Burst-Based Scheduling Algorithms for Non-Blocking ATM Switches with Multiple Input Queues

Ge Nong and Mounir Hamdi, *Member, IEEE*

***Abstract*—This letter quantitatively evaluates two alternative approaches to the scheduling of traffic streams in a high-speed ATM switch with multiple input queues. Specifically, we compare a previously proposed algorithm, called it *parallel iterative matching* (PIM)—which is a cell-based scheduling algorithm—with our newly proposed algorithm—which is a burst-based variation of the PIM scheduling algorithm. Extensive simulation results will demonstrate that burst-based PIM scheduling outperforms cell-based PIM scheduling under a variety of realistic parameters.**

***Index Terms*—Au: Please supply index terms. E-mail keywords@ieee.org for a list.**

I. INTRODUCTION

IT is shown in Karol *et al.* [1] that the head-of-line (HOL) blocking of an input-queued ATM switch limits its throughput to a maximum of 58.6% under uniform random traffic, and much lower than that for bursty traffic. One method for overcoming the HOL blocking problem that is receiving a lot of attention from both academia and industry is *multiple* input-queued ATM switches. In these switches, each input maintains a separate queue of cells destined for each output port. The selection of cells to be transmitted in each time slot from the input ports to the output ports is accomplished using a scheduling algorithm which is a key factor in achieving high-performance using these ATM switches. Several algorithms such as parallel iterative matching (PIM) [2], iSLIP [3], and RPA [4] have been proposed in the literature.

All of these scheduling algorithms are based on cell-by-cell scheduling. This can put a burden on the time complexity and performance of these scheduling algorithms. In particular, these scheduling algorithms should find a maximum sequence of cell transmissions across the switch such that at most one cell is transported from an input port and at most one cell is destined for an output port, and all this should be done within one *single* cell transmission time.

On the other hand, traffic streams in the real world are often characterized as bursty. Most of the application level data units (ADU), such as a video frame, are too large to be encapsulated into a single 53-byte ATM cell and must be segmented into a sequence of cells in order to be transmitted over ATM networks. As a result, consecutive arriving cells in a burst are strongly correlated by having the same destination which addresses the same output of the switch. Our intuitive idea behind this observation is that the cells of a burst should be scheduled as a whole and transmitted continuously in an ATM switch rather than schedule them on a cell-by-cell basis. In other words, the scheduling algorithm should attempt to find a maximum sequence of burst transmissions across the switch rather than a maximum sequence of cell transmissions across the switch.

The main advantage of using a burst-based scheduling algorithm is twofold. First, to an application, the performance metrics of its data units (i.e., bursts) are more relevant performance measures than ones specified by individual cells. Second, by using burst-based scheduling, we can afford not to perform scheduling at each single cell transmission time. This is due to the fact that once the first cell of a burst is scheduled to be transmitted across the switch, all the remaining cells of this burst will be transmitted in the following time slots without interruption and without any scheduling decision.

In this letter, we implement a variation of the PIM scheduling algorithm such that the scheduling decisions are made at the burst level rather than at the cell level. Then, we quantitatively compare its performance with the original PIM scheduling algorithm. The results and findings in this letter can be directly applied to other proposed scheduling algorithms such the iSLIP [3] and the RPA [4].

II. THE SWITCH MODEL AND PIM SCHEDULING

In this section, we present a brief overview of the ATM switch architecture and the PIM scheduling algorithm. Interested readers may refer to [2], [5] for further details.

A. The Switch Model

The ATM switch under consideration is an $N \times N$ non-blocking switch. Each input queue of the switch is a random access buffer. This random access buffer can be viewed as N FIFO queues, each of which is used to store the cells that are destined for one of the N output ports. The architecture of this switch is shown in Fig. 1. The first cell in each queue can be selected for transmission across the switch in each time slot, with the following constraints:

1) At most one cell from any of the N queues in an input port can be transmitted in each time slot.
2) At most one cell could be received by a single output port in each time slot.

Two criteria must be considered when designing the switch: 1) the switch must route as many cells as possible to maximize the throughput and 2) the switch must solve the output

Manuscript received February 1, 1999. The associate editor coordinating the review of this letter and approving it f or publication was Prof. V. S. Frost. This work was supported in part by the Hong Kong Research Grant Council under the Grant RGC/HKUST 100/92E.

The authors are with the Department of Computer Science, The Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: hamdi@cs.ust.hk).

Publisher Item Identifier S 1089-7798(00)05684-2.

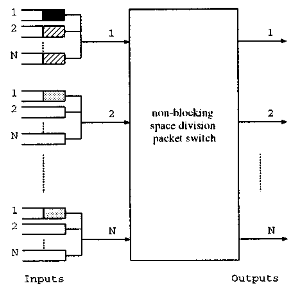


Fig. 1. Architecture of a multiple-input queues ATM switch.

contention problem. As a result, the switch scheduling algorithm that decides, for each time slot, which inputs transmit their queued cells to which outputs is of paramount importance. One such effective algorithm is termed *parallel iterative matching* [2].

B. Parallel Iterative Matching (PIM)

Anderson *et al.* [2] proposed an efficient scheduling algorithm called *parallel iterative matching* (PIM) which uses parallelism, randomness, and iteration to find a maximal matching between the inputs that have queued cells for transmission and the outputs that have queued cells (at the inputs) destined for them. For more details, please refer to [2]. It was shown through analytical modeling and computer simulations, that with as few as 4 iterations, the throughput of a PIM switch exceeds 99% [5]

III. The Burst-Based PIM (BPIM)

One drawback of the PIM algorithm is that it should get executed and produce results (i.e., maximal matching) within the time it takes to transmit one ATM cell. For example, at 1-Gb/s link speed, this time is less than 0.5 μs. In particular, for large switch sizes and using a large number of PIM iterations, it is difficult to produce results within this short time even when using expensive state-of-art microprocessors or special-purpose hardware.

Motivated by the above observation and given the fact that ATM traffic is bursty and correlated in nature, we propose a variation of the PIM algorithm, denoted burst-based PIM (BPIM), that performs scheduling at the burst level rather than at the cell level. Each burst of cells participates during the scheduling only when the burst head cell is at the HOL position. Once a matching between an input and an output is set up to transmit the head cell of a HOL burst, that matching will be *valid* until all cells of this HOL burst have been transmitted.

Burst-Based PIM Algorirhm:

1) Keep all the matchings in the last time slot which are still being *valid* (e.g., the last cell of the burst did not get transmitted yet) at the current time slot unchanged.
2) Each unmatched input sends a request with a given priority to every output for which it has a buffered *head cell of the HOL burst*.
3) If an unmatched output receives any requests, it chooses the *highest priority* one to grant.
4) If an input receives any grants, it chooses the *highest priority* one to accept and notifies that output.
5) Iterate steps 2–4 until a maximal matching is found or until a fixed number of iterations is performed.

Step 1 of the BPIM guarantees that all cells of a burst will be transmitted across the switch without any interruption. In particular, the probability that a large number of HOL bursts complete their transmissions at the same time slot is very low especially when the traffic load is high. As a consequence, steps 2–4 are performed only on a subset of the inputs and outputs which is one major advantage of burst level scheduling over cell level scheduling. Hence, less iterations would be required to find a *maximal* matching among the smaller set of unmatched inputs/outputs. The priority in steps 2–4 can be defined elaborately. For example, we choose the priority according to the length of the burst such as *smallest burst first* (SBF) or *largest burst first* (LBF). For comparison purposes, we also use a *random select* (RND) priority.

TABLE I
Parameters Used in Simulation Experiments

Parameter	Value
switch size	16-by-16
buffer size	4096 cells/queue
algorithm iterations	1, 2, 3
mean burst size	1, 4, 16, 64 cells

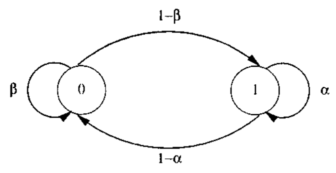


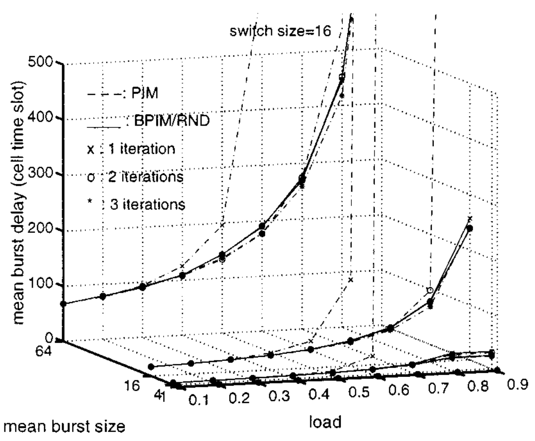
Fig. 2. The 2-MMBP bursty traffic model.

IV. Simulation Results

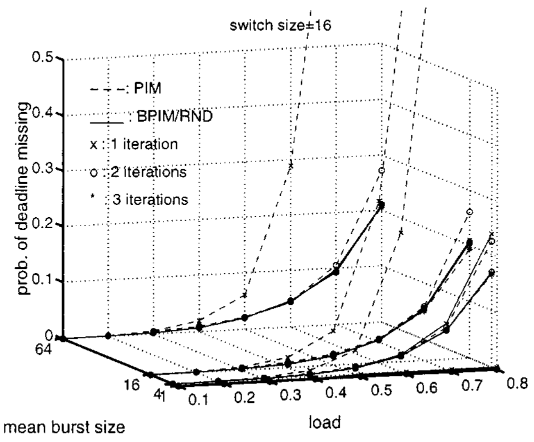
To investigate how much the original PIM algorithm can take advantage from the burst-based scheduling concept, a series of simulations were designed and carried out. Table I gives a summary of the parameters used in the simulation experiments. The bursty traffic that arrives at each input is modeled by a 2-states Markov Modulated Bernoulli Process (2-MMBP) and is illustrated by Fig. 2. The traffic sources alternate between active (1) and idle (0) periods and the mean time interval that the traffic source is being active or is being idle are exponentially distributed with values of $(1-\alpha)^{-1}$ and $(1-\beta)^{-1}$, respectively [6]. When the traffic source is active, a cell is generated. The mean load λ offered by a 2-MMBP with parameters β and α is thus $\lambda = (1-\beta)/(1-\alpha)$. A *burst* consists of the cells generated by the consecutive active states. Once the traffic load λ and the mean burst size τ are given, α and β can be expressed as functions of λ and τ as follows:

$$\alpha = 1 - \frac{1}{\tau} \quad \beta = \frac{1-\lambda(2-\alpha)}{1-\lambda}$$

In this letter we focus our attention on: 1) Finding the *mean burst delay*. As we mentioned earlier such a parameter is more relevant to a network application—it could be the delay of an ADU—than the mean cell delay which is one frequent parameters used in evaluating ATM scheduling algorithms [2], [5]. The



(a)



(b)

Fig. 3. The mean burst delay/probability of deadline missing as functions of the mean traffic loads and mean burst size for BPIM and PIM scheduling.

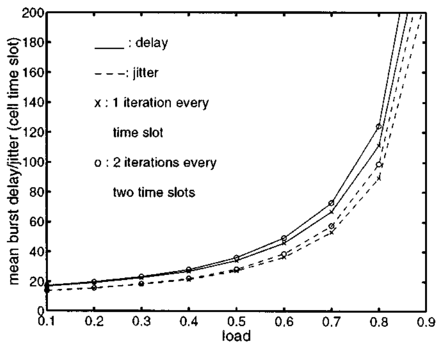


Fig. 4. The mean burst delays for the BPIM scheduling when it is invoked every 2 iterations. Switch size = 16, burst size = 16.

burst delay is defined as the time interval between the appearance of the first cell of the burst at the switch's input link and the arrival of the last cell of the burst at the switch's output link. 2) Finding the probability that a burst will miss its assigned deadline. In our simulations, a burst's assigned deadline is set to be *10 times of the mean burst size*. (We have tested other values that lead to the same conclusion)

Fig. 3(a) and (b) shows the mean burst delay/probability of deadline missing as a function of mean traffic load with mean burst lengths from 1 to 64 cells, for a 16×16 switch. An interesting observation is that the performance of the BPIM scheduling algorithm is insensitive to the iteration number, especially when the mean burst length is large. This is one major advantage of BPIM scheduling over the PIM scheduling algorithm as it does not require expensive high-performance microprocessors to be executed on time. We can see that the various curves for the BPIM scheduling algorithm are almost clustered together under all mean burst lengths, which indicates that one iteration is enough for the BPIM scheduling algorithm to find a *maximal* matching under any traffic load.

The curves of 1-iteration of the BPIM scheduling algorithm are very close to the curves of the 3-iteration PIM scheduling algorithm. This advantage of burst-based scheduling can be explored to design large PIM switches operating at extremely high speeds where the time slot is too short to execute the PIM scheduling with multiple iterations. We also observed that for a 16 by 16 switch under a traffic with a mean burst length of 16 cells, the maximum throughput using 1-iteration PIM scheduling was around 0.6 (exactly 0.63 [5]) and the maximum throughput using 1-iteration BPIM scheduling exceeded 0.9.

As illustrated above, if the ATM switch links are operating at a very high speed, one way of being able to execute the scheduling algorithm on time (i.e., within one cell transmission time) and still get reasonable performance is to use burst-based scheduling with one *single* iteration. An alternative way of solving the same problem is to execute (i.e., invoke) the BPIM scheduling algorithm every few cells transmission time slots rather than every single cell transmission time slot, and then iterate it more than once (e.g., 2 iterations) to get a better matching between the inputs and the outputs. Fig. 4 illustrates this strategy. As can be seen from the figure, if we invoke the BPIM scheduling algorithm every two-cell transmission time and iterate it twice, we still get reasonably good performance.

V. Conclusion

We proposed a variation of the PIM algorithm implemented on the *AN2* switch [2]. Our algorithm performs scheduling at the burst level rather than at the cell level for the following reasons: 1) It is more suitable for providing quality of service for bursty applications; and 2) there is no need for expensive microprocessors to be used for the execution of these algorithms since 1-iteration of this algorithm is sufficient to achieve good performance.

References

[1] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347–1356, Dec. 1987.

[2] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Computer Syst.*, vol. 11, no. 4, pp. 319–352, Nov. 1993.

[3] N. Mckeown, P. Varaiya, and J. Walrand, "Scheduling cells in an input-queued switch," *Electron. Lett.*, vol. 29, no. 25, pp. 2174–2175, 1994.

[4] M. G. A. Marsan, A. Bianco, and E. Leonardi, "RPA: A simple efficient ,and flexible policy for input buffered ATM switches," *IEEE Communications Lett.*, vol. 1, pp. 83–86, May 1997.

[5] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of nonblocking ATM switches with multiple input queues," *IEEE/ACM Trans.Networking*, vol. 7, pp. 60–74, Feb. 1999.

[6] A. Adas, "Traffic models in broadband networks," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 82–89, July 1997.