. Although its application is limited in the sense noted above (its execution time grows exponentially with buffer capacity), it is nevertheless very useful. In particular, in addition to providing exact results for the probabilities of rare cell-loss events, it can serve as a reference for assessing the nature and magnitude of errors that result from approximate analytic models and/or solution techniques. Although simulation is often used for this purpose, such practice is reasonable only if the simulation results are themselves highly accurate (high confidence with respect to small confidence intervals).

Further, as we emphasize in the development that follows, it is sometimes possible to integrate the use of exact solutions with certain types of approximation techniques, thereby extending the scope of the former. For example, if the approximation algorithm is iterative in nature (as in the case of convolution, or more specifically, the algorithm we consider below) then an exact solution can be usefully employed for the initial iteration. This leads to more accurate approximate evaluations, even for realistically large switches with bursty traffic.

The approximation technique we propose is new (*Algorithm* 2) and is based on a decomposition of the system into smaller systems involving fewer output ports. Comparisons (see section 4) of algorithm-2 results with those of algorithm 1 (for small switch sizes and very low loss requirements) and with simulation data (for larger systems with relatively high losses) reveal that the approximations obtained are reasonably accurate. These results are then used to estimate the advantage, in terms of memory saving, of a shared-buffer architecture relative to a simpler architecture that employs a dedicated, fixed-capacity buffer for each output port. With such estimations, the required shared-buffer capacities for very low admissible cell loss probabilities can be likewise estimated.

Assumptions concerning the switch and its traffic are discussed in section 2. This is followed by descriptions of the two algorithms, including their integration (section 3) and, in turn, a presentation of the results just mentioned (section 4). Section 5 then summarizes what was accomplished, with appendices A and B providing some solution details that were omitted in section 3.

2 THE SWITCH

The switch considered has a typical shared-buffer architecture, i.e., memory space available to store ATM cells is dynamically shared among all the output queues. Incoming cells arrive from N input ports and are addressed to one of R output ports. Provided there is available space in a common buffer of finite capacity K (the maximum number of cells that can be stored), via an appropriate pointer structure (maintained in a separate memory space), a cell is then stored in a logical FIFO output queue corresponding to its address. A cell is lost if and only if no buffer space is available when the cell arrives. The switch is assumed to operate synchronously at the cell level; in a given time slot (the time required to completely transmit/receive a cell on a port of the switch), we presume that the following two operations take place in the order indicated.

Send: For each non-empty logical queue, the least recently arrived cell cell is served and its buffer space is freed.
Receive: Each incoming cell is stored in the buffer (if there is available space) and the pointer chain is appropriately updated; these cells will be served in the next slot.

The traffic at each of the N input ports is represented by a 2-state (On/Off) Markov chain where these individual sources are assumed to be statistically independent. In the On state, a cell arrives with probability 1 while in the Off state there are no arrivals. The dwell times in each state (number of time slots between entry and exit) are geometrically distributed variables, with means L and I for the On and the Off states, respectively.

The activity ρ_{in} of an individual source is the fraction of time the source is in the On state and is given by $\rho_{in} = L/(I+L)$. The destination address of each cell (i.e. the output port it is queued to) is a random variable that's uniformly distributed over the R output ports and is independent of the destinations of previously arrived cells. This assumption attempts to capture the situation where each input link carries the superposition of a large number of low bit-rate connections, each connection addressed to a possibly different output link.

As is well known, a purely random (memoryless) traffic model, where each input behaves as a Bernoulli source, is a special case of the model just described. Specifically, the above reduces to the Bernoulli case if $L = 1/(1 - \rho_{in})$ and $I = 1/\rho_{in}$. Finally, we let ρ denote the offered load, as reflected by the utilization of an output port (assuming no cell losses), i.e., $\rho = N{\cdot}\rho_{in} / R$.

3 THE ALGORITHMS

As mentioned in our introductory remarks, we choose to employ both exact and approximate model-based methods to determine the steady-state probability distribution of shared-buffer occupancy, given the switch/traffic assumptions stated above. (Other measures, such as loss probability are then based on this distribution.) These are described in the subsections that follow, with some of the mathematical details being deferred to appendix A (algorithm 1) and appendix B (algorithm 2). However, before proceeding with these descriptions, it is helpful to introduce some assumptions, terminology, and notation which are common to both algorithms.

Time is assumed to be discrete, where a time instant t takes values in the set $T = \{0, 1, 2, \dots\}$. The duration between successive instants t and $t + 1$ is interpreted as the t th time slot, where the enumeration begins with time slot 0 and instant t represents the beginning of slot t, i.e., it occurs before the intraslot "send" and "receive" operations described in section 2.

M_t = number of sources in the On state during slot t

$X_{i,t}$ = number of cells in the buffer at time t addressed to output port i, $i = 1, 2, \dots, R$

Y_t = $\sum_{i=1}^{R} X_{i,t}$ = number of cells in the buffer at time t.

By these definitions, along with our earlier assumptions concerning switch dimensions N and K, for any $t \in T$, these variables are thus constrained to have integer values in the ranges

$$0 \le M_t \le N \tag{1}$$

$$0 \le X_{i,t} \le K; \quad i = 1, 2, \dots, R \tag{2}$$

$$0 \le Y_t \le K \tag{3}$$

Since there is an arrival at an input port if and only if the source for that port is in the On state, M_t is just the number of arrivals during slot t, including some that may be lost if the buffer is full. However, if the buffer capacity is at least N (which we tacitly assume throughout the discussion) then, for all $t \in T$,

$$M_t \leq Y_t \tag{4}$$

3.1 Algorithm 1 (Exact)

Let $X_t = (X_{1,t}, X_{2,t}, \ldots, X_{R,t})$ be the R-dimensional vector-valued random variable that represents the cell-occupancy of the shared buffer at time t. If, further, we let X denote the corresponding stochastic process, i.e., $X = \{ X_t \mid t \in T \}$ then, without simplification, the state space Q of X quickly becomes computationally intractable, even for relatively small values of R and K. For example, if $R = 8$ and $K = 40$ then $|Q| \approx 4 \cdot 10^8$. ($|Q|$ is the cardinality of the state space Q.)

A key observation that drastically reduces the size of the state space (while still supporting an exact solution) is the following. Due to assumptions concerning i) the identical probabilistic nature of individual input sources and ii) the uniformity of cell routing, it is possible to lump (partition) the state space Q according to the following equivalence relation. Letting q_i denote the number of cells in the shared buffer that are destined for output port i ($i = 1, 2, \ldots, R$), two states $q = (q_1, q_2, \ldots, q_R)$ and $q' = (q_1', q_2', \ldots, q_R')$ are ***equivalent*** (and, hence, in the same lump) if and only if q' is a permutation of q. Letting $\overline{Q}$ denote the resulting partition of Q, it then suffices to consider the corresponding reduced stochastic process $\overline{X} = \{\overline{X}_t | t \in T\}$ where, for all $t \in T$, $\overline{X}_t$ is the equivalence class (lump) that contains state X_t. For various choices of queue capacity K, the extent of this reduction is indicated in Tables 1 and 2, where the number of output ports is $R = 4$ and $R = 8$, respectively. Specifically, these tables compare the size of the original state space Q with that of the reduced space $\overline{Q}$, where we see that reductions of several orders of magnitude are possible.

Table 1: State-space size reduction if $R = 4$.

	$K = 10$	$K = 20$	$K = 40$	$K = 80$
$\lvert Q\rvert$	10^3	10^4	$1.3 \cdot 10^5$	$1.9 \cdot 10^6$
$\lvert \overline{Q}\rvert$	94	$7.1 \cdot 10^2$	$7.3 \cdot 10^3$	$9.2 \cdot 10^4$

Table 2: State-space size reduction if $R = 8$.

	$K = 10$	$K = 20$	$K = 40$	$K = 80$
$\lvert Q\rvert$	$4.3 \cdot 10^3$	$3.1 \cdot 10^6$	$3.8 \cdot 10^8$	$7.4 \cdot 10^9$
$\lvert \overline{Q}\rvert$	$1.3 \cdot 10^2$	$2.0 \cdot 10^3$	$7.3 \cdot 10^4$	$8.1 \cdot 10^5$

To obtain a feasible means of determining the steady-state probability distribution of $\overline{X}$, each state $\overline{q} \in \overline{Q}$ can be conveniently represented by an ordered pair (b,e) consisting of a sequence of "occupancy values" b and an "occupancy vector" e (see appendix A). Using this representation, algorithm 1 is based on a functional formulation of transitions to intermediate states (during a slot) that result from the "send" operation and, in turn, each intraslot arrival. Beginning with state $\overline{X}_t$, which expresses the buffer occupancy at the end of slot t, and accounting for these intraslot transitions during slot t+1, the resulting state (following the final cell arrival) is then the next state $\overline{X}_{t+1}$ of the lumped buffer model. (Again, see appendix A for further details.) Accordingly, if we account for the behavior of the Markovian source model $M = \{ M_t \mid t \in T \}$ then, given that $\overline{X}_t = (b,e)$ and $M_t = m$ (the number of On sources during slot t), these functions, together with the transition probabilities of M, determine the conditional probabilities

$$P\left[\overline{X}_{t+1} = (b',e'), M_{t+1} = m' \middle| \overline{X}_t = (b,e), M_t = m\right] \tag{5}$$

for all $(b',e') \in \overline{Q}$ and all $m' \in \{0,1,\ldots,N\}$. In other words, if we let Z_t be the pair of variables $(\overline{X}_t, M_t)$ and consider the corresponding stochastic process $Z = \{ Z_t \mid t \in T \}$ then the transition probabilities of Z at time t are given by (5). Beginning with some arbitrary distribution for the initial state variable $Z_0 = (\overline{X}_0, M_0)$ the distribution of Z_{t+1} can thus be determined iteratively from the distribution of Z_t and the transition probabilities (5) at time t, for $t = 0,1,2,\ldots$ until a steady-state (stationary) condition is sufficiently well approximated. More precisely, the computation terminates when, for all $(b,e) \in \overline{Q}$ and all $m \in \{0,1,\ldots,N\}$ the absolute value of the relative difference

$$\frac{P[Z_{t+1} = ((b,e),m)] - P[Z_t = ((b,e),m)]}{P[Z_t = ((b,e),m)]}$$

is less than some very small positive number. Given that t satisfies this condition, the distribution we seek is then obtained by summing over the states of the source model M, i.e., for all $(b,e) \in \overline{Q}$,

$$P\left[\overline{X}_t = (b,e)\right] = \sum_{m=0}^{N} P[Z_t = ((b,e),m)].$$

Although application of this algorithm becomes intractable for large values of R and K, as indicated in tables 1 and 2, the reduction in state space size provided by the reduced model permits feasible solutions for switches with moderate dimensions. For example, when implemented on an HP9000 series 700 workstation, algorithm 1 can accommodate an 8×8 switch with random traffic and a buffer capacity of K=60 or bursty traffic and a buffer capacity of K=40. Moreover, and as noted in section 1, exact models are likewise very useful if a large system can be approximately decomposed into smaller subsystems that admit to such representation. For instance, in addition to more obvious uses such as convolution, this type of "divide and conquer" approach was employed by the approximate method developed in [3]. Details as to how algorithm 1 can be exploited in concert with algorithm 2 are presented at the end of the subsection that follows.

3.2 Algorithm 2 (Approximate)

This algorithm approximates the shared-buffer occupancy probabilities via an iterative procedure that considers, at each successive step, subsystems of growing size. Presuming an $N\times R$ switch with a shared buffer of capacity K, for a specified integer r, where $1 \le r \le R/2$, we initially choose two disjoint subsets $1(r)$ and $2(r)$ of the set $\{1, 2, \ldots, R\}$ of all output ports, where each subset has cardinality r. Although just how these subsets are chosen is relatively arbitrary, to simplify the discussion we assume that both R and r are integer powers of 2. Moreover, without loss of generality, we can let $1(r)$ be output ports 1 through r and $2(r)$ be ports $r + 1$ through $2\,r$, i.e,

$1(r) = \{1, 2, \ldots, r\}$ and $2(r) = \{ r+1, r+2, \ldots, 2r\}$.

The shared buffer, together with the $2r$ output ports $1(r) \cup 2(r)$, will be referred to simply as an *r-subsystem*. In keeping with the notation of the exact method, the buffer state at a given time t is described by the random variables

$X_{i(r),t}$ = number of cells in the buffer at time t addressed to output ports in $i(r)$, $i = 1,2$

and, to represent arrivals and departures in an analogous fashion, we let

$W_{i(r),t}$ = number of incoming cells at time t addressed to ports in $i(r)$, $i = 1, 2$
$Z_{i(r),t}$ = number of cells that depart the buffer at time t from output ports in $i(r)$, $i = 1, 2$.

Relative to this model of an r-subsystem, and recalling that M_t is the number of sources in the On state at time t, let us now consider the following limiting distributions concerning arrivals (A_r), combined buffer occupancy and source activity (B_r, referred to as the ``buffer-source" distribution), and departures (D_r).

$$A_r(w_1, w_2 | m) = \lim_{t\to\infty} P\left[W_{1(r),t} = w_1, W_{2(r),t} = w_2 | M_t = m\right] \tag{6}$$

$$B_r(x_1, x_2, m) = \lim_{t\to\infty} P\left[X_{1(r),t} = x_1, X_{2(r),t} = x_2, M_t = m\right] \tag{7}$$

$$\begin{aligned} D_r(z | x, m) &= \lim_{t\to\infty} P\left[Z_{1(r),t} = z | X_{1(r),t} = x, M_t = m\right] \\ &= \lim_{t\to\infty} P\left[Z_{2(r),t} = z | X_{2(r),t} = x, M_t = m\right] \end{aligned} \tag{8}$$

Computation of the conditional arrival probabilities $A_r(w_1, w_2 \mid m)$ is straightforward since, by the uniform routing assumption,each arriving cell has a probability $1/R$ of being addressed to a given output port. Hence, as both $1(r)$ and $2(r)$ have cardinality r, for either subset $i(r)$ (i=1,2), the probability of an arrival being addressed to a port in $i(r)$ is simply r/R. With this observation, let $BI_{n,p}$ denote the binomial distribution having parameters n and p, i.e., for $0 \le i \le n$,

$$BI_{n,p}(i)=\binom{n}{i}p^i(1-p)^{n-i}.$$

Then in case all arrivals are accepted (the "no-loss" case), the formulation of A_r is immediate, i.e.,

$$A_r(w_1,w_2|m)=BI_{m,2r/R}(w_1+w_2)\cdot BI_{w_1+w_2,1/2}(w_1). \tag{9}$$

To extend (9) so that it can account for cell losses, we assume further that there is no statistical dependence between the address of an arriving cell and the event that it is one the cells discarded among the *m* that arrive. In this case, the extension is easily obtained.

The departure distribution D_r, on the other hand, is more difficult to determine once the value of *r* is greater than 1. It is here that we choose to introduce an approximate computation based on the following recursive formulation of D_r in terms of $D_{r/2}$ and $B_{r/2}$ (where $r > 1$). (The inexact nature of this formula will be discussed in a moment.)

$$D_r(z|x,m)=\sum_{x_1=0}^{x}\sum_{z_1=0}^{z}D_{r/2}(z_1|x_1,m)D_{r/2}(z-z_1|x-x_1,m)E_{r/2}(x_1|x_1+x_2,m) \tag{10}$$

where $E_r(x_1 \mid x_1+x_2, m)$ is the probability that x_1 cells in the buffer are addressed to output ports in $1(r)$, given that i) x_1+x_2 are addressed to ports in $1(r) \cup 2(r)$ and ii) m sources are active, i.e.,

$$E_r(x_1|x_1+x_2,m)=\frac{B_r(x_1,x_2,m)}{\sum_{i,j|i+j=x_1+x_2}B_r(i,j,m)}. \tag{11}$$

The knowledge of A_r and D_r permits the formulation of the transition probabilities for any pair of states in the model determined by *r*, i.e., the model that represents an aggregation of output ports according to sets $1(r)$ and $2(r)$. This, in turn, permits the computation of the steady-state buffer-source distribution B_r, using an iterative method similar to that employed by algorithm 1. This method relies on both A_r and D_r in the sense mentioned above. Accordingly, for a given value of *r* (i.e., a given iteration of algorithm 2), the calculations of A_r and D_r must precede that of B_r. Once B_r is computed, if $2r = R$ then the computation terminates since, in this case, the *r*-subsystem accounts for all the output ports. If not, the number of ports considered is doubled (i.e., *r* is replaced by 2*r*) and the computations are repeated for this larger *r*-subsystem; in particular, the new values for D_r are obtained using the recursion given by (10). For additional details concerning the computation of B_r, please see appendix B . Accordingly, algorithm 2 can be summarized as follows.

Step 1: $r=1$. $D_1(0|x,m)=\begin{cases}1 & \text{if } x=0\\ 0 & \text{else}\end{cases}$ and $D_1(1|x,m)=\begin{cases}1 & \text{if } x>0\\ 0 & \text{else}\end{cases}$.

```
Step 2:  Compute $A_r$ (see (9)).
Step 3:  Compute $B_r$ (see appendix B).
Step 4:  If $2r = R$, exit; otherwise continue.
Step 5:  $r \leftarrow 2r$.
Step 6:  Compute $D_r$ (see (10)).
Step 7:  Go to Step 2.
```

This algorithm yields a fairly good approximation of the steady-state occupancy probabilities of a shared-buffer switch with bursty traffic. The two principal sources of approximation error are the following.

1. In solving each r-subsystem model (Step 3), we assume that the storage capacity for cells addressed to the $2r$ ports in $1(r) \cup 2(r)$ coincides with the buffer capacity K. In reality, this capacity is shared among cells destined for all R ports.
2. In the same step, we assume that the number of departures from ports in $1(r)$ is independent of the number of departures from ports in $2(r)$, i.e., for all $t \in T$, the random variables $Z_{1(r),t}$ and $Z_{2(r),t}$ are statistically independent. This is not generally true.

The number of main iterations of this algorithm is clearly $\log_2 R$. However, it's important to note that this number can be reduced if an r-subsystem can be solved directly for a value of r that is greater than 1. This can be done by applying algorithm 1 to a special $N \times r$ shared-buffer system, where sources in the On state transmit cells with probability r/R. Accordingly, Step 1 of (modified) algorithm 2 then begins at a value $r > 1$, where data for this value is supplied by algorithm 1. This combined use of both algorithms (the "integrated" approach referred to in the title and introduction) obviously reduces the execution time. Moreover, it also improves the precision of the results, since each iteration introduces an error of approximation. Further discussion of the nature of such errors is deferred to the end of section that follows.

4 RESULTS

Recalling some of the motivation that was mentioned at the outset (see section 1), because of the severe requirements imposed on ATM cell loss probabilities, highly accurate results (with high levels confidence) are difficult to obtain by simulation. Although there has been some progress in the development of fast simulation techniques for rare events, e.g., various forms of "importance sampling", these typically rely on very special knowledge of the system in question. If approximate analytic methods are used instead then, even though they often provide reasonably accurate results in the higher probability region of buffer occupancy, they tend to be much less accurate for large occupancy values. In other words, the asymptotic behavior of the distribution (its tail) is not well approximated. However, for purposes such as determining buffer dimensions that insure satisfactory loss performance with respect to stringent cell loss probability requirements, accurate knowledge of this tail is crucial.

To pursue this matter in terms of the development of the previous section, we first examine the use of algorithm 1 as it applies to two of the logical output queues of an $N \times N$ shared buffer. Further, we suppose that the capacity of this buffer is large enough so that, effectively, it can be regarded as infinite. This situation can thus be represented by an (exact) model of an N×2 shared-buffer switch, where On sources transmit cells with probability $2/N$. To satisfy the "effectively infinite" assumption, we take the buffer capacity K to be large enough to insure cell loss probabilities that are less than 10^{-15}. Due to the small number of output ports, this model can be efficiently solved using algorithm 1. The purpose of the analysis is to study the correlation among the occupancy distributions of two queues in an N×N system. For the random (Bernoulli) traffic case, the joint occupancy distribution of the two queues was obtained in both [3] (using z-transforms) and [5]. The analysis here is similar to the latter, but is extended to the case of bursty sources.

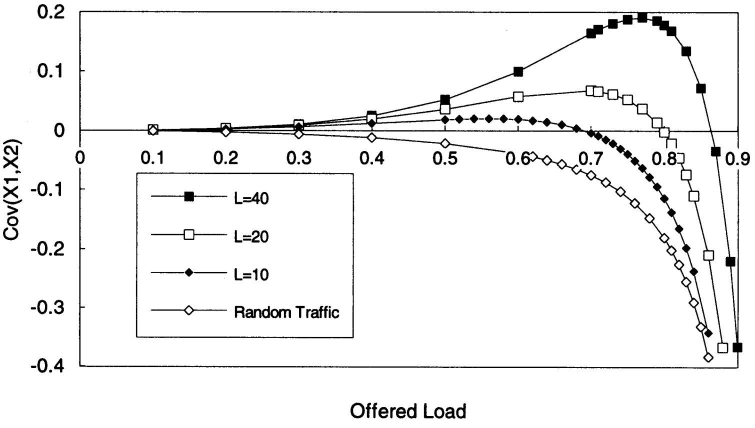

Figure 1 Covariance between the length of two queues.

Figure 1 displays the covariance (between two queues) as a function of offered load, for random and bursty traffic and N equal to 16. Among other things, it is interesting to note the sign of the covariance. For the random traffic case, the covariance is always negative, while in the case of bursty traffic it is positive for low loads and becomes negative as the load increases.

The knowledge of the joint distribution of the occupancy probability of the two queues, together with the assumption of a effectively infinite buffer, permits the variance of the buffer's occupancy distribution to be formulated as follows. Let Y_∞ denote the random variable whose probability distribution is the limiting distribution of Y_t as $t \rightarrow \infty$, i.e., Y_∞ is the steady-state number of cells in the shared buffer. Similarly, let $X_{1,\infty}$ and $X_{2,\infty}$ be the random variables representing the steady-state occupancy of queue 1 and queue 2, respectively, in the N×2 system described above. Then, taking the subscripts ∞ to be implicit (context should suffice to convey the steady-state interpretation), the mean and variance of $Y = Y_\infty$ can be formulated as follows in terms of $X_1 = X_{1,\infty}$ and $X_2 = X_{2,\infty}$, where we let queue 1 be representative of single-queue behavior. Note that since X_1 and X_2 are identically distributed, X_2 could likewise serve this purpose.

$$E[Y] = N \cdot E[X_1] \tag{12}$$
$$\mathrm{Var}(Y) = N \cdot \mathrm{Var}(X_1) + N(N\text{-}1) \cdot \mathrm{Cov}(X_1,X_2) \tag{13}$$

Formula (13) can also be used to determine the error (with regard to variance) introduced by assuming that the queues are statistically independent. (The latter assumption is convenient since it permits the distribution of Y to be obtained via the N-fold convolution of the distribution of a single queue.) Assuming such independence, $\mathrm{Var}(Y) = N{\cdot}\mathrm{Var}(X_1)$ and, accordingly, the error due to this assumption is given by $\delta = N(N\text{-}1){\cdot}\mathrm{Cov}(X_1,X_2)$. Further, since the sign of δ is clearly the sign of the covariance, an approximation based on convolution underestimates the variance if $\mathrm{Cov}(X_1,X_2) > 0$. Since there is no error with regard to the value of $E[Y]$, we can reasonably expect that, by using convolution, the results are conservative only if the covariance is negative. Also, it appears that this approximation gets worse for growing values of $|\mathrm{Cov}(X_1,X_2,)|$ and becomes useless when this value approaches that of $N{\cdot}\mathrm{Var}(X_1)$. This is borne out by Figure 2, which compare occupancy distributions obtained by convolution with corresponding (and more accurate) results determined by simulation. Specifically, this plot demonstrate that convolution provides an overly optimistic estimate of the distribution of Y in the case of heavy, bursty traffic and, hence, cannot be relied on for practical applications.

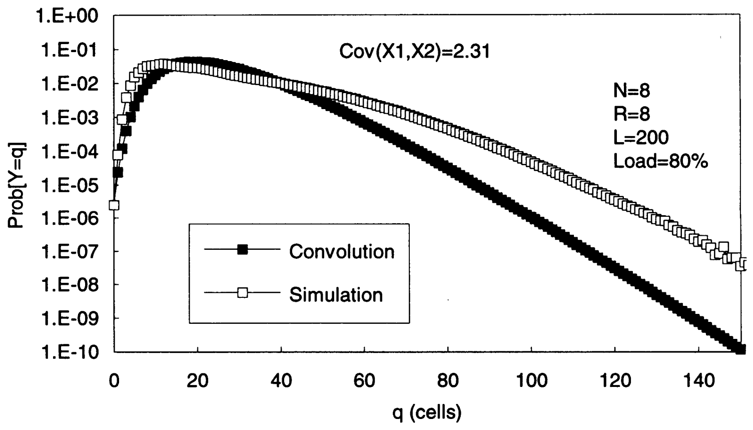

Figure 2: Comparison between convolution and simulation.

It is also worth noting that the knowledge of mean and variance of the shared-buffer occupancy distribution permits another approximation of loss performance. As mentioned in section 1, for the case of a shared buffer submitted to random traffic, [3] has proposed approximating the occupancy probability of an (infinite) shared buffer with the density of an appropriate Gamma distribution. This particular distribution was considered because of the exponential asymptotic behavior of its density function, which is typical of many queueing systems. More precisely, by computing the mean and variance of the occupancy distribution of an infinite shared buffer, this Gamma distribution is chosen such that its first two moments match the computed values. The loss probability of a K-capacity buffer is then estimated as the probability that a random variable with this Gamma density has a value greater than K.

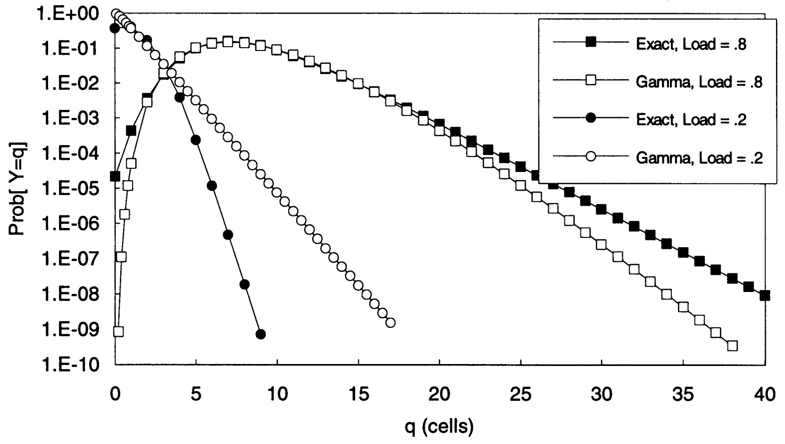

Figure 3: Comparison between exact results and Gamma function approximation.

Figure 3 displays the occupancy distributions of a 4x4 switching element, with $K = 50$ and offered loads ρ equal to 0.2 and 0.8, respectively. The Gamma distribution's density function is then compared with the distribution obtained from algorithm 1, indicating that the Gamma density provides a fairly good approximation for the higher probability states. On the other hand, one can see that it fails to capture asymptotic behavior in the low probability region. Moreover, we see that the estimation errors in this region are load-dependent, with values being overestimated in case $\rho = 0.2$ and underestimated if $\rho = 0.8$.

We now turn to the analysis that utilizes algorithm 2. As noted earlier, this algorithm estimates the loss probability of a shared-buffer system with bursty traffic, even in cases where the buffer is large. To validate this approach, we compare the results obtained by algorithm 2 with those obtained by simulation (for large buffers and high loads) and by algorithm 1 (for smaller buffers).

Figures 4 and 5 plot the loss probability as as function of buffer capacity K for both an 8x8 (Figure 4) and a 16x16 (Figure 5) switch. In each case, we consider random traffic and three different values of offered load ($\rho = 0.2, 0.7, 0.8$).

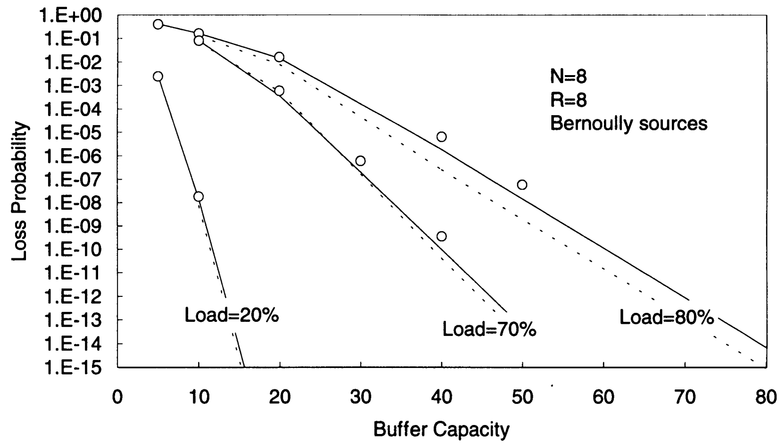

Figure 4: Comparison between exact and approximate results; random traffic.

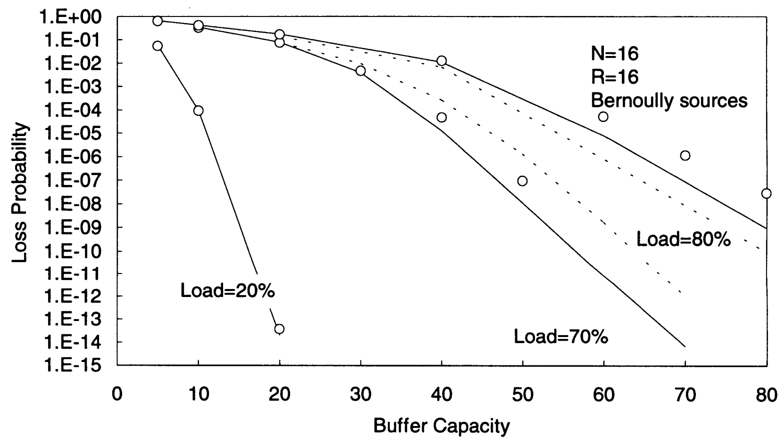

Figure 5: Comparison between exact and approximate results; random traffic.

Comparisons of the model result (straigth line) with simulation data and with exact analytic results (both plotted as circles) reveal that the error increases as the capacity K gets larger. However, in all the cases considered, the error's value is less than an order of magnitude. Included in the same figures are some plots of results obtained from the algorithm described in [5] (dashed lines). In all cases, it can be seen that algorithm 2 provides a more accurate estimate of cell loss probability.

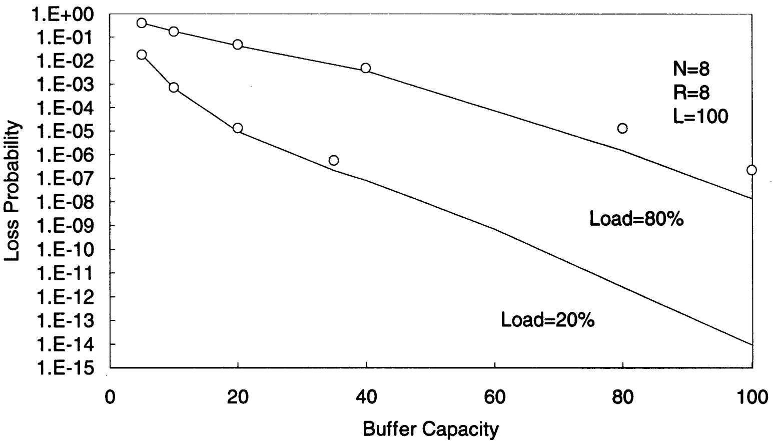

Figure 6: Comparison between exact and approximate results; $L = 100$.

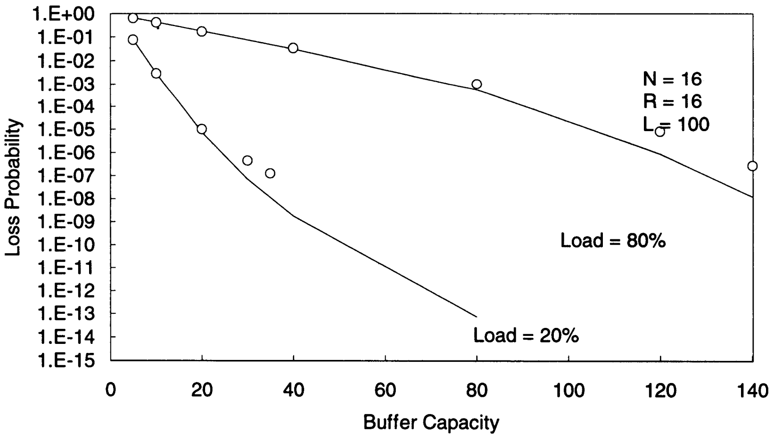

Figure 7: Comparison between exact and approximate results; $L = 100$.

Figures 6 and 7 are similar to Figures 4 and 5, respectively, except that we now consider traffic sources that are bursty. Specifically, the mean burst length of a source is L=100 and, again, two instances of offered load are considered, namely $\rho = 0.2$ and $\rho = 0.8$.

To illustrate another application that integrates the use of an exact method (algorithm 1) with an approximate formulation (described below), we estimate the advantages, in terms of required storage capacity, of a shared-buffer architecture as compared with a (simpler) switch having dedicated output queues (no sharing). For given values of N, R, L, and ρ_{in} along with an admissible (maximum allowed) cell loss probability p_a, let K be the capacity of the shared buffer and let K' be the capacity of each of the output buffers in the dedicated case. Hence, RK' is the total capacity of the latter. Further, let s denote the fraction of this capacity that is required in the case of a shared-memory switch (presuming the same value of p_a for each), i.e., $s = K/RK'$. Thus the value of s (the "relative saving") can vary from a minimum of $1/R$ (corresponding to the greatest theoretical reduction in required memory) to a maximum of 1 (no reduction).

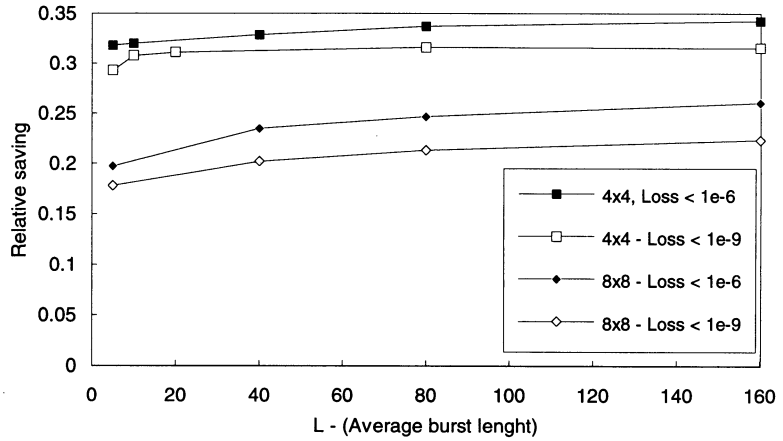

Figure 8: Memory saving afforded by buffer sharing.

Figure 8 illustrates how this relative saving varies as a function of the mean burst length L for different choices of $N = R$ (4, 8). The activity ρ_{in} of a source is equal to 0.8 and two loss probability targets are considered. As indicated by the figure, we have the following observations with respect to the combinations of parameter values considered.

a) The advantage of buffer sharing increases (as reflected by smaller values of s) as the number of ports gets larger.

b) Likewise, buffer sharing is more advantageous as the loss probability target becomes lower (more severe).

For very low loss probability targets p_a, observation b) suggests an empirical means of obtaining a quick and conservative estimate of the required capacity K of a shared-buffer switch. Let $K(p_a)$, $K'(p_a)$, and $s(p_a)$ be the the values of K, K', and s that result from a particular choice of p_a. Then, from the definition of s, it follows that

$$K(p_a) = s(p_a)RK'(p_a). \tag{14}$$

If the value of p_a is relatively high (in an ATM context, $p_a \geq 10^{-6}$) then $K(p_a)$ can be obtained by simulation. As for $K'(p_a)$, algorithm 1 can serve to determine its value (letting $R = 1$), even in cases where p_a is much smaller (as we exploit below). Then, to "scale up" these calculations to a more severe loss requirement, let p_a' denote a lower admissible cell loss probability, i.e., $p_a' < p_a$. Then by observation b) it follows that $s(p_a') < s(p_a)$ and hence, applying (14), we have

$$K(p_a') = s(p_a')RK'(p_a') < s(p_a)RK'(p_a') = K(p_a)K'(p_a')/K'(p_a). \tag{15}$$

Given the value of $K'(p_a')$, which again can be accurately determined using algorithm 1, (15) thus provides a conservative estimate of shared-buffer capacity in cases where the target loss probability is much lower (e.g., $p_a' \leq 10^{-9}$), namely

$$K_{est}(p_a') = K(p_a)K'(p_a')/K(p_a)$$

For both a 4×4 and an 8×8 switch ($R = 4, 8$) and as a function of traffic burstiness, Figure 9 illustrates the extent to which $K_{est}(p_a')$ overestimates the actual values.

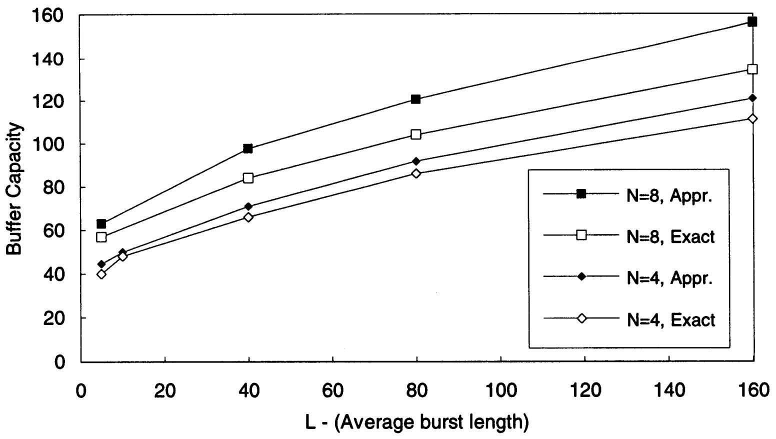

Figure 9: Validation of the empirical dimensioning rule.

Here, the original and modified target loss probabilities considered are $p_a = 10^{-6}$ and $p_a' = 10^{-9}$, respectively, under an assumed offered load of $\rho = 0.8$. As can be observed, the relative error of the estimate (i.e., the quantity $(K_{est} - K)/K$) is somewhat smaller for $R = 4$ as compared with $R = 8$. For both switches, this error increases slowly as the mean burst length L becomes larger. In particular, for an 8×8 switch with $L = 160$, the relative error is approximately 0.2 (20%).

5 SUMMARY

By employing an exact solution method (algorithm 1) along with a new approximate method (algorithm 2), we have shown that a synergistic use of both can be beneficial in the context of shared-buffer switch evaluation. Aside from the usual advantages associated with comparing exact vs. approximate results, we find that true integration, where both are employed for a single purpose, can likewise be very useful. Further, we have shown that algorithm 2, particularly if used in concert with algorithm 1, can provide reasonably accurate results even for realistically large switches in the presence of a bursty traffic environment. Finally, by examining the reduction in buffer capacity, relative to a dedicated-buffer architecture, that results from buffer sharing, we have found that memory saving increases as the target loss probability decreases. In turn, this suggested an empirical means of conservatively dimensioning a shared-buffer switch such that, even in the case of extremely low target probabilities, the capacity so determined is reasonably close to what's actually required.

6 REFERENCES

[1] Devault, M., Cochennec, J. and Servel, M. (1988) The PRELUDE ATD experiment: Assesments and future prospects. *IEEE Journal on Selected Areas in Communications*

[2] Hluchyj, M. and Karol, M. (1988) Queueing in high performance packet switching. *IEEE Journal on Selected Areas in Communications*

[3] Eckberg, A.E. and Hou, T.C. (1988) Effects of output buffer sharing on buffer requirements in an ATDM packet switch, in *Proc. INFOCOM '88*, New Orleans, LA.

[4] Bianchi, G. and Turner, J., (1993)Improved queueing analysis of a shared buffer switching network, in *Proc. INFOCOM '93*, San Francisco, CA.

[5] Meyer, J. F., Montagna, S. and Paglino, R. (1993) Dimensioning of an ATM switch with Shared Buffer and Threshold Priority. *Computer Networks and ISDN Systems*, **26**, 95-108

[6] Louvion, J., Boyer, P. and Gravey, A. (1988) A discrete time single-server queue with Bernoulli arrivals and constant service time, in *Proc. 12th Int'l Teletraffic Congress,* Torino, Italy

[7] Petit, H. and Desmet, M. (1990) Performance evaluation of shared buffer multiserver output queue switches used in ATM, in *Proc. 7th ITC Seminar*, Morristown, NJ

[8] Yamashita, H., Perros, H.H. and Hong, S.W. (1991) Performance modeling of a shared buffer ATM switch architecture, in *Proc. 13th Int'l Teletraffic Congress*, Copenaghen, Denmark

APPENDIX A

Recalling notation introduced in section 3.1, we let Q be the set of possible (buffer occupancy) states of the R logical queues of a shared buffer of finite capacity K, i.e., if q_i is the number of cells stored in logical queue i ($0 \le q_i \le K$, $1 \le i \le R$) then

$$Q = \{(q_1, q_2, \ldots, q_R) \mid 0 \le q_1 + q_2 + \ldots + q_R \le K\}.$$

In turn, we identify states that are permutations of one another via an equivalence relation on Q, letting $\overline{Q}$ denote its corresponding partition (set of equivalence classes). As noted in section 3.1, the shared buffer can then be represented by the reduced stochastic process $\overline{X} = \{\overline{X}_t \mid t \in T\}$, where $\overline{X}_t$ is the equivalence class that contains state X_t.

In what follows, we show how the transition structure of X can be described, in part, by a convenient representation of the elements of $\overline{Q}$. Specifically, if $q = (q_1, q_2, \ldots, q_R) \in Q$, let $\overline{q}$ denote its equivalence class, i.e., the state in $\overline{Q}$ that contains q. Then an *occupancy value* for $\overline{q}$ is the value of some coordinate q_i of q. If, further, we let $b = (b_1, b_2, \ldots, b_r)$ be a listing, in increasing order, of all the different occupancy values for $\overline{q}$ (where $1 \le r \le R$), we can define the *occupancy vector* of $\overline{q}$ to be the r-tuple $e = (e_1, e_2, \ldots, e_r)$, where, e_j is the number of different logical queues having occupancy value b_j ($1 \le e_j \le R$). Note that, by the definitions of $\overline{Q}$ and $\overline{q}$, it follows that of both b and e are invariant relative to the choice of a representative state $q \in \overline{q}$. It is also easily shown that if $\overline{q}$ and $\overline{q}'$ are distinct states in $\overline{Q}$ then the corresponding ordered pairs (b,e) and (b',e') are likewise distinct. In other words, this pair of r-tuples provides a unique representation of a state in $\overline{Q}$. Moreover, the set of all such representations is just the set of all ordered pairs (b,e), with $b = (b_1, b_2, \ldots, b_r)$ and $e = (e_1, e_2, \ldots, e_r)$, such that

$1 \le r \le R$,
$0 \le b_i \le K$,
$b_1 < b_2 < \ldots < b_r$,
$\sum_{j=1}^{r} e_j = R$, and
$\sum_{j=1}^{r} b_j e_j \le K$

From this point on, a state of the process $\overline{X}$ will be identified with its corresponding pair (b,e), where the latter is now regarded as an element of $\overline{Q}$. In these terms, the transition structure of the process can be formulated as follows.

Given that the system is in a state $\overline{X}_t = (b,e)$ at the beginning of time slot t, the state of the system after departures caused by the "send" operation in slot t (this is an intermediate state that precedes the subsequent arrivals; hence, it is not an explicit part of the behavior of $\overline{X}$) is given by the function $d(b,e)$, where

$$d(b,e)=\begin{cases}((b_1-1,b_2-1,\dots,b_r-1),e) & \text{if } b_1>0\\ ((b_1,b_2-1,\dots,b_r-1),e) & \text{if } b_1=0 \text{ and } b_2>1\\ ((0,b_3-1,\dots,b_r-1),(e_1+e_2,e_3,\dots,e_r)) & \text{if } b_1=0 \text{ and } b_2=1\end{cases}$$

Suppose now that (b,e) is the intermediate state so determined by the function d. The next state $\overline{X}_{t+1}$ is then a function of the arriving cells as well as (b,e). Assuming that each cell is randomly addressed to one of the output queues, the probability that an incoming cell is addressed to one of the e_j logical queues (each with b_j cells) is clearly e_j/R. In a manner similar to how d is defined, a function a (suggesting "arrival") then determines an intermediate state (unless it's the last arrival during slot t) that results from an entry of an arriving cell to one of these e_j queues. Specifically, the function a (whose arguments are the intermediate state (b,e), along with the value j that identifies the queue subset containing the arrival's destination queue) can be expressed as follows.

$a((b,e),j) =$

$$=\begin{cases}((b_1,\dots,b_j,b_j+1,b_{j+1},\dots,b_r),(e_1,\dots,e_{j-1},e_j-1,1,e_{j+1},\dots,e_r)) & \text{if } b_{j+1}>b_j+1 \text{ and } e_j>1\\ ((b_1,\dots,b_{j-1},b_j+1,b_{j+1},\dots,b_r),(e_1,\dots,e_{j-1},1,e_{j+1},\dots,e_r)) & \text{if } b_{j+1}>b_j+1 \text{ and } e_j=1\\ (b,(e_1,\dots,e_{j-1},e_j-1,e_{j+1}+1,e_{j+2},\dots,e_r)) & \text{if } b_{j+1}=b_j+1 \text{ and } e_j>1\\ ((b_1,\dots,b_{j-1},b_{j+1},\dots,b_r),(e_1,\dots,e_{j-1},e_{j+1}+1,e_{j+2},\dots,e_r)) & \text{if } b_{j+1}=b_j+1 \text{ and } e_j=1\end{cases}$$

As described in section 3.1, these functions then serve to formulate the transition probabilities of the composite process $Z=\{(\overline{X}_t, M_{t+1}) \mid t\in T\}$.

APPENDIX B

The key step of the approximate method proposed in section 3.2 consists of finding the steady-state distribution $B_r(x_1,x_2,m)$, given the distributions $D_r(z\mid x,m)$ and $A_r(w_1,w_2 \mid m)$ of the departure and arrival processes, respectively. Recalling the exact meanings of each, we have

$B_r(x_1, x_2, m)$	=	the steady-state probability of having x_1 cells in the buffer addressed to output ports in the set $1(r) = \{1, 2,\dots,r\}$, x_2 addressed to output ports in the set $2(r) = \{r+1, r+2, \dots, 2r\}$, and m sources in the On state.
$D_r(z \mid x, m)$	=	the steady-state probability of having z cell departures during a slot from ports in $1(r)$ (alternatively $2(r)$), given that x cells are addressed to these ports and m sources are active.
$A_r(w_1, w_2 \mid m)$	=	the steady-state probability of having w_1 cell arrivals during a slot addressed to ports in $1(r)$ and w_2 addressed to ports in $2(r)$, given that m sources are active.

To compute $B_r(x_1, x_2, m)$, we employ an iterative algorithm similar to the one used for the exact model. Let $B_r^t(x_1, x_2, m)$ be the probability distribution, at time t, which, in the limit (as

$t \to \infty$), yields the steady-distribution $B_r(x_1, x_2, m)$. Further, let $C_r^t(x_1, x_2, m)$ be the probability distribution of the intermediate state, during slot t, that results from cell departures (but is prior to slot t arrivals). Then it can be shown that, for all $t \in T$,

$$C_r^t(x_1,x_2,m) = \sum_{i=x_1+\min(x_1,1)}^{\min(x_1+r,K)} \sum_{j=x_2+\min(x_2,1)}^{\min(x_2+r,K-i)} B_r^t(i,j,m) D_r(i-x_1|i,m) D_r(j-x_2|j,m)$$

$$B_r^{t+1}(x_1,x_2,m) = \sum_{i=\max(x_1-m,0)}^{x_1} \sum_{j=\max(x_2-m+x_1-i,0)}^{x_2} \sum_{l=0}^{N} C_r^t(i,j,l) A_r(x_1-i,x_2-j|m) S(m|l)$$

where $S(m|l)$ is the (time-invariant) probability that m sources are active in a slot, given that l were active in the previous slot. As earlier, we then iteratively compute $B_r^t(x_1, x_2, m)$ for growing t until we reach a time t that yields a sufficiently close approximation of the steady-state distribution $B_r(x_1,x_2,m)$. Again, the specific criterion for termination is a value of t such that the maximum relative difference between the probabilities for slots t and t+1 is less than some very small number.

Sergio Montagna was born in Italy in 1955, He received a degree in physic from the University of Pavia (Italy) in 1978. He joined the Central Research Laboratories of Italtel in January 1983. His current research is concerned in the performance evaluation of ATM systems and networks.

Roberto Paglino was born in Italy in 1962, He graduated from the Universita' Statale di Milano (Milan, Italy), where he obtained a degree in computer science. Since 1987 he has been with Italtel as a researcher in the sytem engineering department of the central R&D laboratories. His interest are in the performance and architecture of ATM systems.

John F. Meyer received the B.S. degree from the University of Michigan, Ann Arbor, the M.S. degree from Stanford University, Stanford, CA, and the Ph.D. degree in communication sciences from the University of Michigan in 1957, 1958, and 1967, respectively. He is currently a Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. He has been active in computer and system research for 35 years and has published widely in the areas of fault-tolerant computing and model-based evaluation of system performance, dependability, and performability.