Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

PERFORMANCE EVALUATION OF ATM ARCHITECTURES IN MULTIMEDIA TRAFFIC ENVIRONMENTS

Siddharth Behera and Gopal Meempat

Department of Computer Science and Engineering
115 Ferguson Hall
University of Nebraska-Lincoln
Lincoln, NE 68588-0115

ABSTRACT

A simulation model for the performance evaluation of fast packet switching architectures under multimedia traffic conditions is presented. The simulator is based on an integration of the SLAM-II tool with special-purpose C routines that model the internal structure and operational details of the basic switching fabrics. Two types of traffic sources, namely, Bernoulli sources and correlated sources (modeled as two-state Markov chains) are considered in this study. The latter class of sources provide an accurate model for multimedia traffic components, such as voice and video, which are inherently bursty. The results of the simulation indicate the significant deterioration of the switch performance under correlated traffic from that predicted by the Bernoulli models. Furthermore they illustrate the efficacy of the input-windowing mechanism in remedying this drawback to a substantial degree.

1 INTRODUCTION

The advent of the fiber-optic transmission technology has extended the scope of the integrated services digital network (ISDN) to the arena of broadband applications. The spectrum of possible applications include full-motion video, high-speed medical imaging and high-definition television (HDTV), to name only a few. Such applications typically generate information at rates of 50–150 Mbytes/s per connection. A full exploitation of the bandwidth potential of the fiber-optic transmission components is constrained by the speed limitations of the switching components which are expected to be based on conventional electronics at least for a few more decades. Current technology limits the switching speeds of electronic devices to about a Gbit/s. In view of this bottleneck, there has been a concerted effort within the communications community to design efficient switching architectures which would employ conventional devices and yet match the speed capabilities of optical transmission components. This effort has culminated in what has come to be known as the asynchronous transfer mode (ATM) or fast packet switching technique, the underlying principle of which is an exclusively hardware-based and highly parallel implementation. In an ATM environment, information is divided into elementary units referred to as *cells* which are transmitted across the network independently of each other.

Several types of ATM switch architectures have been proposed by various researchers in the recent past (Andindo and Seeto 1988 – Szymanski and Shaikh 1989, Jenq 1983 – Tobagi 1990). Each of these architectures has its own intrinsic advantages and disadvantages. For example, the crossbar switch (Tobagi 1990) which features a low cell blocking probability and delay is rather expensive to implement. In contrast, the banyan architecture (Tobagi 1990, Szymanski and Shaikh 1989, Jenq 1983) merits from a low implementational cost, especially for large switching systems, but suffers from a cell delay and blocking probability significantly higher than in the crossbar switch. Specifically, the number of crosspoint connections in a crossbar switch (Fig. 1) equals the square of the number of I/O lines (N) whereas those in a banyan switch (Fig. 3) equals $2N \log_2 N$. The organization of the banyan network is based on arranging several $p \times p$ (*e.g.*, $p = 2$) crossbar modules in multiple stages (Fig. 3). The connectivity within a banyan is such that there exists a unique path from each input node to each output node. This enables the so-called *self-routing* of input cells through the banyan fabric based on the values of the various bits of the cell address field. The magnified cell loss and delay in the banyan network is attributable to the blocking and queueing that occur at the shared internal stages of the fabric.

Given the broad spectrum of techniques available

for ATM switching, the major challenge ahead in implementing operational broadband ISDN's lies in the development of appropriate performance models. An accurate performance evaluation would serve to assess the functionality of the various architectures in specific traffic environments. In addition, it would lay the groundwork in developing optimal algorithms for the management of network resources. Several researchers have conducted performance studies of certain specific classes of ATM switches in the past few years (Szymanski and Shaikh 1989, Karol, Hluchyj, and Morgan 1987, Hluchyj and Karol 1988, Jenq 1983). It may, however, be noted that the majority of these attempts were based on the rather restrictive assumption of uniform Bernoulli statistics for the traffic arrival processes. Probabilistic models based on this assumption fail to capture the intrinsic time-correlation characteristics of actual traffic. This limitation underscores the need for further efforts at estimating the performance of various architectures subject to the more realistic, correlated (or bursty), traffic conditions.

In this paper, we conduct a performance evaluation of some of the representative switching architectures within a bursty traffic environment. Our approach will be based on discrete-event simulation to facilitate an accurate representation of the characteristics of the various switching systems, and the statistical attributes of the correlated cell arrival processes. The principal measures of performance that will be employed in this evaluation are (*i*) blocking probability, and (*ii*) delay. In section 2 we will provide a discussion of the different ATM architectures to be considered. Section 3 will focus on the models to be adopted for representing the stochastic behavior of the various traffic sources. This will be followed, in section 4, by a description of the organization of the simulator employed for our performance studies. The results of the simulation will be presented in section 5, and the paper will be concluded in section 6 with a few pertinent remarks.

2 FAST PACKET SWITCHING ARCHITECTURES

In this section, we will examine the basic architectural features and performance enhancement strategies employed in some of the common ATM switching systems. Consistent with the design philosophy of ATM's, it will be assumed that information streams are divided into basic units referred to as *cells*. A *slotted* time structure will be assumed in the following discussion, where a slot is equivalent to the duration of transmission of a cell.

2.1 Basic Configurations

2.1.1 Crossbar Switches

The crossbar switch architecture is comprised of an $N \times N$ array of crosspoint transmission gates each of which could assume two possible states, namely, the *cross* state and the *bar* state (Fig. 1, Tobagi 1990). By appropriately selecting the switch states based on the contents of the address field, each incoming cell can be routed to the appropriate output port (the *self-routing* property) as discussed in (Tobagi 1990). The primary implementational constraint of the crossbar architecture originates from its space complexity (number of transmission gates) which is proportional to the square of the switch size. This limitation renders the crossbar approach unrealistic for large-scale switching systems.

Even though the crossbar switch is internally nonblocking, the problem of *external blocking* needs to be contended with by providing adequate buffering within the fabric. External blocking corresponds to the situation where multiple input cells are being addressed to the same output line in which case only one of the requests can be honored immediately, and the other cells need to be scheduled for retransmission during future slots. Buffering may be provided in crossbar switches (in their basic form) at the input nodes. However, input-buffered switches suffer from the drawback of *Head-of-line* (HOL) blocking which prevents such switches from attaining a saturation throughput of unity (Karol, Hluchyj, and Morgan 1987, Hluchyj and Karol 1988).

2.1.2 Knockout Switches

Knockout switches (Fig. 2, Tobagi 1990) overcome the limitation of crossbar switches caused by the HOL blocking by providing cell buffering at the outputs rather than at the inputs. This is achieved by N number of parallel buses, each of which is used for broadcasting from one of the input ports. Each output port listens to all the buses simultaneously, and hence is able to filter out all cells addressed to it instantaneously. If the total number of cells addressed to an output exceeds one, then only one of them will be transmitted out of the switch in the ongoing slot, with the others being retained in the output buffer for future retransmission. The performance improvement (due to output queueing) resulting from this approach incurs the the cost of each output node being required to receive and process up to N cells (unless a concentrator is employed as explained in (Tobagi 1990)) during a single slot period.

2.1.3 Banyan Networks

A banyan Network consists of several smaller $p \times p$ crossbar switches (*e.g.*, $p = 2$) which are arranged into rows and columns (Fig. 3). The number of crossbar switches depends upon the number of I/O lines of the banyan as well as the dimension (p) of each crossbar. For instance, if the banyan has N external I/O lines and employs 2×2 internal crossbars then each of the $\log_2 N$ columns will be comprised of $N/2$ crossbars resulting in a total of $(N/2)\log_2 N$ crossbars, or equivalently, $2N \log_2 N$ cross-point gates. This amounts to a significant reduction in the implementational complexity for large N. The type of connectivity provided among the different columns ensures that there exists a unique path between every pair of external input and output lines.

The cost reduction of the banyan approach is attained at the expense of a corresponding performance degradation. This degradation is attributable to the so-called *internal-blocking* property of banyans. Internal-blocking refers to a situation where multiple input cells addressed to disjoint output lines could still contend with each other to access the same output port of an internal crossbar which is shared by the routing paths of these cells. To circumvent the high internal cell blocking caused by this phenomenon, buffering is necessary not only at the inputs of the banyan fabric, but also at the inputs of each internal crossbar (intermediate buffers). The performance of the banyan is influenced by (*i*) the number of external I/O lines, (*ii*) the amount of buffering provided for each of the internal queues, and (*iii*) the *window-size* of each crossbar if input windowing (section 2.2.1) were to be employed. Even with the provision of intermediate buffering, the internal cell loss of a banyan could reach unacceptable levels if continued collisions were to occur due to long streams of cells being addressed to fixed outputs (*hot-spot* traffic).

2.2 Schemes for Performance Improvement

The performance (cell blocking and delay) provided by the basic switch architectures discussed above may be improved by certain clever modifications that do not incur a substantial implementational cost. In the following, we discuss a few such enhancements.

2.3 Input Windows in Input-Queued Switches

The drawback due to HOL blocking in crossbar switches can be partially overcome by relaxing the strict FIFO discipline employed within the input queues. Specifically, the input processor is equipped with the functionality to examine a *window* of cells at the heads of various input queues, and select the group of cells to be transmitted during each slot in such a way as to maximize the number of conflict-free output requests (Hluchyj and Karol 1988). The original input-queued switch may be thought of as a special case of this generalization with a window-size of one. It is intuitive that if an arbitrarily large window-size were to be employed, then the throughput of the input-queued switch should approach that of an output-queued switch with identical amount of buffering.

2.3.1 Banyan Networks in Tandem

As mentioned earlier, the uniqueness of path between each I/O pair in a banyan switch leads to unacceptable levels of cell blocking under hot-spot and correlated traffic conditions. One of the methodologies that may be adopted to overcome this drawback is to connect multiple banyans in tandem (Anindo and Seeto 1988). If, for example, two banyan stages were to be connected in tandem, then there will be N possible paths corresponding to each I/O pair, *via* the N outputs of the first stage. In this scenario, the first stage may be employed as a randomization network (to offset the hot-spot and correlation characteristics), and the second stage as a routing network. Specifically, each internal node of the first banyan stage route each incoming cell randomly to one of its outputs (ensuring that collisions do not occur, however). This presents a randomized uniform load at the inputs of the routing stage; the load pattern within the router will be devoid of any harmful correlation characteristics, and hence will result in a superior throughput as compared to the single banyan.

One of the disadvantages of a multistage banyan is the possibility of *out-of-sequence* delivery of packets flowing between various I/O pairs. This could be unacceptable if the system supports real-time applications, where a time-ordered delivery of data cells would be vital for the integrity of information. To overcome out-of-sequence delivery, one may employ a resequencing buffer at each output of the network in conjunction with a sequence number appended to each cell. In the example that we will consider in the sequel, it will be assumed that each input is permanently connected to an output, with the set of I/O associations being disjoint (hot-spot traffic). To describe the operation of the resequencing algorithm within this scenario, let us assume that the sequence number of the last cell delivered by a particular output is n. Upon the arrival of the next cell (with sequence number $n + i$), a decision regarding the deliv-

ery/buffering/rejection is made as follows:

1. If $i < 1$ then the new arrival is rejected immediately.

2. If $i = 1$ then the new arrival (with seq. no. $n+1$ as well as all other cells waiting in the buffer with consecutive sequence numbers $n+2, n+3, \ldots$ are delivered immediately, and n is updated accordingly.

3. If $i > 1$ then

a. If the buffer is not full then the new cell is inserted into the buffer, with the latter being maintained as an ordered list.

b. If the buffer is full then the "oldest" cell from among the occupants of the buffer and the new arrival is rejected; n is set equal to the sequence number of the rejected cell.

It is apparent that the probability of cell rejection is inversely related to the size of the resequence buffer, but a large buffer will also result in a large resequencing delay.

3 TRAFFIC MODELS FOR MULTIMEDIA SOURCES

In this simulation study, we have considered two types of traffic streams. The simplest type of stream considered is comprised of a set of independent identically distributed Bernoulli sources (Fig. 4) applied to the inputs of the switch under investigation. The probability of a cell arrival at an input during each slot is denoted by the parameter p. For a more realistic representation of multimedia sources, we have also considered a correlated source model represented as a 2-state Markov chain (Fig. 5). In this model, it is assumed that the source generates a cell only during those slots during which it is in the ON state. The parameters α and β respectively represent inverses of the mean sojourn times (in number of slots) in the OFF state and the ON state. The long-term probability p of a cell arrival during an arbitrary slot, for this case, is given by $p = \alpha/(\alpha + \beta)$. Hence, by selecting the parameters p and β appropriately, the arrival probability and time-correlation characteristic may be varied independently of each other. Statistical experiments conducted in the past indicate that similar models provide fairly accurate representations for a wide variety of real-time sources such as voice and video.

4 DEVELOPMENT OF THE SIMULATOR

The simulation studies reported herein were conducted on a Sun *SPARCstation*1 workstation using the tool SLAM-II (Simulation Language for Alternative Modeling) (Pritsker 1986) in conjunction with FORTRAN and C. SLAM-II presents a flexible and powerful simulation platform by integrating the event-driven, activity-driven and process-driven approaches for discrete event simulation along with tools for continuous simulation. It provides a comprehensive set of user-callable FORTRAN routines which may be effectively tailored to fit alternate modeling approaches and interfaced with external subroutines.

Since SLAM - II does not feature built-in functions to simulate the architectures and internal operations of ATM switches, it was necessary to code external subroutines to incorporate these functions. In view of the structuring and flexibility of C, this language was selected for coding the ATM switch simulator. This necessitated a careful interfacing of the user-defined C routines to the FORTRAN environment provided by SLAM-II. In particular, the UNIX operating system does not specify any fixed standard for returning the result of a function call from C to FORTRAN. On the other hand, there does exist a standard for procedure or subroutine calls between FORTRAN and C. To circumvent the problem with function calls, one may adhere to the SUN-4 conventions, but this will render the resulting code unportable. To retain portability, the implementation that we adopted was based on directing all function calls from C to the SLAM-II library to a group of external FORTRAN procedures (as per UNIX conventions) which in turn called the SLAM-II functions. Within this setup, the result returned by a SLAM-II function, if any, could be returned by the external FORTRAN module to the C module as a parameter, since all parameters are passed from C to FORTRAN by reference, rather than by value. In this sense, a narrow FORTRAN interface was employed between the SLAM-II platform and the user-encoded C module for ATM switch modeling. On the other hand, whenever SLAM-II procedures were to be accessed, direct calls were made from C to the SLAM-II library as per UNIX conventions.

Within this setup, SLAM-II was used for the startup of the simulator, creation of information sources, setting up of queues, and for keeping track of time and statistics. The C module, on the other hand, set up the connectivities of the different switches, kept track of the number of cells in the initial input queues, moved cells between successive switch stages as per the routing conventions under investigation, measured the delay incurred by each cell and finally,

at the end of the simulation, computed the average cell delay, throughput and other relevant performance measures.

The program was initiated by setting up the switch connections within the C module. This was followed by a transfer of control to the SLAM-II module which initialized the simulator and defined the network operational parameters such as α, β and p. The cell sources were then created in accordance with the specified parameters and cells generated by these sources were routed automatically to their respective input queues. Immediately prior to the entry of each cell to the corresponding queue, the simulation time was recorded and the destination address of the cell was selected. Once every time cycle (the cell transmission time, or slot period) a software interrupt (EVENT, see Fig. 8) was generated, and control was transferred back to the C module (*via* an external FORTRAN routine interface). The C module performed the operations that were necessary to simulate the movement of cells through the switch fabric. In the case of a banyan network, the program first moved the cells waiting in the last stage, then the cells in the last but one stage, and so forth till all the stages were processed. This ensured that a cell would move ahead from its current stage if and the buffer at the next stage had free space, or if a cell in the latter buffer moved ahead in the current time cycle. At each stage, the cell was routed to the upper output or the lower output of the internal crossbar, depending on its address. If a crossbar switch (or an intermediate crossbar of a banyan switch) incorporated input-windowing (lookahead feature), then the corresponding search and selection were also performed at this stage. After completing all the update functions during the current slot, control was returned to the SLAM-II module. The cell-sources within the SLAM-II module were now triggered; they generated fresh cells to join input queues, in accordance with the parameters p, α and β. Upon completion of simulation, control was passed back to the C module and the relevant performance measures were computed.

The generation of cells within the SLAM-II environment was carried out by invoking the "create" function as shown in Fig. 6–7. In the case of correlated sources (Fig. 6), a single cell was generated at the initiation by a "create" node. This cell got multiplied into N cells (N = number of I/O lines of the switch) within the "Multiple Source Generator" (MSG) module; each of the generated cells was assigned a distinct input line number. Following the initiation of these cells, both the create and MSG modules stayed inactive for the rest of the simulation. Each of the generated cells entered the *on* state of the corresponding input line within the "Correlator" block (Fig. 6). The cell that entered the *on* state bifurcated into two, one of which was forwarded to the input queue of the switch. The other one either returned back to the *on* state (with probability of $1-\beta$) or proceeded to the *off* state (with probability β). If a cell was located within the *off* state during a subsequent triggering of the source, then this cell would move to the *on* state with probability $1-\alpha$, or return to the *off* state with probability α. Thus, during each time cycle, the "Correlator block" was reactivated and a cell was forwarded to the appropriate input queue as determined by the system parameters. It may be noted that even though every cell was processed by the same code, SLAM-II guaranteed that the N cell streams were independent of each other. In the case of a Bernoulli source (Fig. 7), the "Create" block remained active throughout the duration of the simulation. A cell was generated during each cycle and this cell got multiplied into N cells. Each of these cells was forwarded either to the respective input with probability p or to a sink with probability $1-p$.

The slot processing performed by the C module for systems that incorporated crossbar switches with input-windowing deserves special consideration. In the following, we concisely describe the processing steps involved in this situation. This description applies to a crossbar switch with N input lines, with the size of each input buffer being b and the window size being w.

1. Record the destination addresses of the first w packets from all input queues in an array of size Nw. If the number of cells in a particular queue is less than w, then record the destination addresses of all the cells from that queue.
2. For $i = 1, \ldots, N$, repeat (*a*), (*b*) and (*c*):

(a) Let j = Number of cells competing for output i.

(b) If $j = 0$, or if the buffer at the next stage is full (applicable only for an intermediate crossbar of a banyan network), then skip through (*c*); else if $j = 1$ then identify the unique cell addressed to output i and move this cell to output i; else select one of the cells competing for output i at random and move to output i.

(c) Mark the input node from which a cell was moved so that no additional cells will be moved from this input during the current cycle (to ensure fairness).

It may be noted that the only difference between an independent crossbar switch and an intermediate

crossbar within a banyan fabric is that the movement of cells to outputs in the latter is conditioned also on the availability of buffer space at the next stage (step 2(*b*)).

5 RESULTS OF THE PERFORMANCE EVALUATION

In this section, we report the results of the simulation studies that we have conducted. We focus on the measures of the cell blocking probability and average cell delay to characterize the performance.

In Fig. 9, we compare the throughput performance of an 8×8 crossbar switch with that of an 8×8 crossbar that employs input-windowing, under Bernoulli traffic conditions. The input buffer size is assumed to be 100. The advantages of input-windowing is apparent from this figure; a window-size of 2 improved the saturation throughput to 0.79 and that of 4 improved the saturation throughput to 0.89, as compared to a value of 0.615 obtained without using input-windowing. Fig. 10 depicts the corresponding delay performance. Again the performance improvement provided by input-windowing is apparent.

We have conducted simulation studies of crossbars with various window sizes under correlated traffic conditions as well. The results for this case, corresponding to the throughput performance did not show any significant difference from those for the case of Bernoulli traffic, depicted in Fig. 9. However, the average cell delays under correlated traffic tends to be significantly higher than those in the case of Bernoulli traffic, as shown by a comparison of Fig. 11 to Fig. 10. This discrepancy, which is most pronounced when input-windowing is not used accounts for the repeated cell collisions that can occur due to the burstiness in the traffic pattern. An interesting observation in this case is that the discrepancy in the delay between the case of correlated traffic and Bernoulli traffic becomes less pronounced as larger window-sizes are used. This indicates that input-windowing is in particular effective in compensating for the adverse effects of burstiness in the traffic pattern.

Fig. 12 illustrates the throughput performance of a $2^3 \times 2^3$ banyan network for various sizes of intermediate buffers and windows. The lowermost curve corresponds to the case where a double buffer is provided at the inputs of each intermediate stage (with the window size being one), which yields a saturation throughput of 0.58. As mentioned earlier, an intermediate cell within the switch can move only if there is adequate space in the buffer at the next stage, or if a cell at the next stage will move during the current cycle. Therefore it is intuitive that increasing the size of the intermediate buffer should improve the throughput, and this is exemplified in Fig. 12. In particular, increasing the intermediate buffer size from two to four resulted in a throughput improvement from 0.58 to 0.69. Fig. 12 also depicts the improvement attained by using different window sizes. A buffer size of two permits window-sizes of up to 2; increasing the window-size from one two in this case resulted in a throughput improvement from 0.58 to 0.64. A buffer size of four permits a maximum window size of four; increasing the window-size from one to four in this case improved the saturation throughput from 0.64 to 0.83.

The delay performance of the banyan network under Bernoulli traffic conditions is illustrated in Fig. 13, with buffer and window sizes being identical to those considered in Fig. 12. Fig. 14 illustrates the variation of the delay parameter for a banyan switch under correlated traffic conditions. A comparison of Fig. 14 to Fig. 13 will show the significantly higher average delays (within the unsaturated region) caused by traffic correlations as compared to those corresponding to the Bernoulli traffic. As in the case of the crossbar switch, usage of larger window sizes (in intermediate crossbars) serves not only to improve the overall performance but also to reduce the discrepancy between the average delays of the correlated and Bernoulli traffic.

As mentioned earlier, the major performance limitation of banyan networks is attributable to the internal blocking of cells caused by the sharing of internal paths. The effect of this phenomenon will be even more pronounced if one were to consider hot-spot traffic (*i.e.* all traffic generated by each input being destined for a fixed output line, with the set of I/O connectivities being conflict-free). This is apparent from the dashed curve shown on Fig. 15 which illustrates the delay performance of an $2^3 \times 2^3$ banyan under hot-spot traffic conditions (note that increasing the sizes of intermediate buffers and windows will not be helpful in this case). As mentioned in Section 2.2.2, one way to circumvent the adverse effects of hot-spot traffic conditions is to employ two identical banyan networks in tandem with the first stage being employed as a randomization network and the second stage as a router network. Fig. 15 also illustrates the delay performance yielded by the tandem banyan arrangement; the performance improvement attained by the latter modification is apparent from this figure.

The performance improvement obtained by employing the tandem organization could, however, lead to out-of-sequence delivery of cells at the destinations, due to the variable delays along the multiple paths.

One strategy to circumvent this drawback would be to employ a resequencing buffer each output stage. The specific details of the resequencing algorithm were described in Section 2.2.2. As part of the simulation effort, we have studied the blocking probability within resequencing buffers of various sizes applied at the outputs of the tandem network considered in Fig. 15. The results of this performance study illustrated in Fig. 16 demonstrate the significance of using a resequence buffer of adequate size to maintain the rejection probability within acceptable levels. In particular, inadequate resequence buffering (*e.g* the case of a buffer size of 10 shown in Fig. 16) could offset the performance improvement yielded by the tandem organization as a majority of cells that pass through the switch will be rejected due to the extent of out-of-sequence delivery being too large to be "cushioned" by the resequencing algorithm.

6 CONCLUDING REMARKS

In this paper we have conducted a performance evaluation of various switching architectures based on the Asynchronous Transfer Mode (ATM) under multimedia traffic conditions. The methodology adopted here is based on discrete-event simulation to facilitate an accurate representation of the architecture of the switches and stochastic behavior of the multimedia sources. The two broad classes of switching systems considered include the basic crossbar switches and the multistage banyan networks. Under each case, we examined various buffering and windowing strategies. The two types of traffic sources considered were the memoryless Bernoulli sources and the the bursty sources, represented as 2-state Markov chains. The latter model is appropriate to accurately characterize the correlation effects observed in multimedia traffic such as voice and video. The results obtained by our simulation effort indicate the deterioration in performance when correlation effects are taken into account. Input windowing may be employed as an effective mechanism to circumvent the drawbacks associated with head-of-line blocking and correlation effects. The results discussed in this paper indicate that apart from providing an overall improvement in delay and saturation throughput, the window-based approach also leads to a significant reduction in the discrepancy in performance delivered to memoryless and bursty traffic sources. A comparison of the basic banyan network and a tandem banyan arrangement under hot-spot traffic conditions has also been conducted as part of this study. Our results illustrate the significant performance improvement provided by the tandem arrangement. However, the latter approach can lead to out-of-sequence delivery of cells at the destinations, and this may be unacceptable in certain real-time applications. One way to remedy this drawback would be to employ a set of resequencing buffers at the outputs of the switch. This scenario has also been simulated to characterize the cell loss in the resequence buffers. The corresponding blocking performance curves underscore the need to use an adequate buffer size in order to take full advantage of the performance improvement obtained with the tandem arrangement.

As part of the future extension to this work, we are investigating the development of analytical models for the scenarios considered in this paper. Several researchers have developed analytical models for certain classes of switching systems under Bernoulli traffic conditions. However, as noted here, the performance under multimedia traffic conditions differ significantly from that predicted by the Bernoulli models, due to the correlation characteristics of the former. In view of this, the major challenge ahead would be to develop approximate analytical models for the basic crossbar and banyan switches subject to correlated traffic, and to extend these models to the case of input-windowing.

REFERENCES

Anido, G.J., and A.W. Seeto. 1988. Multipath Interconnection: A Technique for Reducing Congestion Within Fast Packet Switching Fabrics. *IEEE Journal on Selected Areas in Communications* 6: 1480-1488.

Karol, M.J., M.G. Hluchyj, and S.P. Morgan. 1987. Input Versus Output Queueing on a Space-Division Packet Switch. *IEEE Transactions on Communications* 35: 1347-1356.

Hluchyj, M.G., and M.J. Karol. 1988. Queueing in High-Performance Packet Switching. 1988. *IEEE Journal on Selected Areas in Communications* SAC-6: 1587-1597.

Jenq, Y-C. 1983. Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network. *IEEE Journal on Selected Areas in Communications* SAC-1: 1014-1021.

Pritsker, A.A.B. 1986. *Introduction to Simulation and SLAM-II*, 3rd Edition. New York: Halsted Press.

Szymanski, T., and S. Shaikh. 1989. Markov Chain Analysis of Packet-switched Banyans with Arbitrary Switch sizes, Queue sizes, Link Multiplicities and Speedups. In *Proceedings of the IEEE INFOCOM*, 960-971. Ottawa, Canada.

Tobagi, F.A. 1990. Fast Packet Switch Architec-

tures for Broadband Integrated Services Digital Networks. 1990. *Proceedings of the IEEE* 78: 133–167.

AUTHOR BIOGRAPHIES

SIDDHARTH BEHERA received his Bachelor's degree in Computer Science and Engineering from M.S. University of Baroda, Baroda, India, in 1988. He is currently a graduate student in the Department of Computer Science and Engineering, University of Nebraska, Lincoln, N.E. His research interests are in the performance evaluation of switching architectures, simulation and queueing theory.

GOPAL MEEMPAT received his Bachelor's and Master's degrees in Electrical and Communication Engineering from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in Electrical Engineering from the University of Arizona, Tucson, A.Z., in 1989. Since 1989 he has been an Assistant Professor in the Department of Computer Science and Engineering, University of Nebraska, Lincoln, N.E. His research interests span the areas of broadband ISDN's, performance modeling, queueing, simulation and optimization.

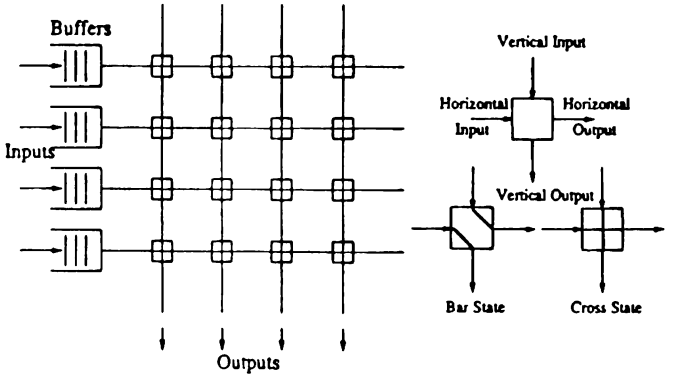

Figure 1: The Crossbar Switch Architecture

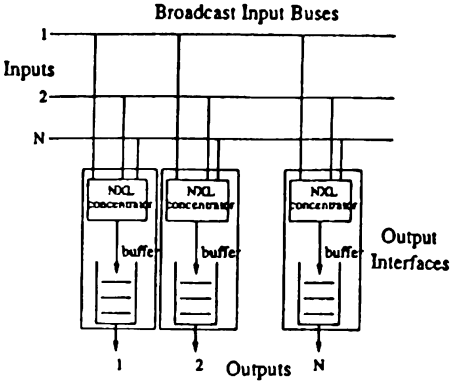

Figure 2: The Knockout Switch Architecture

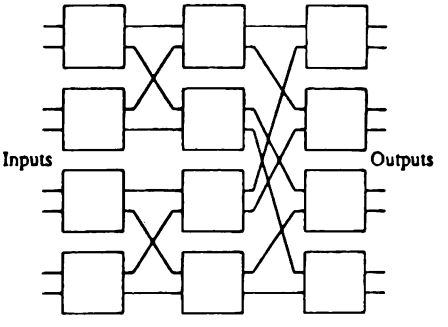

Figure 3: A $2^3 \times 2^3$ Banyan Network

Figure 4: The Bernoulli Source Model

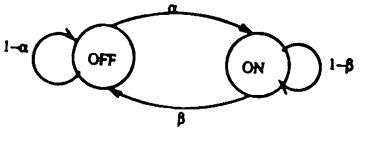

Figure 5: The Correlated Source Model

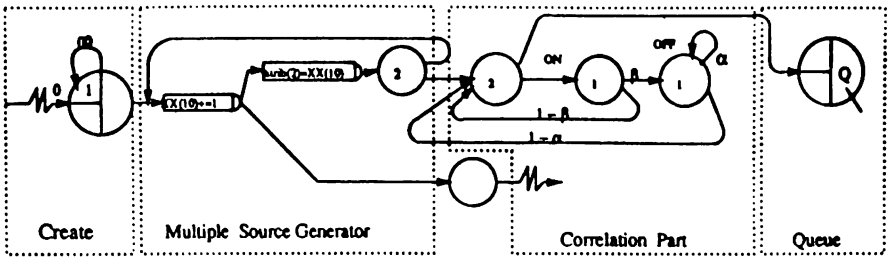

Figure 6: SLAM-II Model for Correlated Source

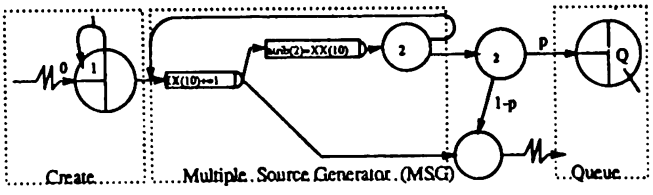

Figure 7: SLAM-II Model for Bernoulli Source

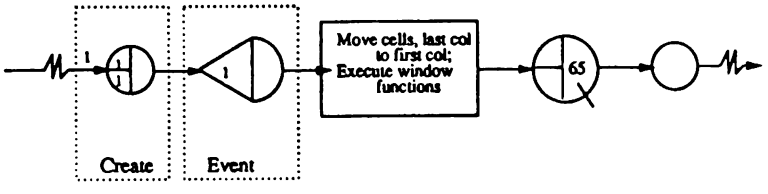

Figure 8: Software Interrupt for Slot Processing

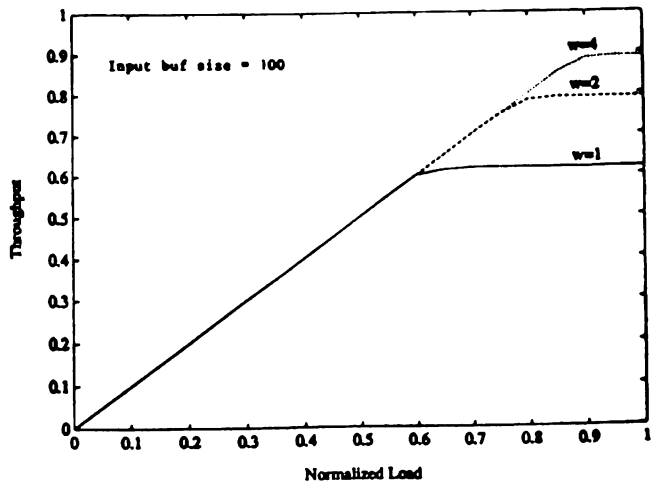

Figure 9: Throughput Performance of Crossbar Switch

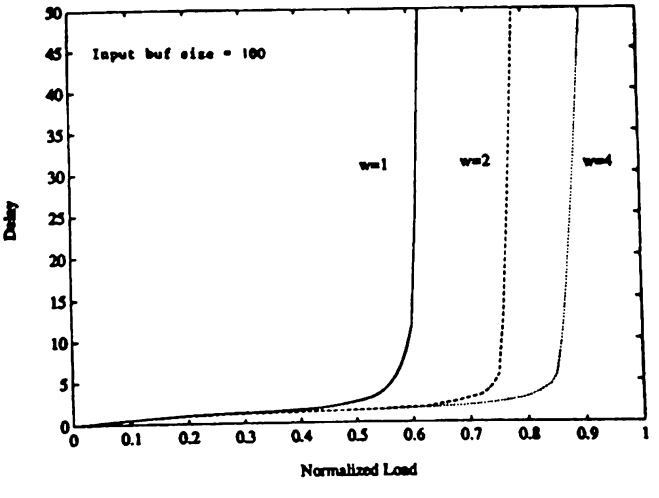

Figure 10: Delay Performance of Crossbar Switch (Bernoulli Source)

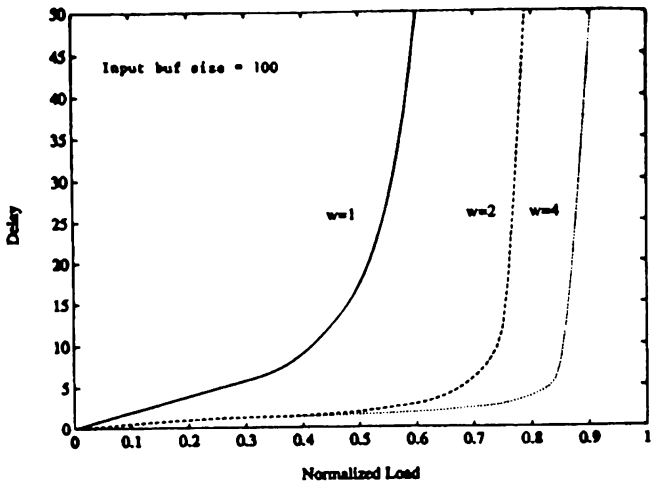

Figure 11: Delay Performance of Crossbar Switch (Correlated Source)

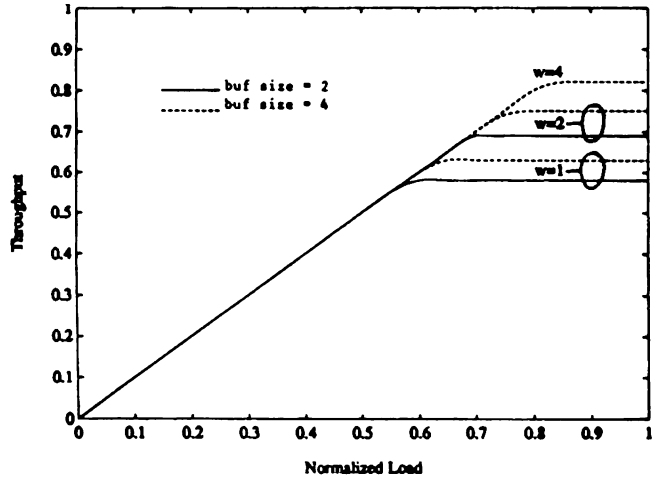

Figure 12: Throughput Performance of Banyan Network

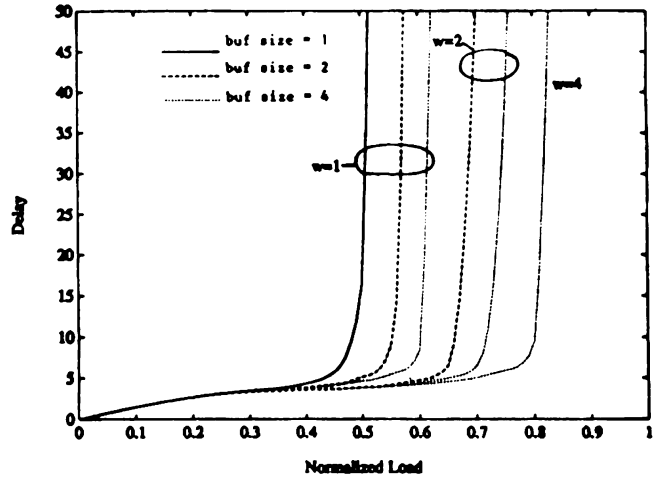

Figure 13: Delay Performance of Banyan Network (Bernoulli Source)

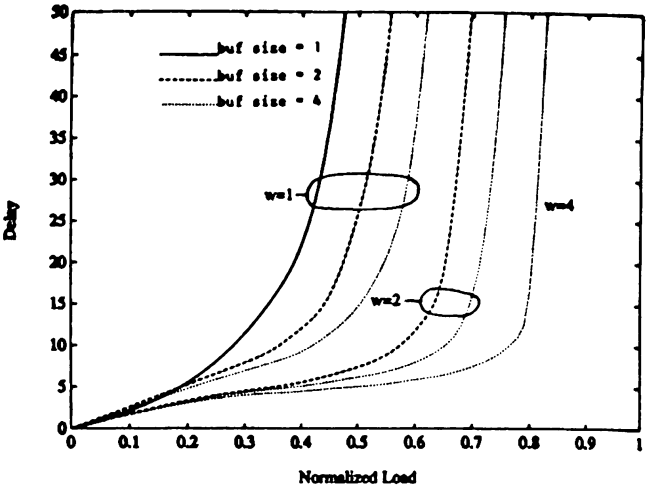

Figure 14: Delay Performance of Banyan Network (Correlated Source)

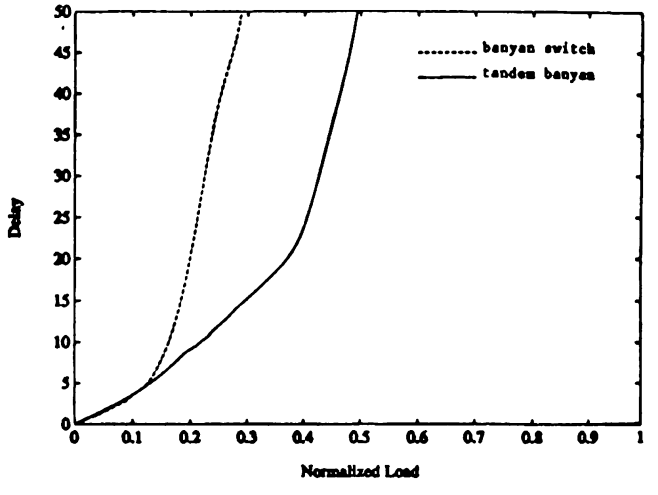

Figure 15: Delay Performance: Banyan Switch vs. Tandem Banyan

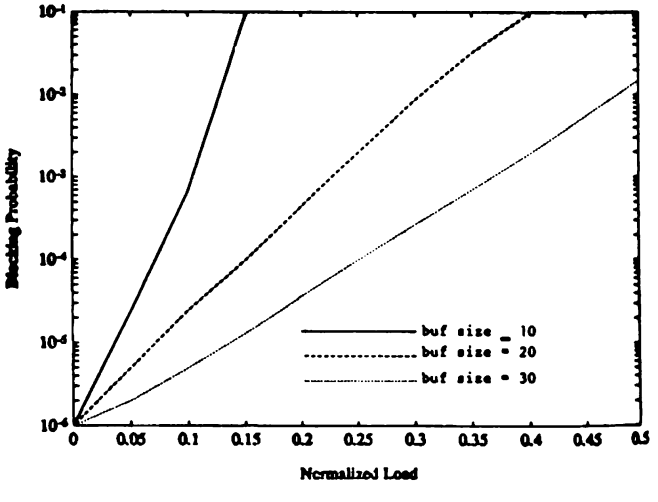

Figure 16: Blocking Probability at the Resequencing Buffer