the interconnected learning automata. The genetic adaptation and population-based approach also helps the interconnected learning automata to escape from local optima. The system seems to be particularly beneficial to problems involving large action spaces where improved convergence is most required.

Due to the high degree of variability inherent at the initial stages of the GLA, the influence to which the crossover operator has on the population is difficult to assess and further studies need to be undertaken to determine if it is, as expected, providing a valuable role. The small study performed here does indicate that it performs a useful function with uniform crossover giving superior convergence results in some instances.

Furthermore, a learning automata rule which is ergodic and with no absorbing states, such as the linear reward/penalty, may be better suited to the genetic adaptation of learning automata for some environments such as nonstationary ones. Further work will also provide a firm theoretical basis for the convergence of the GLA.

References

[1] L. Chambers, Ed., *The Practical Handbook of Genetic Algorithms*. London, U.K.: Chapman & Hall, 2001.

[2] C. L. Karr and L. M. Freeman, Eds., *Industrial Applications of Genetic Algorithms*. Boca Raton, FL: CRC, 1999, International Series on Computational Intelligence.

[3] M. L. Tsetlin, "On behavior of finite automata in random media," *Automat. Telemekh.*, vol. 22, pp. 1345–1354, Oct. 1961.

[4] N. Baba, *New Topics in Learning Automata Theory and Applications*, M. Thoma, Ed. New York: Springer-Verlag, 1984, Lecture Notes in Control and Information Sciences.

[5] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[6] K. Najim and A. S. Poznak, *Learning Automata—Theory and Applications*. New York: Pergamon, 1994.

[7] K. S. Narendra and R. M. Wheeler, "An N-player sequential stochastic game with identical payoffs," *IEEE Trans. Syst., Man, Cybern.*, vol. 13, 1983.

[8] A. A. Hashim, S. Amir, and P. Mars, "Application of learning automata to image data compression," in *Adaptive and Learning Systems: Theory and Applications*, K. S. Narendra, Ed. New York: Plenum, 1986.

[9] G. P. Frost, T. J. Gordon, M. N. Howell, and Q. H. Wu, "Reinforcement learning of active and semi-active vehicle suspension control laws," *Proc. Inst. Mech. Eng. A, Power Process Eng.*, vol. 210, pp. 249–257, 1996.

[10] Q. H. Wu, "Learning coordinated control of synchronous machines in multimachine power systems," in *Proc. IEEE Syst., Man, Cybern. Conf.*, vol. 3, 1993, pp. 728–733.

[11] C. K. K. Tang and P. Mars, "Games of stochastic learning automata and adaptive signal processing," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 851–856, May/June 1993.

[12] S. Baluja, "Population-based incremental learning: A method of integrating genetic search-based function optimization and competitive learning," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, 1994.

[13] S. Baluja and R. Caruana, "Removing the genetic from the standard genetic algorithm," in *Proc. 12th Int. Conf. Machine Learning*, July 1995, pp. 38–46.

[14] M. Munetomi, Y. Takai, and Y. Sato, "StGA: An application of genetic algorithm to stochastic learning automata," *Syst. Comput. Jpn.*, vol. 27, p. 68–78, Sept. 1996.

[15] K. R. Ramakrishnan, "Hierarchical systems and cooperative games of learning automata," Ph.D. dissertation, Dept. Elect. Eng., Indian Inst. Sci., Bangalore, 1982.

[16] A. Lakshmivarahan, "A learning approach to the two-person decentralized team problem with incomplete information," *Appl. Math. Comput.*, vol. 8, pp. 51–78, 1981.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[19] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 2000.

[20] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 2–9.

[21] W. M. Spear and K. A. De Jong, "An analysis of multipoint crossover," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 301–315.

[22] T. Bäck, "Selective pressure in evolutionary algorithms: A characterization of selection methods," in *Proc. 1st IEEE Conf. Evolutionary Computation*. Piscataway, NJ: IEEE Press, 1994, pp. 57–62.

[23] D. B. Fogel and A. Ghozeil, "A note on representations and variation operators," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 159–161, 1997.

[24] Z. Michalewicz, *Genetic Algorithms $+$ Data Structures $=$ Evolutionary Programs*. New York: Springer-Verlag, 1992.

[25] D. H. Ackley, "An empirical study of bit-vector function optimization," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. New York: Pitman, 1987, pp. 170–204.

[26] L. J. Eshelman and J. D. Schaffer, "Crossover's niche," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 9–14.

[27] K. A. De Jong, "The analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1975.

Learning Automata-Based Bus Arbitration for Shared-Medium ATM Switches

Mohammad S. Obaidat, Georgios I. Papadimitriou, Andreas S. Pomportsis, and Haralambos S. Laskaridis

***Abstract*—Although new high-bandwidth network technologies are being introduced and widely deployed, asynchronous transfer mode (ATM) is still considered one of the most important network technologies currently in use. A number of ATM switches architectures have been proposed in the literature. However, industry has shown that is better to use the well-known shared-medium technique in the design of these ATM switches. In this paper, four variations of a new distributed scheme are proposed for the arbitration of a shared bus of an ATM switch. These schemes are based on learning automata. Taking advantage of the bursty nature of ATM traffic, the new arbitration scheme shows a superb performance compared to the time division multiple access (TDMA) scheme.**

***Index Terms*—Asynchronous transfer mode (ATM) switches, Asynchronous transfer mode (ATM) systems, improved LABA (iLABA), learning automata-based bus arbitration (LABA), learning automata, random TDMA (RTDMA), shared bus, time division multiple access (TDMA).**

I. Introduction

Asynchronous transfer mode (ATM) was intended to be the switching and multiplexing technique used in the implementation of broadband integrated services digital networks (B-ISDN). Accompanied by a stack of protocols including signaling protocols, ATM was planned to provide end-to-end connectivity with quality of service

Manuscript received November 15, 2001; revised March 1, 2002.

M. S. Obaidat is with the Department of Computer Science, Monmouth University, West Long Branch, NJ 07764 USA (e-mail: obaidat@monmouth.edu).

G. I. Papadimitriou, A. S. Pomportsis, and H. S. Laskaridis are with the Department of Informatics, Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece (e-mail: gp@csd.auth.gr; apombo@csd.auth.gr; haris@csd.auth.gr).

Publisher Item Identifier S 1083-4419(02)06471-3.

(QoS) satisfying the diverse requirements of applications. During the last couple of years, new network technologies, such as Packet over SONET and Gigabit Ethernet, were introduced and widely deployed in the core of large ISPs, providing higher bandwidth connections. However, ATM remains one of the main network technologies in campus networks and in the access network connecting campus or metropolitan networks to the core networks.

Extensive research efforts have been focused on ATM switching systems. Based on the architecture of the switch fabric, ATM switches can be divided into the following four categories [1].

1) *Shared-memory switches*: Examples of industry products include Cisco's lightstream and catalyst product families [2] and Marconi's ASX product family [3].
2) *Shared-medium switches*: Examples can be found in [4] and [5]. Moreover, Marconi ASX-4000 [3] uses a TDM bus to interconnect multiple "switch control processors" installed in the same chassis.
3) *Fully interconnected switches*: The crossbar switch is the most common architecture of this category. Variations of the crossbar switch can be found in [6] and [7].
4) *Space-division switches or multistage interconnection networks (MINs)*: Numerous switches belonging to this category were presented in the literature (e.g., [8]–[12]). Most of them are based on the Banyan multistage interconnection network.

Although a lot of research has been conducted on space-division switches, industry still shows that shared-medium and shared-memory architectures are preferable due to the ease of implementation and low hardware complexity and cost.

Bus arbitration schemes can be divided into two main categories.

1) Schemes with central arbitrator or "queues-state dependent." Algorithms used in such schemes use a central arbiter and have a complexity of at least $O(N)$. They select the input port that is granted permission to transmit over the shared-medium during each circle.
2) Schemes with no central arbitrator. A distributed algorithm running on each input port is responsible for the medium access control.

In this paper, we introduce a new learning automata-based bus arbitration (LABA) scheme, which is based on a fair distributed algorithm running on each input port. Although centralized control (arbitrator) is not employed, the bursty nature of ATM traffic is taken into consideration when deciding which input port will be granted permission to transmit. In comparison with previous related work [5], [6], this paper presents an enhanced, simpler algorithm, as well as the advantages of adopting modularity in the switch design.

The rest of the paper is organized as follows. In Section II, two variations of the bus arbitration scheme are presented along with efficient ways of implementing the algorithms. In Section III, modular versions of the two basic LABA schemes are presented in order to investigate the scalability of the proposed scheme. In Section IV, extended simulation results are presented in order to demonstrate the superiority of the proposed architectures in comparison to other well-known arbitration schemes without centralized control. Simulation results are also used to compare the four variations with each other and demonstrate the effectiveness of using modular architectures. Finally, Section V concludes the paper.

II. The LABA Switch

We consider an ATM switch with N input and N output ports (denoted as $I_1, \ldots, I_N$ and $O_1, \ldots, O_N$, respectively), with each port running at R Mb/s. Ports are interconnected over a shared bus (as depicted in Fig. 1) running at B Mb/s. Therefore, the bus has a speed-up factor of $S = B/R$, where S is an integer. It will be shown in Section IV that S can be less than N, but not much smaller. Each input and output port has a buffer with a capacity of Q cells, which is necessary due to the speed-up factor of $S > 1$ and the absence of central arbiter.

Each input port is equipped with a discretized learning automaton [13]–[16]. The N learning automata, operate concurrently to offer a distributed arbitration scheme. The exact operation of the learning automata is described in the following subsection. A number of N learning automata, one for each input port, are used instead of one central learning automaton, in order to avoid having a single point of failure and avoid delay and complexity due to communication between the bus arbitrator (LA) and the input ports.

The proposed architecture has the following characteristics:

1) it uses simple shared-medium architecture;
2) it enables easy implementation of switching broadcast and multicast traffic;
3) it has superior performance in comparison to other well-known bus arbitration schemes, under bursty traffic;
4) it offers fair share of the bus's bandwidth among the active input ports, i.e., ports with incoming traffic.

A. The LABA Scheme

As mentioned above, the bus arbitration scheme is based on N learning automata, which are initialized simultaneously, operate concurrently and independently, and use as the only feedback information the fact of whether a cell was transmitted over the bus by the input port that was granted permission to transmit in the previous time slot. By using the same feedback information, they grant permission to the same input port to transmit at each time slot.

Each learning automaton contains the same probability distribution $P(t)$ over the set of input ports. Thus, $P(t) = \{P_1(t), P_2(t), \ldots, P_N(t)\}$, where $P_j(t)$ is the choice probability of input port I_j at time slot t. Each probability $P_j(t)$ can take $k+1$ discrete values:

$P_j(t) \in \{0, 1/k, 2/k, \ldots, 1\}$ where k is an integer.

During each time slot, an input port I_m is selected in the following way in order to be granted permission to transmit.

1) If $\sum_{n=1}^{N} P_n(t) = 0$, then I_m is selected randomly under uniform distribution.
2) If $\sum_{n=1}^{N} P_n(t) \neq 0$, then I_m is selected randomly conforming to the normalized probability distribution $\Pi(t) = \{\Pi_1(t), \Pi_2(t), \ldots, \Pi_N(t)\}$ where: $\Pi_j(t) = P_j(t)/\sum_{n=1}^{N} P_n(t)$.

Selection takes place regardless of whether or not the input port has a cell to transmit. Thus, during a time slot, the bus can be either busy or idle. This is used as feedback information for the choice probability update of the selected input port

$$\begin{aligned} P_m(t+1) &= P_m(t) + \frac{1}{k}, \quad \text{if bus is busy and } P_m(t) < 1 \\ P_m(t+1) &= P_m(t) - \frac{1}{k}, \quad \text{if bus is idle and } P_m(t) > 0 \\ P_m(t+1) &= P_m(t), \quad \text{otherwise.} \end{aligned}$$

Since simultaneous initialization of the learning automata is a prerequisite for the operation of the distributed arbitration scheme, and in order for the switch to be "hot-swappable," reinitialization of all learning automata is mandatory when new ports are inserted in the switch while operating.

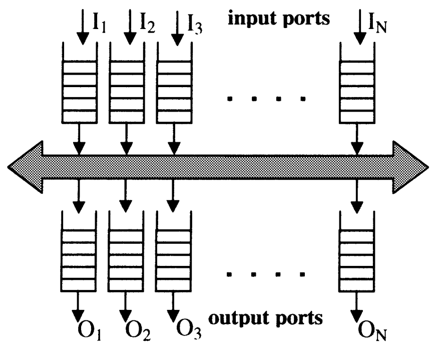

Fig. 1. Combined input/output queuing shared bus switch.

B. The iLABA Scheme

The algorithm presented in the previous subsection seems to have a defect: when choice probability of input port I_j returns to zero, after a period of inactivity, I_j may be granted permission to transmit again only when all choice probabilities return to zero (i.e., when $\sum_{n=1}^{N} P_n(t) = 0$), regardless of whether or not there are arrivals in port I_j. In the following, we introduce a variation of the basic scheme that we call the *improved LABA (iLABA).*

In iLABA, each probability $P_j(t)$ can take k discrete values: $P_j(t) \in \{1/k, 2/k, \ldots, 1\}$. Thus, choice probabilities are never set to zero. During each time slot, an input port is granted permission to transmit by selecting a random I_m conforming to the normalized probability distribution $\Pi(t) = \{\Pi_1(t), \Pi_2(t), \ldots, \Pi_N(t)\}$ where: $\Pi_j(t) = P_j(t)/\sum_{n=1}^{N} P_n(t)$

When the learning automata are initialized, all choice probabilities are set to $1/k$. We will comment on the differences of the two variations of the arbitration scheme in Section IV, based on the simulation results.

C. Implementing the Bus Arbitration Schemes

A fast algorithm of constant complexity, independent of N, is now presented for the implementation of the following two operations during each time slot: 1) selection of port I_m to be granted permission to transmit over the shared bus, and 2) choice probability $P_m(t)$ update.

We consider a variable length array A that consists of "probability cells." Each probability cell represents a probability mass of $1/k$ and belongs to a specific input port. The number of probability cells in A belonging to input port I_j is proportional to $P_j(t)$. Therefore, in order to select an input port according to probability distribution $\Pi(t)$, we can select one of the probability cells in A at random following the uniform distribution. The owner's input port of the selected probability cell is granted permission to transmit. When the choice probability of I_j has to be increased by $1/k$, a probability cell is added at the end of array A, with owner I_j. If the choice probability of I_j has to be decreased by $1/k$, a probability cell with owner I_j is removed from array A.

Two versions of the algorithm are presented below, corresponding to the variations of the arbitration scheme: LABA and iLABA. In the iLABA algorithm, array A is initialized with one probability cell belonging to each of the input ports, whereas in the LABA algorithm, A is initialized with no probability cells. In these algorithms, function "Random" produces a random real number in the range $[0, 1]$, using uniform distribution. However, in order to reduce the computational complexity, this function can be implemented using precomputed random numbers. Function Bus(t) returns one of two values: "busy" or "idle," representing the status of the shared bus during time slot t.

```
Procedure LABA_Input_port_j
{- - - Initialization - - -}
For i := 1 to N.
  number_of_cells[i] := 0
A_size := 0
{- - - Normal operation - - -}
For t := 1 to +∞
  If A_size > 0 then
    index := ⌊Random ⋆ A_size⌋
    m := A[index]
  Else
    m := ⌊Random ⋆ N⌋
  I_m transmits if it has cell to transmit
  If (bus (t) = busy) and (number_of_cells[m] < k)
    Inc (A_size)
    A[A_size] := m
    Inc (number_of_cells[m])
  Else if (bus (t) = idle) and (number_of_cells[m]>0)
    A[index] := A[A_size]
    Dec (A_size)
    Dec (number_of_cells[m])
```

```
Procedure improved_LABA_Input_port_j
{- - - Initialization - - -}
For i := 1 to N
  A[i] := i
  number_of_cells[i] := 1
A_size := N
{- - - Normal operation - - -}
For t := 1 to +∞
  index := ⌊Random ⋆ A_size⌋
  m := A[index]
  I_m transmits if it has cell to transmit
  If (bus(t) = busy) and (number_of_cells[m] < k)
    Inc (A_size)
    A[A_size] := m
    Inc (number_of_cells[m])
  Else if (bus(t) = idle) and (number_of_cells[m] > 1)
    A[index] := A[A_size]
    Dec (A_size)
    Dec (number_of_cells[m]).
```

Apparently, iLABA is simpler and can be implemented using fewer processor's cycles. The serial traversal of array A is not required in either LABA or in iLABA. Therefore, the complexity of the algorithms is not proportional to either N or k; the complexity is constant. This fact provides a great advantage regarding scalability, in contrast to existing learning automata, which have complexity $O(N)$.

III. The Modular LABA Switch

In order to increase the scalability of the proposed architecture, we propose in this section a modular variation of the basic architecture described earlier in this paper.

The switch is composed by M modules, interconnected over a shared backplane bus via module interfaces installed in each module, as depicted in Fig. 2. Each module serves P ports. The P input ports, P output ports, and the module interface are connected to the shared module bus. Both the modules' buses and the backplane bus operate using the LABA or the iLABA scheme.

In this model, each module bus operates using a speed-up factor of $S_1 < P$, while the backplane bus operates at speed $S_2 < M$ times

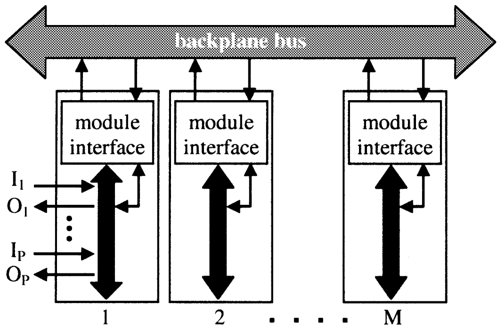

Fig. 2. Modular switch architecture.

faster than the modules' buses, i.e., using a speed-up factor of $S_1 \times S_2 < N$. The notation "$S_1 \times S_2$" will be used from now on to describe the speed-up of the modular LABA (M-LABA) and modular improved LABA (M-iLABA) architectures.

Given that there are no "hot points," it is obvious that $1/M$ of the traffic is switched locally within each module, without using the backplane bus. Thus, both the backplane bus and the modules' buses can operate using the speed-up factor less than the corresponding speed-up factor of the basic (nonmodular) architecture. This has been proven by our simulation results presented in the next section.

IV. Simulation Results

A. Input Traffic Model

Our simulation model is a discrete time model (i.e., time is slotted). During each time slot, a cell may arrive in each input port. In [17], the bursty arrival process to each one of the input queues was modeled using the interrupted Bernoulli process (IBP). An input port can be either in an active state (1) for a geometrically distributed time period or in an idle (0) state. During an active period, in each time slot, a cell arrives at the input port with probability p. By setting $p = 1$, we turn the IBP model into the "ON/OFF model," which is used in the following. At the end of each time slot, an input port may change state. An active input port changes to idle state with probability P_{10} or remains in active state with probability $1 - P_{10}$. An idle input port changes to active state with probability P_{01} or remains idle with probability $1 - P_{01}$, respectively.

According to the formulas used in [18] where the IBP model is used, if $R \in [0, 1]$ is the average load offered to each input port, then

$$R = P_{01}/(P_{01} + P_{10}). \tag{1}$$

If the mean burst size is B cells, then

$$P_{10} = 1/B. \tag{2}$$

From (1) and (2), it can be deduced that

$$P_{01} = R/(B(1 - R)). \tag{3}$$

In our simulation model, we also take into consideration the correlation of cells belonging to the same burst. Given that a burst may be an IP packet segmented in a number of ATM cells, it follows that packets entering the switch from a specific input port belonging to the same burst are likely to have the same destination port. In order to model this fact, we use the correlation factor C in the following way [19]: the first packet of a burst is destined to any of the N output ports with probability $1/N$. The rest of the cells belonging to the same burst are destined to the same output port with probability C and to any other of the $N - 1$ output ports with probability $(1 - C)/(N - 1)$.

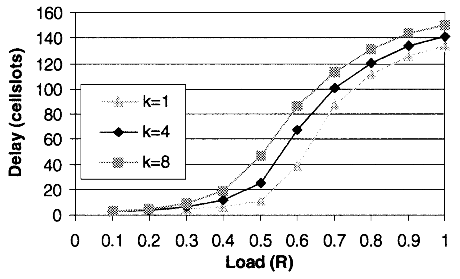

Fig. 3. LABA: Delay for various k values ($N = 32, S = 20$).

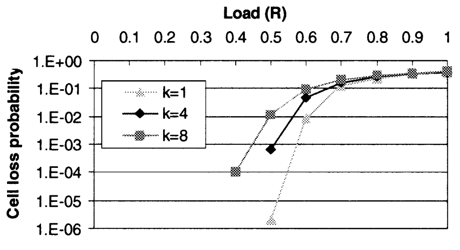

Fig. 4. LABA: Cell loss probability for various k values ($N = 32, S = 20$).

We have run several simulation experiments for different values of k, N, S, and R. We have simulated all LABA, iLABA, modular LABA, and modular iLABA schemes as well as time division multiple access (TDMA) and random TDMA (RTDMA) architectures. In all simulations, we have set $B = 32$ cells, $Q = 128$ cells, and $C = 0.2$. Furthermore, in all modular LABA and iLABA switches, we have used $P = 8$ (i.e., each module has eight ports).

B. Optimal k Values

Figs. 3 and 4 show the delay (in cell slots) and the cell loss probability, respectively, of LABA switches with $N = 32$ and $S = 20$, for various values of k, while R ranges from 0.1 to 1. Obviously, $k = 1$ is the optimal k value.

On the other hand, Figs. 5 and 6 show the delay and the cell loss probability, respectively, of iLABA switches with $N = 32$ and $S = 20$, for various values of k. It appears that the switch with $k = 8$ performs slightly better than switches with other values of k. Apparently, k has a minor effect on the performance of iLABA, compared to the effect of k on LABA.

The fact that the optimal value for LABA is $k = 1$ was intuitively expected. In LABA, when the choice probability of an input port is set equal to zero, this input port may be granted permission to transmit if and only if the choice probabilities of all input ports are set to zero. This turns more into a problem as k increases. When k is set to one, the arbitration scheme follows more of a binary logic dividing the ports into active and inactive ports and sharing the bandwidth fairly among the active ports. Furthermore, if $\sum_{n=1}^{N} P_n(t) = 0, I_m$ is selected at random, and I_m has cells to transmit, then no other input port may be granted permission to transmit, until I_m transmits all the cells it has in its input buffer. Thus, LABA could be characterized as "greedy."

The iLABA scheme, on the other hand, is more "granular," if $k > 2$ (iLABA with $k = 2$ has identical behavior to LABA with $k = 1$). More heavily loaded ports take a larger portion of the bandwidth than lightly loaded ports, as they have greater choice probabilities.

The optimal k values will be used in the rest of the simulation results presented hereafter: $k = 1$ for LABA and M-LABA and $k = 8$ for iLABA and M-iLABA.

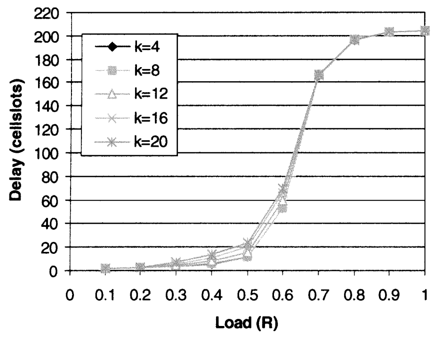

Fig. 5. Delay for various k values $(N = 32, S = 20)$ for the iLABA case.

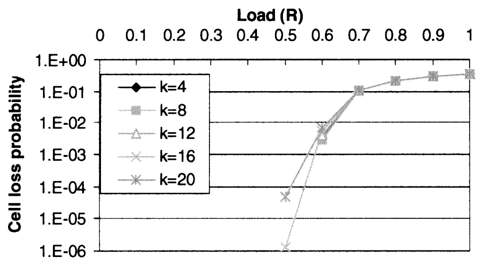

Fig. 6. Cell loss probability for various k values $(N = 32, S = 20)$ for the iLABA case.

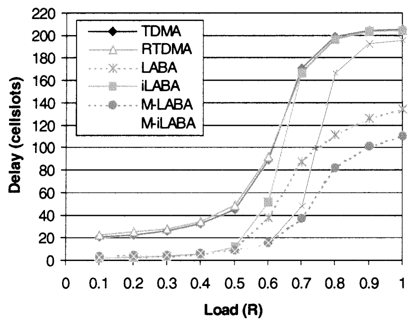

Fig. 7. Delay for $N = 64, S = 40$ switches—various arbitration schemes.

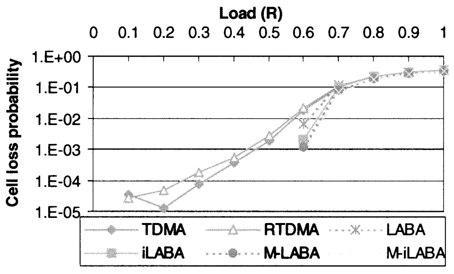

Fig. 8. Cell loss probability for $N = 64, S = 40$—various arbitration schemes.

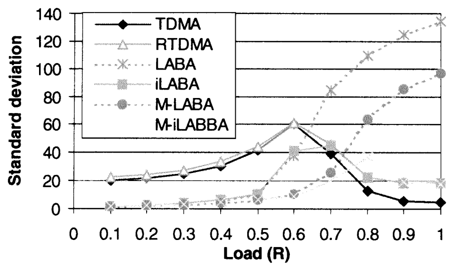

Fig. 9. Standard deviation of delay considering various arbitration schemes for $N = 32, S = 20$ or 6×3.

C. Performance Comparison

We have run several simulation experiments with $N = 32$ and $S = 20$ as well as with $N = 64$ and $S = 40$ for all of the above-mentioned arbitration schemes. Results concerning delay and cell loss probability are presented in Figs. 7 and 8, respectively, for $N = 64$, whereas results for $N = 32$ proved to be identical and are not reported here.

We have to note that in our simulations the M-LABA and M-iLABA switches with $N = 32$ have four modules with eight ports each and $S_1 = 6, S_2 = 3$. The M-LABA and M-iLABA switches with $N = 64$ have eight modules with eight ports each and $S_1 = S_2 = 6$. Thus, in both cases $S_1 \times S_2$ is less than S used in other arbitration schemes, yet both modular switches perform significantly better than the non-modular counterparts, as the traffic load is distributed over a number of buses. Moreover, we can make the following observations.

We define the "critical point" of each scheme as the minimum value of R, where excessive cell loss probability of approximately 0.1 appears. Prior to the critical points (i.e., $R = 0.7$ for LABA and iLABA and $R = 0.8$ for M-LABA and M-iLABA), all four LABA variations have significantly better performance than TDMA (Time Division Multiple Access) and RTDMA (Random Time Division Multiple Access), in terms of both delay and cell loss probability. LABA family schemes' superiority over TDMA and RTDMA is due to the fact that burstiness is taken under consideration: the use of the shared bus during the past time slot is used as feedback information. This feature is not used in TDMA and RTDMA where there is no feedback information of any kind.

Prior to the critical points, iLABA and M-iLABA have smaller loss probability than their counterparts LABA and M-LABA. As explained in SectionIV-B, iLABA is more granular than LABA and bus's bandwidth is more fairly shared among active ports according to their load. It is a little harder for LABA switch to realize that a previously inactive port I_m changed state to active if $P_m(t) = 0$. This leads to slightly higher loss probability than iLABA.

Up to the critical points, minus 0.1 iLABA and M-iLABA have the same delay as LABA and M-LABA, respectively. After this point, iLABA and M-iLABA follow the steep incline of TDMA and RTDMA, while LABA and M-LABA have different behavior. We estimate that this is again due to greedy behavior of LABA: when an input port is found active and is granted permission to transmit, it will keep transmitting until its input buffer is emptied. Under high load, when input buffers are highly utilized, this behavior is advantageous to average delay, as some cells experience very low delay.

In the following, we investigate the standard deviation of delay, as a "fairness" metric. Results presented in Fig. 9 show that TDMA, RTDMA, iLABA, and M-iLABA have similar behavior: delay's standard deviation increases as delay increases, it has a maximum value at the critical point and then decreases. LABA and M-LABA, on the other hand, have different behavior: delay's standard variation increases as delay increases, throughout the range of R.

For R smaller than the critical point minus 0.1, simulation results for LABA and iLABA are almost equal (for $0.1 \leq R \leq 0.6$). The same holds for M-LABA and M-iLABA (for $0.1 \leq R \leq 0.7$). Therefore, the greedy behavior of LABA does not seem to affect delay variation, which is an obvious metric to be used as fairness metric. On the other hand, TDMA and RTDMA experience significantly higher delay variation up to their critical point minus 0.1 $(R = 0.6)$. However, the steep increase of LABA delay's standard deviation beyond the critical point probably reveals a "fairness problem" in this range.

Thus, in the same range of high loads, a large difference between LABA and iLABA exists regarding both delay and delay's standard

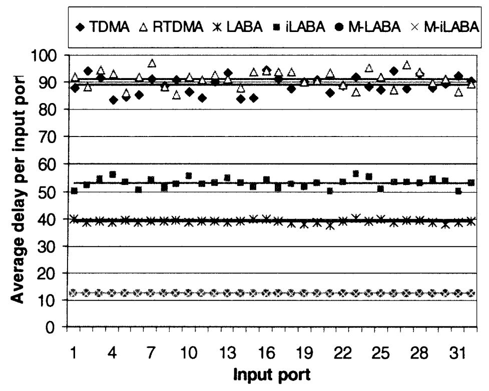

Fig. 10. Average delay of cells per input port ($N = 32, S = 20$ or 6 × 3).

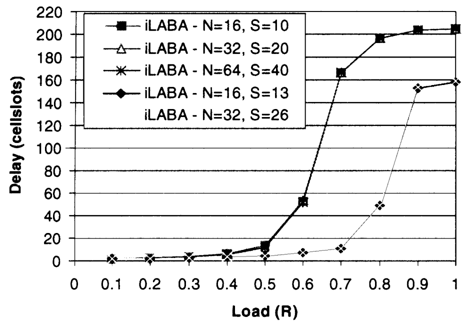

Fig. 11. Delay for iLABA scheme for various values of N and S.

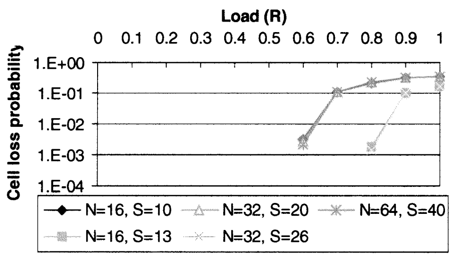

Fig. 12. Loss probability for iLABA scheme for various values of N and S.

variation. The former is in favor of LABA, while the latter is in favor of iLABA. However, the usefulness of the simulation results beyond the critical points are doubtful, because the corresponding range could be characterized as "nonoperational."

Finally, in order to investigate fairness among ports, we calculated the average delay experienced by cells of each input port. The results for all switches are plotted in Fig. 10. It can be observed that the higher the total average, the higher the deviation of the port averages. LABA schemes seem more fair in the arbitration among ports, while modular schemes are the fairest among LABA family schemes.

D. Scalability Analysis

In order to investigate the scalability of the proposed scheme, we have run several simulation experiments of the iLABA scheme for different values of N (16, 32 and 64) keeping the S/N ratio constant, equal to 0.625. As expected, the results of both delay and cell loss probability presented in Figs. 11 and 12 prove that performance is the same when increasing the number of ports provided that the speed-up factor is increased proportionally. We have also run simulations for slightly higher values of S, increasing the S/N ratio to 0.8125. We can observe that by increasing S by 30%, delay can be decreased even by 93% and loss probability can be decreased even by 2 orders of magnitude.

Similar simulation experiments were run for the M-iLABA switch for different values of S_1 and S_2. Again, results show that, when increasing the number of ports (modules), delay and cell loss remain almost the same provided that S_2 increases proportionally.

V. Conclusion

To conclude, we have presented four variations of a new distributed bus arbitration scheme, based on learning automata located in each input port, which take into consideration the burstiness of ATM traffic. We have presented extensive simulations that reveal the superior performance of the proposed schemes compared to other well-known noncentralized arbitration schemes: TDMA and RTDMA. However, simulation results show that, in order to support heavy load traffic $(R \approx 1)$ with little cell losses, a high speed-up factor $(S \approx N)$ is needed. Modular configurations can improve the performance of the proposed scheme and allow for the implementation of large-scale switches with fewer requirements, in terms of speed-up.

References

[1] T. M. Chen and S. S. Liu, *ATM Switching Systems*. Norwood, MA: Artech House, 1995.
[2] B. Danitz, "Campus modular switching product update," in *Networkers 2000*. Session 2810.
[3] [Online]. Available: Marconi web site: http://www.marconi.com/html/solutions
[4] T. K. Woo, "A novel switching architecture for ATM networks," *IEEE Trans. Commun.*, vol. 45, pp. 411–415, Apr. 1997.
[5] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "An efficient adaptive bus arbitration scheme for scalable shared-medium ATM switch," *Comput. Commun.*, vol. 24, pp. 790–797, May 2001.
[6] G. I. Papadimitriou and A. S. Pomportsis, "Learning-automata-based scheduling algorithms for input-queued ATM switches," *Elsevier Neurocomput.*, vol. 31, pp. 191–195, 2000.
[7] S. Motoyama, "Simple high-speed ATM switch with service class priority," *Electron. Lett.*, vol. 36, no. 6, pp. 590–591, 2000.
[8] F. A. Tobagi, T. Kwok, and F. M. Chiussi, "Architecture, performance and implementation of the tandem Banyan fast packet switch," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1173–1193, Aug. 1991.
[9] S. F. Oktug and M. U. Caglayan, "Design and performance evaluation of a Banyan network-based interconnection structure for ATM switches," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 807–816, 1997.
[10] T. X. Brown, "A high-performance two-stage packet switch architecture," *IEEE Trans. Commun.*, vol. 47, pp. 1792–1795, Dec. 1999.
[11] S. H. Byun and D. K. Sung, "The UniMIN switch architecture for large-scale ATM switches," *IEEE/ACM Trans. Networking*, vol. 8, pp. 109–120, Feb. 2000.
[12] R. Kannan and S. Ray, "MSXmin: A modular multicast ATM packet switch with low delay and hardware complexity," *IEEE/ACM Trans. Networking*, vol. 8, pp. 407–418, June 2000.
[13] B. J. Oommen and J. P. R. Christensen, "Epsilon-optimal discretized linear reward-penalty learning automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 6, pp. 451–458, June 1988.
[14] B. J. Oommen and M. Agache, "Continuous and discretized pursuit learning schemes: Various algorithms and their comparison," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 277–287, June 2001.
[15] G. I. Papadimitriou, "Hierarchical discretized pursuit nonlinear learning automata with rapid convergence and high accuracy," *IEEE Trans. Knowl. Data Eng.*, vol. 6, pp. 654–659, Aug. 1994.
[16] K. Najim and A. S. Poznyak, *Learning Automata: Theory and Applications*. New York: Pergamon, 1994.
[17] A. O. Zaghoul and H. G. Perros, "Approximate analysis of a shared-medium ATM switch under bursty arrivals and non uniform destinations," *Perf. Eval.*, vol. 21, pp. 111–129, 1994.
[18] T. K. Woo, "A novel switching architecture for ATM networks," *IEEE Trans. Commun.*, vol. 45, pp. 411–415, Apr. 1997.
[19] A. O. Zaghoul, H. G. Perros, and I. Viniotis, "Performance of bus allocation policies for an ATM switch under bursty arrivals and non uniform destinations," *Proc. 1st IEEE Int. Symp. Global Data Networking*, pp. 37–44, 1993.