Analysis of Nonblocking ATM Switches with Multiple Input Queues

Ge Nong, *Student Member, IEEE*, Jogesh K. Muppala, *Member, IEEE*, and Mounir Hamdi, *Member, IEEE*

***Abstract*— An analytical model for the performance analysis of a multiple input queued asynchronous transfer mode (ATM) switch is presented in this paper. The interconnection network of the ATM switch is internally nonblocking and each input port maintains a separate queue of cells for each output port. The switch uses parallel iterative matching (PIM) [7] to find the maximal matching between the input and output ports of the switch. A closed-form solution for the maximum throughput of the switch under saturated conditions is derived. It is found that the maximum throughput of the switch exceeds 99% with just four iterations of the PIM algorithm. Using the *tagged input queue* approach, an analytical model for evaluating the switch performance under an independent identically distributed Bernoulli traffic with the cell destinations uniformly distributed over all output ports is developed. The switch throughput, mean cell delay, and cell loss probability are computed from the analytical model. The accuracy of the analytical model is verified using simulation.**

***Index Terms*— Analytical modeling, ATM switch, computer simulation, performance evaluation.**

I. Introduction

THE ASYNCHRONOUS transfer mode (ATM) has been recommended by CCITT as an integrated transport technique for future broad-band ISDN. As a result, there is a growing interest in identifying suitable switching architectures for ATM. Virtually all proposals of switching fabric architectures for ATM networks are based on self-routing structures in which each packet autonomously find its path through the interconnection network to the desired output port [1]–[3]. This feature enables the design of switching fabrics characterized by a very high throughput, on the order of gigabits per second. These design proposals were based on various types of queueing strategies: input queueing, dedicated internal queueing, shared internal queueing, or output queueing [2], [3].

Each of these queueing strategies is characterized by certain advantages and drawbacks. The simplest approach, however, is input queueing. In this architecture, each input port maintains a first-in-first-out (FIFO) queue of cells, and only the first cell in the queue is eligible for transmission during a given time slot. The drawback of FIFO queueing is that, when the cell at the head of the queue is blocked, all cells behind it in the queue are prevented from being transmitted, even when the output port they need is idle. This is called *head-of-line* (HOL) blocking. It was shown through mathematical analysis and computer simulation that HOL blocking limits the throughput of each input port to a maximum of 58.6% under uniform random traffic, and much lower than that for bursty traffic [4], [5].

Various approaches have been proposed to overcome the problems associated with FIFO input queueing: adopting a switch expansion, a windowing technique, or a channel grouping technique [2]. In the former case, the internal switch bandwidth is expanded, using replication for example, so that it can transmit more than one cell to an output port in a single time slot. Windowing is a technique conceived to relieve the HOL blocking by also allowing non-HOL cells to contend for the switch output ports. In particular, adopting a window of depth w means that if a cell in position i, $1 \leq i \leq w$, ($i = 1$ identifies the HOL position) is blocked owing to a conflict for the switch output port, a chance is given to the cell in position $i + 1$ until a search of depth w has been completed. Channel grouping is another technique that can be used with input queueing to improve the switch throughput. In this technique, the switch output ports are subdivided into groups of size R each and cells address groups, not single links of a group. Such arrangement is typical of a transit environment in which the switch interfaces a limited number of downstream switches with R links to each of them. Implementing channel groups in an ATM switch is a nontrivial task given the high rates of the links, since in each slot not only we have to guarantee the absence of external conflicts, but also to allocate at the transmission time each output port in a group to a specific input port addressing that group with its HOL cell.

Of particular interest to us in this paper is a recent technique termed *parallel iterative matching* (PIM) and variations of PIM, that are used to reduce the HOL blocking [7], [9]–[12]. In this technique. each input port maintains a separate queue for each output port. During a single time slot, a maximum of one cell per input port can be transferred, and a maximum of one cell per output port can be received. The switch operation is based on a parallel iterative matching algorithm to find the maximal matching between the inputs and outputs of the switch. It is shown that the number of iterations needed to find a maximal matching is N in the worst case, and $O(\log N)$ in the average case. However, it was shown through computer simulation, that with as few as four iterations, the throughput of a 16×16 switch exceeds 99% [7]. As a result, this switch architecture has received a lot of attention from the research

Manuscript received July 21, 1997; revised March 2, 1998; approved by IEEE/ACM Transactions on Networking Editor H. J. Chao. This work was supported in part by the Hong Kong Research Grant Council under Grant RGC/HKUST 6026/97E.

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

Publisher Item Identifier S 1063-6692(99)01817-8.

community, and many commercial and experimental ATM switches based on this queueing technique have already been built, such as the DEC Systems AN2 switch and the Tiny Tera switch [7], [8].

So far, the performance evaluation of PIM switches appearing in the literature have been based on simulation. The main reason for that is the fact that mathematically analyzing PIM ATM switches is difficult. The inventors of PIM switches acknowledge this fact by saying, "In general, it is difficult to mathematically analyze the performance of a PIM switch, even for the simplest traffic models. The problem lies in the evolution and interdependence of the state of each arbiter and their dependence on arriving traffic" [7], [9]. To the best of our knowledge, the research work in this paper is the first attempt at analytical modeling of the performance of the PIM switches. Performance parameters such as throughput, mean cell delay, and cell loss probability are important to evaluate the performance of an ATM switch. For example, a cell loss probability requirement of 10^{-9} is not uncommon [14]. As a result, estimating the rare cell loss probability by simulation is inefficient and sometimes impossible [17]. Consequently, analytical models are of great importance in solving these problems.

The contribution of this paper is twofold. First, we analyze the performance of the PIM ATM switch under saturated conditions [23], i.e., assuming that none of the N^2 input queues is empty, so as to give the maximum throughput of the switch. With this assumption, a recursive closed-form expression for computing the maximum throughput of the switch is derived. Using this expression, we show that for one iteration, the maximum throughput of the switch converges to 0.632 as the size of the switch increases, which agrees with the results given in [7]. We also show that the maximum throughput for a switch of any size exceeds 99% with just four iterations of the PIM scheduling algorithm. It is worth noting that a 100% throughput can be achieved with the same architecture using a maximum weight bipartite matching algorithm [25]. Another algorithm in [9] can also achieve 100% throughput with this same architecture.

We then develop an analytical model of the PIM switch with finite input buffers using the *tagged queue* approach [6], [15], [16]. Assuming an independent identically distributed Bernoulli input traffic with the cell destinations distributed uniformly over all the outputs, we compute interesting performance measures for the PIM switch including throughput, mean cell delay, and cell loss probability as a function of the offered load and switch size. The accuracy of our analytical model has been verified using simulation. The results from the analytical model match closely with the simulation results. This model will be extended in the future under more realistic traffic assumptions such as bursty and correlated traffic. This will render our mathematical analysis more complex. We consider the results of this paper as the foundation for solving the more difficult problem involving bursty and correlated traffic [24].

The remainder of this paper is organized as follows. Section II introduces the switch model and PIM scheduling algorithm. The next two sections present the analytical model of PIM switches both under saturated conditions, and i.i.d. Bernoulli traffic. In particular, Section III presents recursive equations for the maximum throughput of the switch. Section IV develops the analytical model based on the tagged queuing approach. Equations for computing interesting performance measures including throughput, mean cell delay, and mean cell loss probability are derived in this section. Numerical results obtained from the analytical model are presented for switches of different sizes in Section V, and compared with the results from simulation. Finally conclusions are presented in Section VI.

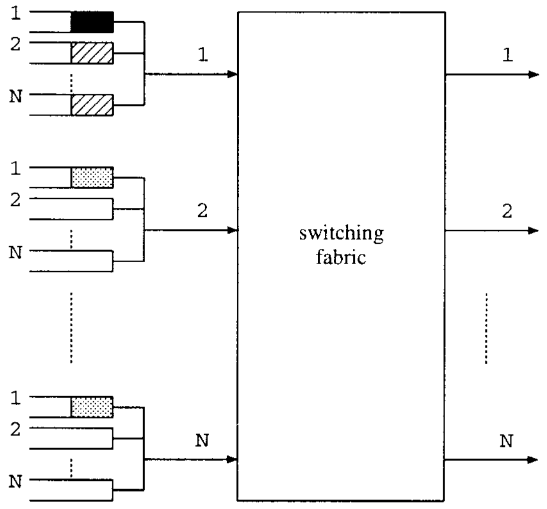


Fig. 1. Architecture of the multiple input queues ATM switch ($N \times N$).

II. The Switch Model and PIM Scheduling

In this section, we present an overview of the PIM ATM switch architecture and define the switch scheduling algorithm (i.e., PIM). Interested readers may refer to [7], [9] for further details.

A. The Switch Model

The ATM switch under consideration is an $N \times N$ nonblocking switch, i.e., the N inputs are connected to the N outputs via a nonblocking interconnection network (e.g., crossbar switch). A cell may be sent from any input to any output, provided that no more than one cell is sent from the same input and no more than one cell is received by the same output. Each input queue of the switch is a random access buffer. This random access buffer can be used to construct 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) only one cell from any of the N queues in an input port can be transmitted in each time slot;
2) only one cell can be transmitted from the N input ports to an output port of the switch at any given time slot. In other words, at most one cell could be received by a single output port.

As mentioned above, the reason for using a random-access buffer instead of a conventional FIFO queue is to avoid the HOL blocking. Two criteria must be considered for designing the switch:

1) the switch must route as many cells among the ones that arrived at its inputs as possible to the appropriate outputs to maximize the throughput;
2) the switch must solve the output contention problem (i.e., more than one cell can be destined for the same output port).

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 PIM [7], [9].

B. PIM

For switches used in high-performance ATM networks, the switch scheduling algorithm must be able to provide high throughput, low latency, and graceful degradation under heavy traffic loads. Anderson *et al.* [7] considered the architecture of a nonblocking switch with random access buffers (as shown in Fig. 1), and cast the switch scheduling problem as a bipartite matching problem of finding conflict-free pairing of inputs to outputs [13]. The high throughput and low latency of an ATM switch dictates that the scheduling algorithm must be able to find a matching of as many conflict-free pairings as possible, using as little time as possible. Note that a *maximum matching* is a matching with the maximum number of paired inputs and outputs, while *maximal matching* is one in which no unmatched input has a queued cell destined for an unmatched output (i.e., no parings can be trivially added). Unfortunately, all of the conventional bipartite maximum matching algorithms have high time complexity with regard to the time constraint imposed by 53-byte ATM cells and gigabit per second links. As a solution, Anderson *et al.* [7] proposed an algorithm, called *parallel iterative matching*, to find a maximal matching.

The algorithm proposed by Anderson *et al.* 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. Maximal matching is used to determine which inputs transmit cells over the nonblocking switch to which outputs in the current time slot. Specifically, their matching algorithm iterates the following three steps until a maximal matching is found or until a fixed number of iterations is performed.

Request: Each unmatched input sends a request to *every* output for which it has a queued cell.

Grant: If an unmatched output receives any requests, it grants to one by *randomly* selecting a request uniformly over all requests.

Accept: If an input receives grants, it accepts one by selecting an output among those that granted to this input.

By considering only unmatched inputs and outputs, each iteration only considers connections not made by earlier iterations.

Note that in step two above, the independent output schedulers *randomly* select a request among contending requests. This has three effects. First, Anderson *et al.* [7] show that each iteration will match or eliminate on average at 3/4 of the remaining possible connections and thus the algorithm will converge to a maximal match in $O(\log N)$ iterations. Second, it ensures that all requests will eventually be granted. Consequently, no input queue is starved. Third, it means that no memory or state is used to keep track of how recently a connection was made in the past.

To facilitate the mathematical analysis in the rest of the paper, we modify the above PIM algorithm. However, the modified PIM algorithm is *logically equivalent* to the original PIM algorithm. As a result, the performance analysis of the modified PIM algorithm is exactly applicable to the original PIM algorithm. The modified PIM algorithm iterates the following two steps until a maximal matching is found, or until a fixed number of iterations are performed:

1) each unmatched input chooses an output *uniformly* over all unmatched outputs for which it has queued cells and sends a request to it;
2) if an unmatched output receives any requests, it chooses one *uniformly* over all requests to grant and notifies each requesting input.

The logical equivalence between the modified PIM algorithm and the original PIM algorithm can be easily derived from the above description of the modified PIM algorithm. If we regroup the N^2 queues according to their destination outputs instead of being grouped by the inputs, the original PIM algorithm is now mapped into its modified counterpart. The modified algorithm can thus be viewed as a mirror image of the original PIM algorithm, with the roles of the input and output ports switched. For the modified algorithm, the input selects among all unmatched outputs for which it has cells queued. This is similar to the *grant* step of the original algorithm. Similarly, step two of our algorithm is equivalent to the *accept* step of the original algorithm.

In the rest of the paper, we will not distinguish between the original PIM and the modified PIM since they are essentially the same.

III. Maximum Throughput of Multiple Iterations PIM

First, the ATM switch with one iteration PIM is analyzed. Then, a recursive formula for the throughput of the switch with multiple iterations PIM is derived. Under saturated conditions, all the queues at each input will have at least one cell. An output selects one uniformly among the input requests.

A. Throughput of One Iteration

The throughput of an ATM switch with one iteration PIM scheduling $\rho(1)$ is equal to the probability that an output O_j gets matched after the first iteration. Since each output selects uniformly from all of the input requests, the probability of an input request being accepted by an output $p = 1/N$. Then

$$\rho(1) = \sum_{k=0}^{N-1} \binom{N-1}{k} \frac{p^k(1-p)^{N-1-k}}{(k+1)} = 1 - \left(1 - \frac{1}{N}\right)^N$$

and

$$\lim_{N\to\infty} \rho(1) = 1 - e^{-1} = 0.632. \quad (1)$$

Let $\Pr\{m(1)\}$ be the probability that $m(1)$ inputs (outputs) get matched and output O_j remains unmatched after the first iteration, then

$$\Pr\{m(1)\} = \binom{N-1}{m(1)} m(1)! \mathcal{S}_N^{(m(1))} \Big/ N^N \quad (2)$$

where $\mathcal{S}_n^{(m)}$ is the *stirling number* of the second kind which gives the number of ways of partitioning a set of n elements into m nonempty subsets. It can be computed by the closed-form expression [22]

$$\mathcal{S}_n^{(m)} = \frac{1}{m!} \sum_{k=0}^{m} (-1)^{m-k} \binom{m}{k} k^n.$$

In (2), $m(1)!\mathcal{S}_N^{(m(1))}$ accounts for the number of possible cases in which N inputs contend for the given $m(1)$ outputs with the condition that each output is requested by at least one input.

Let $\Pr\{n(1)\}$ denote the probability that $n(1)$ inputs (outputs) remain unmatched after the first iteration with the condition that output O_j remains unmatched, then $\Pr\{n(1)\} = \Pr\{m(1) = N - n(1)\}$.

B. Throughput of Multiple Iterations

The throughput of two iterations PIM scheduling is equal to the sum of $\rho(1)$ and the probability that output O_j gets matched in the second iteration, that is

$$\begin{aligned}\rho(2) &= \rho(1) + \Pr\{\text{output } O_j \text{ gets matched in the second iteration}\} \\ &= \rho(1) + \sum_{n(1)=1}^{N-1} \left(1 - \left(1 - \frac{1}{n(1)}\right)^{n(1)}\right) \Pr\{n(1)\}.\end{aligned}$$

Similarly, we can derive the throughput of three iterations PIM scheduling as follows:

$$\begin{aligned}\rho(3) &= \rho(2) + \Pr\{\text{output } O_j \text{ get matched in the third iteration}\} \\ &= \rho(2) + \sum_{n(2)=1}^{N-2} \left(1 - \left(1 - \frac{1}{n(2)}\right)^{n(2)}\right) \Pr\{n(2)\}.\end{aligned}$$

To derive the throughput of i iterations PIM scheduling $\rho(i)$, we first derive the expression for the probability that there are $n(i)$ inputs (outputs) remaining unmatched after the ith iteration scheduling. This is given in a recursive form by

$$\begin{aligned}&\Pr\{n(i)\} \\ &= \sum_{n(i-1)=n(i)+1}^{N-(i-1)} \Pr\{m(i) = n(i-1) - n(i)\} \Pr\{n(i-1)\}\end{aligned} \quad (3)$$

where

$$\Pr\{m(i)\} = \binom{n(i-1)-1}{m(i)} m(i)! \mathcal{S}_{n(i-1)}^{(m(i))} \Big/ (n(i-1))^{n(i-1)}.$$

Using (3), the throughput of i iteration PIM scheduling can be derived as follows:

$$\rho(i) = \rho(i-1) + \sum_{n(i-1)=1}^{N-(i-1)} \left(1 - \left(1 - \frac{1}{n(i-1)}\right)^{n(i-1)}\right) \cdot \Pr\{n(i-1)\}.$$

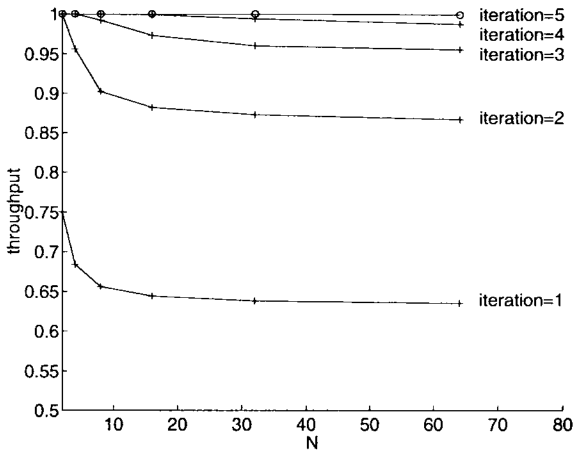


Fig. 2. Maximum throughput as a function of switch size and number of iterations.

Fig. 2 shows the results for the maximum throughput of a PIM switch as a function of switch size and number of iterations. As shown in this figure, the maximum throughput of an ATM switch with one iteration PIM scheduling converges to 0.63 [which corresponds to (1)] when the switch size grows. Furthermore, the throughput increases significantly after each iteration of PIM scheduling. Four iterations are sufficient for achieving maximum throughput of about 99% for a switch of any size.

IV. Queueing Model and Analysis of Multiple Iterations PIM

In this section, we model the ATM switch with PIM scheduling using queueing theory and analyze the underlying Markov chain. Our method uses the concept of *tagged queues* in modeling the PIM switch leading to a smaller state space. The concept of *tagged input queue* has been successfully used to evaluate the FIFO input-queued switch model [6], [15], [16]. These switches involve a single stage of contention resolution. On the other hand, for the switch with PIM scheduling, the contention resolution process consists of two stages. As

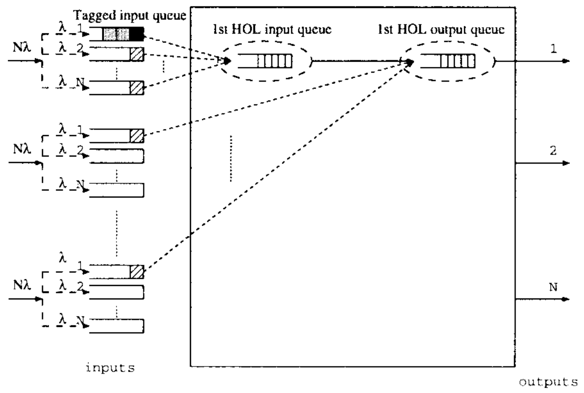


Fig. 3. An example of the queueing model for the PIM switch.

observed from the algorithm descriptions of PIM, a HOL cell in an input queue will contend for transmission not only with the HOL cells of the same input, but also the HOL cells destined for the same output. As a result, the corresponding model is more complicated than for the FIFO input-queued switch. We make the following assumptions in developing the PIM switch model:

1) the switch operates synchronously;
2) every input queue has the same buffer size, namely b_i;
3) cells arrive at the inputs according to an *i.i.d.* Bernoulli process with parameter $N\lambda$ ($0 < N\lambda \leq 1$). The destinations of the cells are uniformly distributed over all the outputs. Only one cell can arrive at each input in a time slot. For an $N \times N$ switch, if an input's load is $N\lambda$, then every queue at this input has an offered load of λ;
4) new cells arrive only at the beginning of the time slots, and cells depart only at the end of the time slots.

Under the above assumptions, all the input queues will exhibit the same behavior when the system attains steady-state. A queue at input i with output j as the destination is denoted by $Q(i, j)$. Fig. 3 shows an example of the queueing model for the PIM switch. In this example, the occupancy of $Q(1, 1)$ is taken as the *tagged input queue*, the number of HOL cells at input 1 is represented by the *1st HOL input queue*, and the number of HOL cells addressed for output 1 is denoted by the *1st HOL output queue*. Both the *HOL input queue* and the *HOL output queue* are virtual queues which do not exist in a real PIM switch, but are useful for our mathematical analysis.

A. Markov Model

Analyzing the queueing model of the PIM switch requires the construction of the underlying Markov chain Z. The states of the Markov chain Z are sampled at the end of the time slots and can be expressed as a triplet (L, W_i, W_o), where L, W_i, and W_o refer to the lengths of the *tagged input queue, virtual HOL input queue,* and *virtual HOL output queue,* respectively. The state-space of this three-dimensional Markov chain is

$$\{(0,0,0),(l,w_i,w_o) | 1 \leq l \leq b_i, 1 \leq w_i \leq N, 1 \leq w_o \leq N\}$$

and are ordered in a lexicographic order, i.e.,

$$(0,0,0),(1,1,1),(1,1,2),\cdots,(1,2,1),(1,2,2),\cdots,(b_i, N, N).$$

The set of states

$$\{(l, 1, 1),(l, 1, 2), \cdots, (l, 2, 1), (l, 2, 2), \cdots, (l, N, N)\}$$

will be labeled as states in level l of the Markov chain. (Note that when the tagged input queue is empty, we are not interested in the lengths of the HOL input queue and the HOL output queue. Thus all states with $l = 0$ are collapsed into a single state $(0, 0, 0)$, i.e., we are not explicitly keeping track of the state of the HOL input queue and the HOL output queue when the tagged input queue is empty.) This Markov chain has two important attributes, namely: 1) in moving from the state (l, w_i, w_o) to a state $(l + i, w_i', w_o')$, for $i \geq 1$, the chain must visit all intermediate levels at least once and 2) this Markov chain is a *quasi birth and death (QBD)* process, with block-partitioned form of transition probability matrix. These attributes enable us to solve the Markov chain in an recursive way as we will show in the Appendix. The transition probability matrix T of the Markov chain is defined as follows:

$$T = \begin{bmatrix} C_1 & C_2 & 0 & & & & & & \\ C_0 & A_1 & A_2 & 0 & & & & & \\ 0 & A_0 & A_1 & A_2 & 0 & & & & \\ 0 & 0 & A_0 & A_1 & A_2 & 0 & & & \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & & & \\ 0 & 0 & 0 & \cdots & 0 & \cdots\ 0 & A_0 & A_1 & A_2 \\ 0 & 0 & 0 & \cdots & 0 & \cdots\ 0 & 0 & D_0 & D_1 \end{bmatrix}$$

where $C_1 + C_2 e = 1$ and $C_0 + (A_1 + A_2)e = (A_0 + A_1 + A_2)e = (D_0 + D_1)e = e$, e is a column vector of ones of length N^2.

Let

$$P_{\text{blo}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)} \left(P'_{\text{blo}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)} \right)$$

denote the probability that the HOL cell of the tagged input queue is blocked, and

$$P_{\text{suc}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)} \left(P'_{\text{suc}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)} \right)$$

denote the probability that the HOL cell of the tagged input queue is transmitted given that the remaining HOL cells at the end of the last time slot is (w_i, w_o), the remaining HOL cells at the end of the current time slot is (w_i', w_o') and there is a new cell arrival (is no new cell arrival) at the *tagged input queue* at the beginning of the current time slot. For the first case (there is a new arrival cell at the *tagged input queue* at the beginning of the current time slot), we define three matrices B_0, B, and S as (4)–(6), shown at the bottom of the next page.

In case there is no new cell arrival at the *tagged input queue* at the beginning of current time slot, we define three matrices B', B_0' and S' similar to as B, B_0, and S by replacing $P_{\text{blo}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)}$ in $B$with $P'_{\text{blo}, W_t(w_i', w_o') | W_{t-1}(w_i, w_o)}$, $P_{\text{blo}, W_t(w_i', w_o') | W_{t-1}(0, 0)}$ in B_0

with $P'_{blo,W_t(w_i',w_o')|W_{t-1}(0,0)}$, and $P_{\text{suc},W_t(w_i',w_o')|W_{t-1}(w_i,w_o)}$ in S with $P'_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$, respectively.

By using the above equations, the element matrices in the transition probability matrix T can be computed as

$$\begin{array}{lll} C_0 = S'e & C_1 = 1 - B_0 e & C_2 = B_0 \\ A_0 = S' & A_1 = S + B' & A_2 = B \\ D_0 = S + S' & D_1 = B + B'. & \end{array}$$

The remaining subsections will cover the computation of the success and blocking probabilities we defined above, i.e., $P_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$, $P'_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$, $P_{\text{blo},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$, and $P'_{\text{blo},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$, respectively. Provided that these probabilities are computed, the transition probability matrix T can be constructed. Once the transition probability matrix is known, it is a routine matter to derive the steady-state equations by utilizing the properties of Markov chains, and solving the equations to obtain the steady-state probability vector. The steady-state probability vector of the Markov chain Z is given by $\Pi = [\pi_0, \pi_1, \pi_2, \cdots, \pi_l, \cdots, \pi_{b_i}]$, where every element $\pi_l = [\pi_{(l,1,1)}, \pi_{(l,1,2)}, \cdots, \pi_{(l,N,N)}]$, $l > 0$ is a row vector of size N^2, and π_0 is a scalar. Detailed procedures to obtain the steady-state probabilities are presented in the Appendix.

B. Computing the Blocking and Success Probabilities

We now derive the equations for computing the blocking probability $P_{\text{blo},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$ and the success probability $P_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$. The transition of the state of the virtual HOL input/output queues from the state $(w_i,\, w_o)$ to state $(w_i',\, w_o')$ is a two-step process.

1) First, we account for the newly arriving HOL cells to the virtual HOL input/output queues.
2) Then, we consider the transition from the intermediate state to the final state after applying the PIM algorithm.

This process is illustrated in Fig. 4.

1) Arriving Cells at Virtual HOL Queues: Let $K_t(k_t, k_i, k_o)$ denote the number of newly arriving HOL cells at the *tagged input queue/virtual HOL input/virtual HOL output queues* ($k_t/k_i/k_o$ new arrivals to the *tagged input queue/virtual HOL input/virtual HOL output queue*), at the

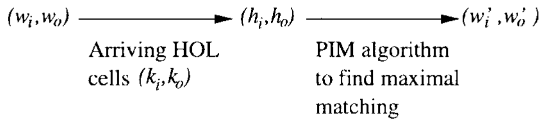

(w_i,w_o)
(h_i,h_o)
(w_i',w_o')
Arriving HOL cells (k_i,k_o)
PIM algorithm to find maximal matching

Fig. 4. Transition of the virtual HOL queues.

beginning of current time slot t. Note that since only one cell can arrive to an input in any time slot, then $0 \le k_i \le 1$. $W_{t-1}(w_i,\, w_o)$ denotes the numbers of remaining HOL cells at the *virtual HOL input/output queue* (w_i/w_o is length of *virtual HOL input/output queue*), at the end of the previous time slot $t-1$. Let $H_t(h_i,\, h_o) = K_t(k_i,\, k_o) + W_{t-1}(w_i,\, w_o)$. Define, $a_{K(k_t,\,k_i,\,k_o)|W(w_i,\,w_o)} = \Pr\{K_t(k_t,\, k_i,\, k_o)|W_{t-1}(w_i,\, w_o)\}$. Let p_0 be the probability that an input queue is empty in a time slot, where $p_0 = (1-\lambda)\pi_0$, and let $p_1 = 1 - p_0$. Two different cases need to be explicitly considered: a) when the tagged input queue $Q(i,\, j)$ is nonempty and b) when $Q(i,\, j)$ is empty.

Case (a): Since the tagged input queue $Q(i,\, j)$ is nonempty, both $w_i > 0$ and $w_o > 0$. If the current state is $(l,\, w_i,\, w_o)$, $(N - w_i)$ input queues of input i and $(N - w_o)$ jth input queues of other inputs will be empty. In the following, let $k_t = 0$ denote that no cell arrives to the tagged input queue in the current time slot, and let $k_t = 1$ denote that a cell arrives to the tagged input queue. Hence,

$$a_{K(k_t,\,k_i,\,k_o)|W(w_i,\,w_o)} = \begin{cases} \binom{N-w_o}{k_o}\lambda^{k_o+1}(1-\lambda)^{N-w_o-k_o} \\ k_t = 1,\, k_i = 0,\, 0 \le k_o \le N - w_o,\, 1 \le w_i,\, w_o \le N. \\ (1-(N-w_i+1)\lambda)\binom{N-w_o}{k_o}\lambda^{k_o}(1-\lambda)^{N-w_o-k_o} \\ k_t = 0,\, k_i = 0,\, 0 \le k_o \le N - w_o,\, 1 \le w_i,\, w_o \le N. \\ (N-w_i)\binom{N-w_o}{k_o}\lambda^{k_o+1}(1-\lambda)^{N-w_o-k_o} \\ k_t = 0,\, k_i = 1,\, 0 \le k_o \le N - w_o,\, 1 \le w_i,\, w_o \le N. \end{cases}$$

Case (b): As stated earlier, when the tagged input queue is empty, we do not explicitly keep track of the state of the virtual HOL input queues and HOL output queues. In this case, all such states are collapsed into the single state $(0,\, 0,\, 0)$ with

$$B_0 = [P_{\text{blo},\,W_t(1,1)|W_{t-1}(0,0)},\quad P_{\text{blo},\,W_t(1,2)|W_{t-1}(0,0)},\quad \cdots,\; P_{\text{blo},\,W_t(N,N)|Wt-1(0,0)}] \tag{4}$$

$$B = \begin{bmatrix} P_{\text{blo},\,W_t(1,1)|W_{t-1}(1,1)} & P_{\text{blo},\,W_t(1,2)|W_{t-1}(1,1)} & \cdots & P_{\text{blo},\,W_t(N,N)|W_{t-1}(1,1)} \\ P_{\text{blo},\,W_t(1,1)|W_{t-1}(1,2)} & P_{\text{blo},\,W_t(1,2)|W_{t-1}(1,2)} & \cdots & P_{\text{blo},\,W_t(N,N)|W_{t-1}(1,2)} \\ P_{\text{blo},\,W_t(1,1)|W_{t-1}(1,3)} & P_{\text{blo},\,W_t(1,2)|W_{t-1}(1,3)} & \cdots & P_{\text{blo},\,W_t(N,N)|W_{t-1}(1,3)} \\ \vdots & \vdots & \cdots & \vdots \\ P_{\text{blo},\,W_t(1,1)|W_{t-1}(N,N)} & P_{\text{blo},\,W_t(1,2)|W_{t-1}(N,N)} & \cdots & P_{\text{blo},\,W_t(N,N)|W_{t-1}(N,N)} \end{bmatrix} \tag{5}$$

$$S = \begin{bmatrix} P_{\text{suc},\,W_t(1,1)|W_{t-1}(1,1)} & P_{\text{suc},\,W_t(1,2)|W_{t-1}(1,1)} & \cdots & P_{\text{suc},\,W_t(N,N)|W_{t-1}(1,1)} \\ P_{\text{suc},\,W_t(1,1)|W_{t-1}(1,2)} & P_{\text{suc},\,W_t(1,2)|W_{t-1}(1,2)} & \cdots & P_{\text{suc},\,W_t(N,N)|W_{t-1}(1,2)} \\ P_{\text{suc},\,W_t(1,1)|W_{t-1}(1,3)} & P_{\text{suc},\,W_t(1,2)|W_{t-1}(1,3)} & \cdots & P_{\text{suc},\,W_t(N,N)|W_{t-1}(1,3)} \\ \vdots & \vdots & \cdots & \vdots \\ P_{\text{suc},\,W_t(1,1)|W_{t-1}(N,N)} & P_{\text{suc},\,W_t(1,2)|W_{t-1}(N,N)} & \cdots & P_{\text{suc},\,W_t(N,N)|W_{t-1}(N,N)} \end{bmatrix} \tag{6}$$

$w_i = w_o = 0$. When a cell arrives into the empty tagged input queue $Q(i, j)$, we have to explicitly restore the states of the virtual HOL queues. This is accomplished as follows. A cell that arrives at $Q(i, j)$ when $Q(i, j)$ is empty will observe that another input queue at input i is nonempty with probability $(1-\pi_0)$. Similarly the cell will observe that a jth input queue at other input ports is nonempty, with probability p_1

$$a_{K(1, k_i, k_o)|W(0,0)} = \lambda \binom{N-1}{k_i-1}\binom{N-1}{k_o-1}(1-\pi_0)^{k_i-1} \cdot \pi_0^{N-k_i} p_1^{k_o-1} p_0^{N-k_o}, \qquad 1 \le k_i;\ k_o \le N;\ w_i = w_o = 0.$$

In this case, we reinterpret the k_i and k_o as if they represent the number of arrivals into the virtual HOL input and output queues in order to restore the states of the corresponding HOL queues. Then $H_t(h_i, h_o)K_t(k_i, k_o)$.

2) Transition to $W_t(w_i', w_o')$: Having determined the number of cell arrivals to the virtual HOL queues, we now consider the transition from the intermediate state to the final state after applying the PIM algorithm.

Given that the *tagged input queue* $Q(i, j)$ is nonempty, the inputs excluding input i are divided into two subsets E and F according to whether the jth queue of the inputs is empty or not. The cardinality of these sets are $(N - w_o)$ and $(w_o - 1)$, respectively. The state of sets E and set F will affect the transitions of *virtual HOL input queue* and *virtual HOL output queue*. For the HOL cell of the tagged input queue, its contention process can be split into two stages. In the first stage, the tagged input queue contends with other nonempty queues at the same input. If it succeeds in the first stage contention, it joins the second stage contention with all successful jth queues from other inputs. Let $Q(i, k)(k \ne j)$ be the successful queue at input i if $Q(i, j)$ is blocked in the first contention stage.

Given $H_t(h_i, h_o)$ and $W_t(w_i', w_o')$, the possible values of W_t in terms of H_t are

$$(w_i', w_o') = \begin{cases} (h_i, h_o) \\ (h_i, h_o - 1), & \text{for } h_o > 0 \\ (h_i - 1, h_o), & \text{for } h_i > 0 \\ (h_i - 1, h_o - 1), & \text{for } h_i > 0 \text{ and } h_o > 0. \end{cases}$$

The last case $(w_i', w_o') = (h_i - 1, h_o - 1)$ can result from two situations: 1) when the HOL cell from the tagged input queue $Q(i, j)$ gets successfully matched or 2) another queue $Q(i, k)$ gets successfully matched, and the output j gets successfully matched to another input. We define the following probabilities associated with the above transitions:

$P_{blo\_00\|H(h_i,h_o)}$	Pr{ the HOL cell at the tagged input queue gets blocked, and $W_t(w_i', w_o') = H_t(h_i, h_o)$ given $H_t(h_i, h_o)$};
$P_{blo\_01\|H(h_i,h_o)}$	Pr{ the HOL cell at the tagged input queue gets blocked, and $W_t(w_i', w_o') = H_t(h_i, h_o - 1)$ given $H_t(h_i, h_o)$};
$P_{blo\_10\|H(h_i,h_o)}$	Pr{ the HOL cell at the tagged input queue gets blocked, and $W_t(w_i', w_o') = H_t(h_i - 1, h_o)$ given $H_t(h_i, h_o)$};
$P_{blo\_11\|H(h_i,h_o)}$	Pr{ the HOL cell at the tagged input queue gets blocked, and $W_t(w_i', w_o') = H_t(h_i - 1, h_o - 1)$ given $H_t(h_i, h_o)$};
$P_{suc\|H(h_i,h_o)}$	Pr{ the HOL cell at the tagged input queue gets transmitted, and $W_t(w_i', w_o') = H_t(h_i - 1, h_o - 1)$ given $H_t(h_i, h_o)$}.

TABLE I
TRANSITIONS FROM (h_i, h_o) TO (w_i', w_o')

$Q(i, j)$	(w_i', w_o')	Probability
blocked:	(h_i, h_o)	$P_{\text{blo\_00}\|H(h_i,h_o)}$
	$(h_i, h_o - 1)$	$P_{\text{blo\_01}\|H(h_i,h_o)}$
	$(h_i - 1, h_o)$	$P_{\text{blo\_10}\|H(h_i,h_o)}$
	$(h_i - 1, h_o - 1)$	$P_{\text{blo\_11}\|H(h_i,h_o)}$
successful:	$(h_i - 1, h_o - 1)$	$P_{\text{suc}\|H(h_i,h_o)}$

Table I summarizes the transitions and their corresponding probabilities.

With respect to whether there is a new arrival at the *tagged input queue* or not, the two blocking probabilities of $P_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$ and $P'_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$ are computed as follows.

Given $r_i = w_i' - w_i$ and $r_o = w_o' - w_o$, the blocking probability $P_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$ is computed as shown in (7), at the bottom of the next page, in which

$$li_0(w) = \begin{cases} 0, & \text{for } w = 0 \\ \sum_{u=1}^{w} \binom{w}{u} u P_{li1}^u (1 - P_{li1})^{w-u}, & \text{for } w > 0 \end{cases} \tag{8}$$

$$lj_0(w) = \begin{cases} 0, & \text{for } w = 0 \\ \sum_{u=1}^{w} \binom{w}{u} u P_{lj1}^u (1 - P_{lj1})^{w-u}, & \text{for } w > 0 \end{cases} \tag{9}$$

$$lj_0'(w) = \begin{cases} 0, & \text{for } w = 0 \\ \sum_{u=1}^{w} \binom{w}{u} u P_{lj1}'^u (1 - P_{lj1}')^{w-u}, & \text{for } w > 0. \end{cases} \tag{10}$$

$li_0(w)$ represents the average number of w input queues that contain only one buffered cell, with condition that there is a new arrival cell at the *tagged input queue* $Q(i, j)$. P_{li1} in (8) is the probability that the length of a queue of the w input queues of input i is equal to one (there is only one buffered cell in this input queue) during a time slot. Similarly, $lj_0(w)/lj_0'(w)$ represents the average number of w jth input queues that contain only one buffered cell given that $Q(i, j)$ is nonempty/empty at the beginning of the current time slot. P_{lj1} and P_{lj1}' in (9) and (10) are the probabilities that the length of a queue of the w jth queues is equal to one in these two cases. The three probabilities of P_{li1}, P_{lj1}, and P_{lj1}' are thus given by

$$\begin{aligned} P_{li1} &= \pi_1 e/(1 - \pi_0) \\ P_{lj1} &= (1 - \lambda)\pi_1 e/(1 - \pi_0) \\ P_{lj1}' &= (\lambda \pi_0 + (1 - \lambda)\pi_1 e)/(1 - \pi_0). \end{aligned}$$

For $P'_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$, there is no new cell arrival at the *tagged input queue*. The blocking probability $P'_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$ can be computed similarly as $P_{\text{blo}, W_t(w_i', w_o')|W_{t-1}(w_i, w_o)}$ provided that $li_0(w)$ is replaced with $li_0'(w)$

$$li_0'(w) = \begin{cases} 0, & \text{for } w = 0 \\ \sum_{u=1}^{w} \binom{w}{u} \cdot \frac{u((1-(N-w+1+u)\lambda)+\lambda(u-1))}{1-(N-w+1)\lambda} P_{li1}^{u}(1-P_{li1})^{w-u}, & \text{for } w > 0 \end{cases} \tag{11}$$

and $a_{K(1,\cdots,\cdots)|W(\cdots,\cdots)}$ is replaced with $a_{K(0,\cdots,\cdots)|W(\cdots,\cdots)}$.

Finally, we can compute the two success probabilities, $P_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$ and $P'_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)}$ as

$$P_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)} = a_{K(1,\,k_i,\,k_o)|W(w_i,\,w_o)} \cdot P_{\text{suc}|H(k_i+w_i,\,k_o+w_o)} \tag{12}$$

$$P'_{\text{suc},\,W_t(w_i',\,w_o')|W_{t-1}(w_i,\,w_o)} = a_{K(0,\,k_i,\,k_o)|W(w_i,\,w_o)} \cdot P_{\text{suc}|H(k_i+w_i,\,k_o+w_o)}. \tag{13}$$

3) Applying the PIM Algorithm: We now compute the probabilities in Table I by considering each iteration of the PIM scheduling algorithm. The state of the switch at the beginning of each iteration ϕ is characterized by the following parameters:

$n(\phi)$ number of unmatched inputs/outputs at the beginning of ϕth iteration;

$$P_{\text{blo},\,W_t(w_i',\,w_o')}|W_{t-1}(w_i,\,w_o) = \begin{cases} 0, \quad \text{for } r_i < -1 \text{ or } r_o < -1 \\ a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_00}|H(w_i',\,w_o')} + a_{K(1,\,r_i,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_01}|H(w_i',\,w_o'+1)} lj_0'(r_o)/w_o' \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_01}|H(w_i',\,w_o')} \left(1 - \frac{lj_0'(r_o-1)}{w_o'-1}\right) \\ \quad + a_{K(1,\,r_i+1,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_10}|H(w_i'+1,\,w_o')} li_0(r_i)/w_i' \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_10}|H(w_i',\,w_o')} \left(1 - \frac{li_0(r_i-1)}{w_i'-1}\right) \\ \quad + a_{K(1,\,r_i+1,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i'+1,\,w_o'+1)} \cdot \frac{li_0(r_i) lj_0'(r_o)}{w_i' \cdot w_o'} \\ \quad + a_{K(1,\,r_i+1,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i'+1,\,w_o')} \cdot \frac{li_0(r_i)(r_o - 1 - lj_0'(r_o-1))}{w_i' \cdot (w_o'-1)} \\ \quad + a_{K(1,\,r_i,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i',\,w_o'+1)} \cdot \frac{(r_i - 1 - li_0(r_i-1)) lj_0'(r_o)}{(w_i'-1) \cdot w_o'} \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i',\,w_o')} \cdot \frac{(r_i - 1 - li_0(r_i-1))(r_o - 1 - lj_0'(r_o-1))}{(w_i'-1) \cdot (w_o'-1)}, \\ \text{for } r_i \geq 1,\ r_o \geq 1,\ w_i = 0, \text{ and } w_o = 0 \\ a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_00}|H(w_i',\,w_o')} \\ \quad + a_{K(1,\,r_i,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_01}|H(w_i',\,w_o'+1)} \cdot (r_o + 1 + lj_0(w_o-1))/w_o' \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_01}|H(w_i',\,w_o')} \cdot (w_o - 1 - lj_0(w_o-1))/(w_o'-1) \\ \quad + a_{K(1,\,r_i+1,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_10}|H(w_i'+1,\,w_o')} \cdot (r_i + 1 + li_0(w_i-1))/w_i' \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_10}|H(w_i',\,w_o')} \cdot (w_i - 1 - li_0(w_i-1))/(w_i'-1) \\ \quad + a_{K(1,\,r_i+1,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i'+1,\,w_o'+1)} \cdot \frac{(r_i + 1 + li_0(w_i-1))(r_o + 1 + lj_0(w_o-1))}{w_i' \cdot w_o'} \\ \quad + a_{K(1,\,r_i+1,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i'+1,\,w_o')} \\ \quad \cdot \frac{(r_i + 1 + li_0(w_i-1))(w_o - 1 - lj_0(w_o-1))}{w_i' \cdot (w_o'-1)} + a_{K(1,\,r_i,\,r_o+1)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i',\,w_o'+1)} \\ \quad \cdot \frac{(w_i - 1 - li_0(w_i-1))(r_o + 1 + lj_0(w_o-1))}{(w_i'-1) \cdot w_o'} \\ \quad + a_{K(1,\,r_i,\,r_o)|W(w_i,\,w_o)} P_{\text{blo\_11}|H(w_i',\,w_o')} \cdot \frac{(w_i - 1 - li_0(w_i-1))(w_o - 1 - lj_0(w_o-1))}{(w_i'-1) \cdot (w_o'-1)}, \\ \text{for } r_i \geq -1,\ r_o \geq -1,\ w_i > 0, \text{ and } w_o > 0 \end{cases} \tag{7}$$

$h_i(\phi)$ number of nonempty queues in input i at the beginning of ϕth iteration, whose outputs are still unmatched;

$h_o(\phi)$ number of nonempty jth queues in $n(\phi)$ inputs (including input i) at the beginning of ϕth iteration matching.

At the end of the iteration, the following parameters can be defined:

$m(\phi)$ number of inputs/outputs that get matched at the end of ϕth iteration, $m(\phi) = n(\phi) - n(\phi+1)$;

$\Delta h_i(\phi)$ number of outputs whose corresponding nonempty queues in input i that get matched at the end of ϕth iteration, $\Delta h_i(\phi) = h_i(\phi) - h_i(\phi+1)$;

$\Delta h_o(\phi)$ number of inputs in set F that get matched at the end of ϕth iteration, $\Delta h_o(\phi) = h_o(\phi) - h_o(\phi+1)$.

For the sake of simplicity, we do not mention the iteration number in the following discussion. If no iteration number is mentioned, then the current iteration ϕ is implied.

Let $x_i x_j$ represent the state of the matching process for input i and output j of the switch, where $x_i, x_j \in \{0, 1\}$ with 0 representing that the input/output is unmatched and 1 representing that the input/output is matched at the end of the current iteration. The possible states of the matching process are 00, 01, 10, and 11. However, the state 11 should explicitly consider if the tagged input queue $Q(i, j)$ at input i is matched. Thus, the state 11 is split into two: 11_{suc} and 11_{blo}, respectively. Given the current state of the switch $(n(\phi), h_i(\phi), h_o(\phi))$ and the current state of the matching process $x_i x_j$, the resulting state of the switch $(n(\phi+1), h_i(\phi+1), h_o(\phi+1))$ and the resulting state of the matching process $x_i' x_j'$ is controlled by the transition probabilities defined:

$P_{blo\_00|00}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and both input i and output j remain unmatched given that both input i and output j were unmatched at the beginning of current iteration};

$P_{blo\_01|00}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, input i remains unmatched and output j gets matched given that both input i and output j were unmatched at the beginning of current iteration};

$P_{blo\_10|00}$ Pr{at the end of current iteration, the HOL cell at the tagged input queue gets blocked, input i gets matched and output j remains unmatched given that both input i and output j were unmatched at the beginning of current iteration};

$P_{blo\_11|00}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and both input i and output j get matched given that both input i and output j were unmatched at the beginning of current iteration};

$P_{blo\_01|01}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and input i remains unmatched given that input i was unmatched and output j was matched at the beginning of current iteration};

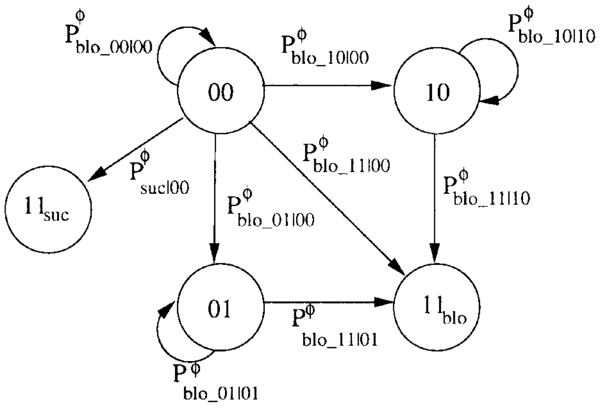

$P^{\phi}_{\text{blo\_00|00}}$
$P^{\phi}_{\text{blo\_10|00}}$
$P^{\phi}_{\text{blo\_10|10}}$
00
10
$P^{\phi}_{\text{suc|00}}$
$P^{\phi}_{\text{blo\_11|00}}$
$P^{\phi}_{\text{blo\_01|00}}$
$P^{\phi}_{\text{blo\_11|10}}$
11_{suc}
01
11_{blo}
$P^{\phi}_{\text{blo\_11|01}}$
$P^{\phi}_{\text{blo\_01|01}}$

Fig. 5. The matching process state transition diagram.

$P_{blo\_11|01}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and input i gets matched given that input i was unmatched and output j was matched at the beginning of current iteration};

$P_{blo\_10|10}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and output j remains unmatched given that input i was matched and output j was unmatched at the beginning of current iteration};

$P_{blo\_11|10}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue gets blocked, and output j gets matched given that input i was matched and output j was unmatched at the beginning of current iteration};

$P_{suc|00}$ Pr{ at the end of current iteration, the HOL cell at the tagged input queue $Q(i, j)$ gets matched with output j given that input i and output j were unmatched at the beginning of current iteration}.

These probabilities are functions of the current state of the switch $(n(\phi), h_i(\phi), h_o(\phi))$. The transitions among the states of the matching process can be represented by the state transition diagram shown in Fig. 5 with the state space $X\{00, 01, 10, 11_{\text{blo}}, 11_{\text{suc}}\}$. We derive equations for the transition probabilities next.

We define the following probabilities associated with the first stage of contention for a cell:

P_{suc1_e} Pr{the HOL cell at kth $(k \neq j)$ queue of an input in set E succeeds in the first stage contention}

P_{suc1_ft} Pr{the HOL cell at jth queue of an input in set F succeeds in the first stage contention}

P_{suc1_fe} Pr{the HOL cell at kth $(k \neq j)$ queue of an input in set F succeeds in the first stage contention}

Here, P_{suc1_e}, P_{suc1_ft}, and P_{suc1_fe} are functions of p_0, and are computed as

$$P_{\text{suc1}\_e} = \frac{1 - \pi_0^{n-1}}{n-1}$$

$$P_{\text{suc1}\_ft} = \sum_{v=1}^{n} \binom{n-1}{v-1} (1-\pi_0)^{(v-1)} \pi_0^{(n-v)} \Big/ v$$

$$P_{\text{suc1}\_fe} = \frac{1 - P_{\text{suc1}\_ft}}{n-1}.$$

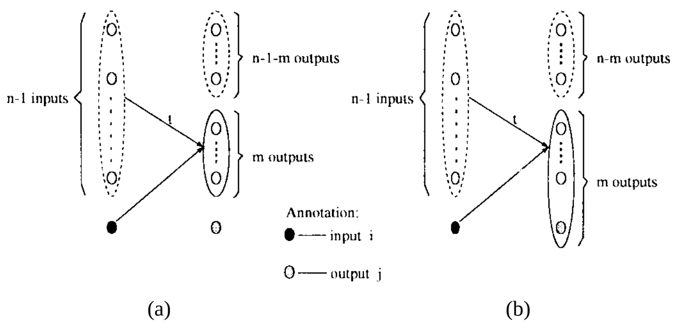


Fig. 6. Diagram for deriving $P_{\text{blo\_00}|00}$, $P_{\text{blo\_01}|00}$, $P_{\text{blo\_10}|00}$, and $P_{\text{blo\_11}|00}$.

Let $t(\max(m,\, h_o-1) \le t \le n-1)$ be the number of queues excluding the queue from input i that succeed in the first stage of contention, and m is the number of outputs contended for by the t inputs. There are three subproblems to be considered in computing the transition probabilities in ϕth iteration given $(n(\phi),\, h_i(\phi),\, h_o(\phi))$ and $(n(\phi+1),\, h_i(\phi+1),\, h_o(\phi+1))$.

1) What is the probability that t inputs contend for m outputs?
2) What is the probability that Δh_o inputs in set F (whose cardinality is $h_o - 1$) get matched?
3) What is the probability that Δh_i out of h_i outputs whose corresponding queues in input i are nonempty get matched?

The equations below consider each of the subproblems in computing the transition probabilities.

We first consider the situation where both input i and output j are unmatched at the beginning of the current iteration. Fig. 6 shows the possible scenarios that can arise in this case. Case a) represents the situation where output j remains unmatched at the end of the current iteration, and case b) represents the situation where output j gets matched.

a) Computing $P_{\text{blo\_00}|00}$: In this case both input i and output j remain unmatched. From Fig. 6(a) given t and m, the probability that the queue that succeeds in the first stage of contention at input i gets blocked at its corresponding output is

$$\begin{aligned}P_{t\to m|blo\_00\_00} &= \sum_{k=1}^{t-m+1} \binom{t}{k}\left(1-\frac{1}{k+1}\right)(m-1)! \\ &\quad \cdot S_{t-k}^{m-1} P_{\text{suc1\_}e}^{t-h_o+1} \pi_0^{(n-1)(n-1-t)} P_{\text{suc1\_}fe}^{h_o-1} \\ &= \left(1-\frac{m}{t+1}\right)(m!S_t^m + (m-1)!S_t^{m-1}) \\ &\quad \cdot P_{\text{suc1\_}e}^{t-h_o+1} \pi_0^{(n-1)(n-1-t)} P_{\text{suc1\_}fe}^{h_o-1}.\end{aligned}$$

Among the m outputs that get matched, Δh_i of them will see their corresponding queues in input i being nonempty. The number of combinations satisfying this condition is

$$C_{\Delta h_i|blo\_00\_00} \binom{h_i-2}{\Delta h_i - 1}\binom{n-h_i}{m-\Delta h_i}. \tag{14}$$

Given that input i is blocked, it is clear that each combination of m out of t inputs gets matched with dual probability. The probability that Δh_o inputs which are elements of the set F get matched is

$$P_{\Delta h_o|blo\_00\_00} = \frac{\binom{h_o-1}{\Delta h_o}\binom{t-h_o+1}{m-\Delta h_o}}{\binom{t}{m}}.$$

Knowing the above probabilities, $P_{\text{blo\_00}|00}$ can be easily computed as

$$\begin{aligned}P_{\text{blo\_00}|00} &= \left(1-\frac{1}{h_i}\right)\sum_{t=\max(m,\, h_o-1)}^{n-1} \binom{n-h_o}{t-h_o+1} \\ &\quad \cdot C_{\Delta h_i|blo\_00\_00} P_{\Delta h_o|blo\_00\_00} P_{t\to m|blo\_00\_00}.\end{aligned}$$

b) Computing $P_{\text{blo\_10}|00}$: In this case, input i gets matched while output j remains unmatched. Hence, we compute only the aggregated probability over the set of all possible Δh_o. The probability that the queue that succeeds in the first stage of contention at input i succeeds in getting matched in the second stage of contention is

$$\begin{aligned}P_{t\to m|blo\_10\_00} &= \sum_{k=1}^{t-m+1} \binom{t}{k}\frac{1}{k+1}(m-1)!S_{t-k}^{m-1} P_{\text{suc1\_}e}^{t-h_o+1} \\ &\quad \cdot \pi_0^{(n-1)(n-1-t)} P_{\text{suc1\_}fe}^{h_o-1} \\ &= \left(1-\frac{m}{t+1}\right)(m!S_t^m + (m-1)!S_t^{m-1}) \\ &\quad \cdot P_{\text{suc1\_}e}^{t-h_o+1} \pi_0^{(n-1)(n-1-t)} P_{\text{suc1\_}fe}^{h_o-1}.\end{aligned}$$

The probability that Δh_o inputs which are elements of set F get matched is

$$P_{\Delta h_o|blo\_10\_00} = \frac{\binom{h_o-1}{\Delta h_o - 1}\binom{t-h_o+1}{m-\Delta h_o}}{\binom{t}{m-1}}.$$

Therefore, $P_{\text{blo\_10}|00}$ is given by

$$\begin{aligned}P_{\text{blo\_10}|00} &= \left(1-\frac{1}{h_i}\right)\binom{n-2}{m-1} \\ &\quad \sum_{t=\max(m-1,\, h_o-1)}^{n-1} \binom{n-h_o}{t-h_o+1} \\ &\quad \cdot P_{\Delta h_o|blo\_10\_00} \cdot P_{t\to m|blo\_10\_00}\end{aligned}$$

c) Computing $P_{\text{blo\_01}|00}$: In this case, output j gets matched while input i remains unmatched. We compute only the aggregated probability over the set of all possible Δh_i. There are two cases to be considered here: (i) $Q(i,\, j)$ fails the first stage of contention, and (ii) $Q(i,\, j)$ survives the first stage of contention. Therefore $P_{\text{blo\_01}|00}$ is the sum of two probabilities

$$P_{\text{blo\_01}|00} P_{\text{blo\_01\_}B|00} + P_{\text{blo\_01\_}S|00}$$

where $P_{\text{blo\_01\_}B|00}$ and $P_{\text{bloc\_01\_}S|00}$ are probabilities for the two cases (i) and (ii), respectively.

Case (i): $Q(i, j)$ fails in the first stage of contention

$$P_{\text{blo\_01\_}B|00} = (1 - 1/h_i) \sum_{t=\max(m, h_o-1)}^{n-1} \binom{n-h_o}{t-h_o+1} \cdot \sum_{u=1}^{\min(h_o-1, t-m+1)} \binom{h_o-1}{u} C_{\Delta h_i|blo\_01\_B\_00} \cdot P^u_{t\to m|blo\_01\_B\_00}$$

where

$$\begin{aligned} P^u_{t\to m|blo\_01\_B\_00} &= \sum_{k=1}^{t-(m-2+u)} \binom{t-u}{k}\left(1-\frac{1}{k+1}\right)(m-2)!S^{m-2}_{t-k-u} \cdot P^u_{\text{suc1\_}ft} P^{h_o-1-u}_{\text{suc1\_}fe} P^{t-h_o+1}_{\text{suc1\_}e} \pi_0^{(n-1)(n-1-t)} \\ &= \left(1-\frac{m-1}{t-u+1}\right)((m-1)!S^{m-1}_{t-u} + (m-2)!S^{m-2}_{t-u}) \cdot P^u_{\text{suc1\_}ft} P^{h_o-1-u}_{\text{suc1\_}fe} P^{t-h_o+1}_{\text{suc1\_}e} \pi_0^{(n-1)(n-1-t)} \end{aligned}$$

$$C_{\Delta h_i|\text{blo\_01\_}B\text{\_00}} = \binom{h_i-2}{\Delta h_i-2}\binom{n-h_i}{m-\Delta h_i}.$$

Case (ii): $Q(i, j)$ is successful in the first stage of contention

$$P_{\text{blo\_01\_}S\text{\_00}} = \frac{1}{h_i} \sum_{t=\max(m, h_o-1)}^{n-1} \binom{n-h_o}{t-h_o+1} \cdot C_{\Delta h_i|blo\_01\_S\_00} P_{t\to m|blo\_01\_S\_00}$$

where

$$P_{t\to m|blo\_01\_S\_00} = \sum_{k=1}^{\min(h_o-1, t-m+1)} \binom{t}{k}\left(1-\frac{1}{k+1}\right)(m-1)!S^{m-1}_{t-k} \cdot P^k_{\text{suc1\_}ft} P^{h_o-1-k}_{\text{suc1\_}fe} P^{t-h_o+1}_{\text{suc1\_}e} \pi_0^{(n-1)(n-1-t)}$$

$$C_{\Delta h_i|blo\_01\_S\_00} = \binom{h_i-1}{\Delta h_i-1}\binom{n-h_i}{m-\Delta h_i}.$$

d) Computing $P_{\text{suc}|00}$: Recalling that in the matching process Markov chain, state 11_{suc} is an absorbing state, so this transition probability is computed without consideration on m

$$P_{\text{suc}|00}\frac{1}{h_i}\sum_{u=0}^{h_o-1}\binom{h_o-1}{u}\frac{1}{u+1}P^u_{\text{suc\_}ft}(1-P_{\text{suc\_}ft})^{h_o-1-u}.$$

e) Computing $P_{\text{blo\_11}|00}$: Then, $P_{\text{blo\_11}|00}$ is computed from the boundary condition as

$$P_{\text{blo\_11}|00} = 1-(P_{\text{blo\_00}|00}+P_{\text{blo\_01}|00}+P_{\text{blo\_10}|00}+P_{\text{suc}|00}).$$

Now we consider the cases where output j has already been matched, while input i is still unmatched. Note that in this case the tagged input queue $Q(i, j)$ is no longer under consideration for matching because output j is already matched.

f) Computing $P_{\text{blo\_01}|01}$: In this case, input i remains unmatched, while output j is already matched. Then

$$P_{\text{blo\_01}|01} = \sum_{t=m}^{n-1}\binom{n-1}{t}C_{\Delta h_i|blo\_01\_01}P_{t\to m|blo\_01\_01}$$

where

$$\begin{aligned} P_{t\to m|blo\_01\_01} &= \sum_{k=1}^{t-m+1}\binom{t}{k}\left(1-\frac{1}{k+1}\right)(m-1)! \cdot S^{m-1}_{t-k}P^t_{\text{suc\_0}}p_0^{n(n-1-t)} \\ &= \left(1-\frac{m}{t+1}\right)(m!S^m_t + (m-1)!S^{m-1}_t) \cdot P^t_{\text{suc\_0}}p_0^{n(n-1-t)} \end{aligned}$$

$$P_{\text{suc\_0}} = \frac{1-\pi_0^n}{n}$$

$$C_{\Delta h_i|blo\_01\_01} = \binom{h_i-1}{\Delta h_i-1}\binom{n-h_i}{m-\Delta h_i}.$$

g) Computing $P_{\text{blo\_11}|01}$: In this case input i gets matched while output j has already been matched at the beginning of the iteration

$$P_{\text{blo\_11}|01} = \sum_{u=0}^{n-1}\binom{n-1}{u}\frac{1}{u+1}P^u_{\text{suc\_0}}(1-P_{\text{suc\_0}})^{n-1-u}.$$

Next we consider the cases where input i is already matched, while output j is still unmatched. Once input i has been matched, for the iterations thereafter, we are interested only in the state of output j, i.e., whether it is matched or unmatched.

h) Computing $P_{\text{blo\_10}|10}$: In this case input i is already matched, while output j remains unmatched at the end of the iteration. Then

$$P_{\text{blo\_10}|10} = \binom{n-1}{m}\sum_{t=\max(h_o-1, m)}^{n}\binom{n-h_o+1}{t-h_o+1} \cdot P_{\Delta h_o|blo\_10\_10}P_{t\to m|blo\_10\_10}$$

where

$$P_{t\to m|blo\_10\_10} = m!S^m_t P^{t-h_o+1}_{\text{suc1\_}e}\pi_0^{(n-1)(n-t)}P^{h_o-1}_{\text{suc1\_}fe}$$

and

$$P_{\Delta h_o|blo\_10\_10} = \frac{\binom{h_o-1}{\Delta h_o}\binom{t-h_o+1}{m-\Delta h_o}}{\binom{t}{m}}.$$

i) Computing $P_{\text{blo\_11}|10}$: In this case, output j gets matched at the end of the iteration. This is feasible only if at least one of jth queues of the $h_o - 1$ inputs in set F succeed in the first stage of contention at their respective inputs

$$P_{\text{blo\_11}|10} = 1-(1-P_{\text{suc1\_}ft})^{h_o-1}.$$

The states of the switch at the end of each iteration $(n(\phi), h_i(\phi), h_o(\phi), x_ix_j)$ can be viewed as a weighted tree with the nodes of the tree corresponding to the switch states. The root of the tree is the initial state of the switch $(N, h_i, h_o, 00)$. All states in level ϕ of the tree correspond to the states of the switch at the end of the ϕth iteration of the PIM algorithm. Weights are assigned to the arcs between the states and are equal to the transition probabilities $P_{\text{blo\_}x'_i x'_j|x_i x_j}$ or $P_{\text{suc}|x_i x_j}$. Each state $(n(\phi), h_i(\phi), h_o(\phi), x_i x_j)$ is assigned a probability $\Pr(n(\phi), h_i(\phi), h_o(\phi), x_i x_j)$ equal to the product of the transition probabilities along the arcs from the root to the state. The probabilities $P_{\text{blo\_00}|H(h_i,h_o)}$, $P_{\text{blo\_01}|H(h_i,h_o)}$, $P_{\text{blo\_10}|H(h_i,h_o)}$, and $P_{\text{suc}|H(h_i,h_o)}$ at the end of Φ iterations of the PIM algorithm can be computed as

$$\begin{aligned}
P_{\text{blo\_}x_i x_j|H(h_i,h_o)} &= \sum_{n(\Phi),\, h_i(\Phi),\, h_o(\Phi)} \Pr(n(\Phi), h_i(\Phi), h_o(\Phi), x_i x_j) \\
P_{\text{suc}|H(h_i,h_o)} &= \sum_{n(\Phi),\, h_i(\Phi),\, h_o(\Phi)} \Pr(n(\Phi), h_i(\Phi), h_o(\Phi), 11_{\text{suc}}).
\end{aligned}$$

C. Solving the Markov Chain

As can be seen from the above equations, π_0 and $\pi_1 e$ must be known in advance in order to compute the steady-state probabilities. In the Appendix, we give detailed procedures to derive the computation formulas for these parameters. The steady-state probabilities are given by

$$\pi_0 = 1 \Bigg/ \left(1 + \sum_{i=1}^{b_i} \prod_{j=1}^{i} \alpha_j e\right) \tag{15}$$

$$\pi_i = \pi_0 \prod_{j=1}^{i} \alpha_j, \quad \text{for } 1 \le i \le b_i \tag{16}$$

where

$$\alpha_i = \begin{cases} A_2(I - D_1)^{-1}, & \text{for } i = b_i \\ A_2(I - A_1 - \alpha_{b_i} D_0)^{-1}, & \text{for } i = b_i - 1 \\ A_2(I - A_1 - \alpha_{i+1} A_0)^{-1}, & \text{for } 2 \le i \le b_i - 2 \\ C_2(I - A_1 - \alpha_2 A_0)^{-1}, & \text{for } i = 1. \end{cases}$$

Recalling the definitions of the matrice appear in α_i, the elements of this matrice are functions of π_0 and $\pi_1 e$. This naturally suggests an iterative solution [16]. Initially, π_0 is set to $1 - \lambda$, which corresponds to the case that there is no new arriving cell at the *tagged input queue* at the beginning of a time slot, and $\pi_1 e$ is approximated by $(1 - 1/N)\lambda\pi_0$. Then the next π_0 is obtained by using (15) and compute the new $\pi_1 e$ by (16). This iterating process continues until the π_0 converge to a fixed point. Finally, the values of steady-state probabilities π_i $(1 \le i \le b_i)$ are computed by (16).

D. Computing the Performance Metrics

Once the steady-state probabilities are known, then interesting performance parameters, such as throughput, mean queue length and mean cell loss probability can be computed directly by using the known parameters. Let ρ, $\overline{Q}$, and P_{loss} be throughput, mean queue length, and mean cell loss probability,

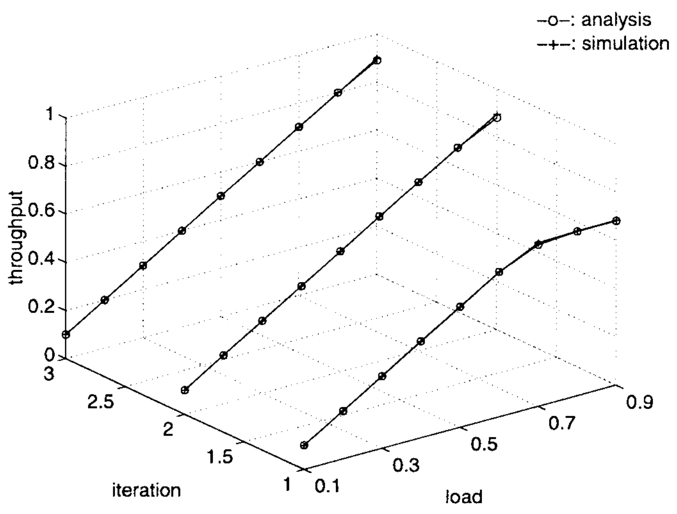


Fig. 7. The throughput of a 4 × 4 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

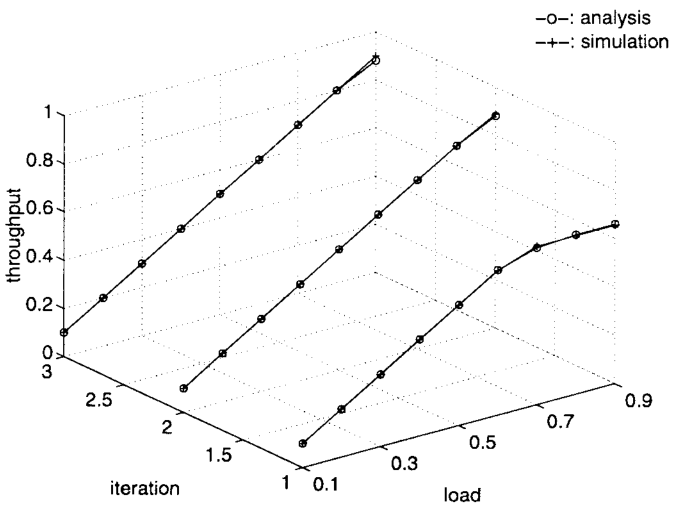


Fig. 8. The throughput of a 8 × 8 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

respectively, then

$$\begin{aligned}
\rho &= \lambda\pi_0(1 - (B_0/\lambda)e) + \sum_{l=1}^{b_i} \pi_l (S + S')e \\
\overline{Q} &= \sum_{l=1}^{b_i} l\pi_l e \\
P_{\text{loss}} &= \pi_{b_i} e.
\end{aligned}$$

Using Little's theorem [21] which holds for a queueing system in steady-states, the mean cell delay $\overline{D}$, the mean queue length $\overline{Q}$, and the queue's throughput ρ are related together as follows:

$$\overline{D} = \overline{Q}/\rho.$$

V. Numerical Results

Both mathematical analysis and simulation results are presented in this section in order to investigate the accuracy of the above queueing model. Figs. 7–9 show the switch throughput as a function of offered load λ for PIM switch sizes 4, 8, and 16, with various PIM scheduling iteration numbers 1, 2,

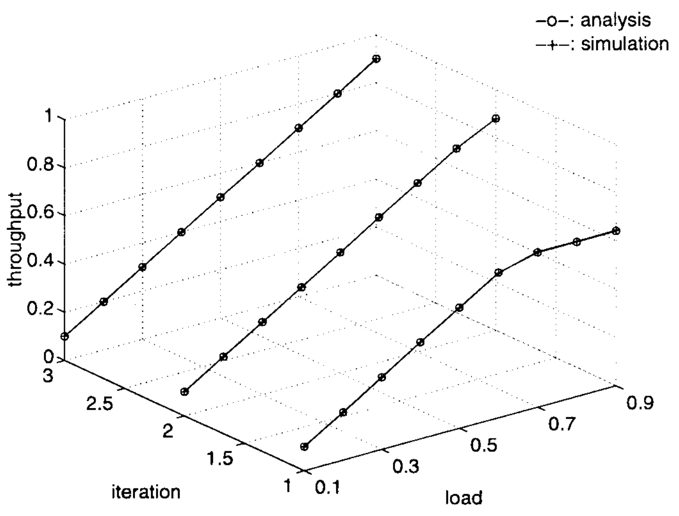


Fig. 9. The throughput of a 16 × 16 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

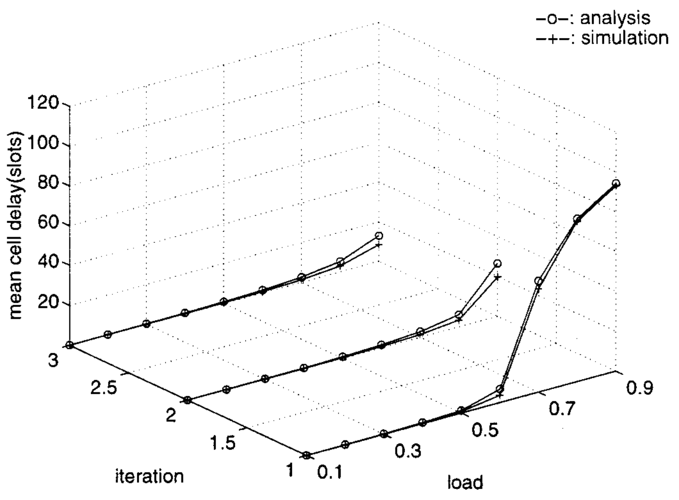


Fig. 11. The mean cell delay of a 8 × 8 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

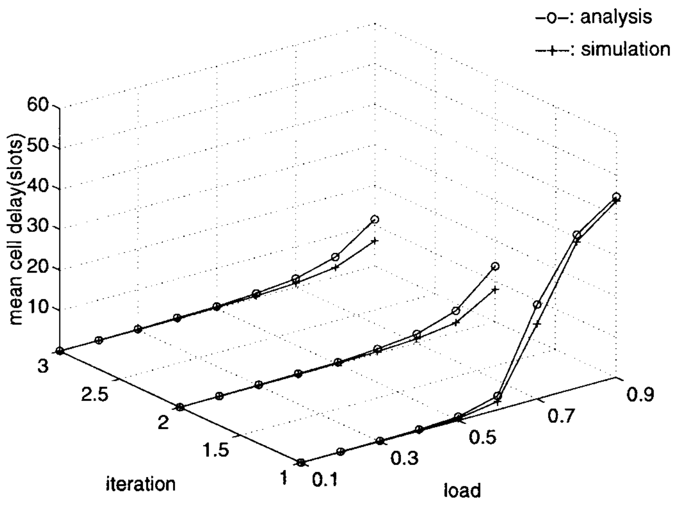


Fig. 10. The mean cell delay of a 4 × 4 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

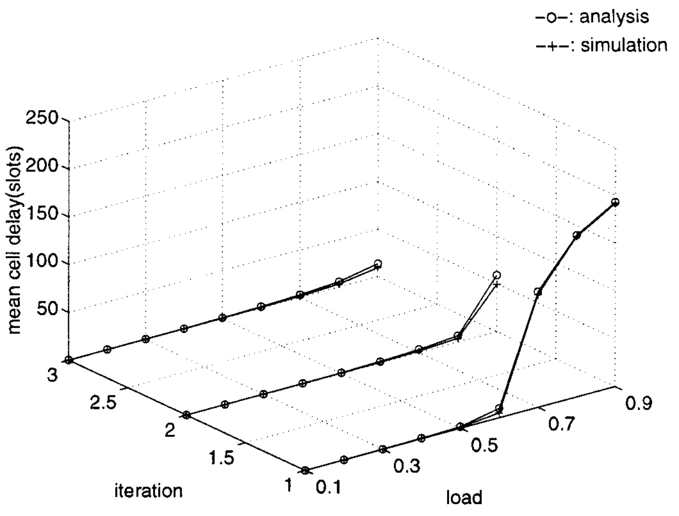


Fig. 12. The mean cell delay of a 16 × 16 PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

and 3, respectively. It can be seen that when the switch size increases, the throughput of the switch decreases under high offered load (greater than 60% when maximum iteration is 1). Also from this figure, we can see that the saturation throughput will increase as the PIM scheduling iteration increases. It is expected that with more iterations, more HOL cells get matched during a scheduling iteration. The curves show that three iterations are enough to get a high throughput >90%. Comparing Figs. 7–9 with Fig. 2, we can see that even under saturated traffic loads, our queueing model approximates the original system quite well.

Figs. 10–12 show the mean cell delay as a function of offered load λ for the different PIM switch sizes 4, 8, and 16 with various PIM iteration numbers one, two, and three. The figures indicate that the mean cell delay increases as the switch size increases and also as the offered load increases. But when the number of PIM scheduling iterations is increased, even from one to two, the mean delay increased slowly with the traffic load as compared with just one iteration. For single iteration PIM scheduling, the mean cell delay increases dramatically when the offered load exceeds 60%, which indicates that PIM switches with single iteration PIM scheduling will be overloaded when the traffic load is greater than 60%. However, for two and three iteration PIM, this *overloaded traffic point* is about 0.8. This phenomenon can also be observed in Figs. 7–9. Notice that when the traffic load is extremely low, such as 0.1, all curves cluster into a single point. It is not difficult to understand that, under low traffic load, the opportunity that more than one HOL cell contends for a common input/output is small. That is, single iteration PIM scheduling is typically enough to find a maximal matching. When the traffic load grows, the chances of conflicts increase and more iterations are needed using PIM scheduling to achieve a maximal matching.

In Fig. 13, the mean cell loss probabilities of PIM switches with queue size of ten cells are given as a function of offered load. It can be seen that, for a medium size PIM switch with three iterations PIM scheduling (such as 16 × 16) with traffic load less than 60%, a buffer size of 10 cells per queue is sufficient to guarantee a cell loss probability $<10^{-9}$.

As mentioned in the introduction of this paper, we are extending these results to more realistic traffic patterns. This

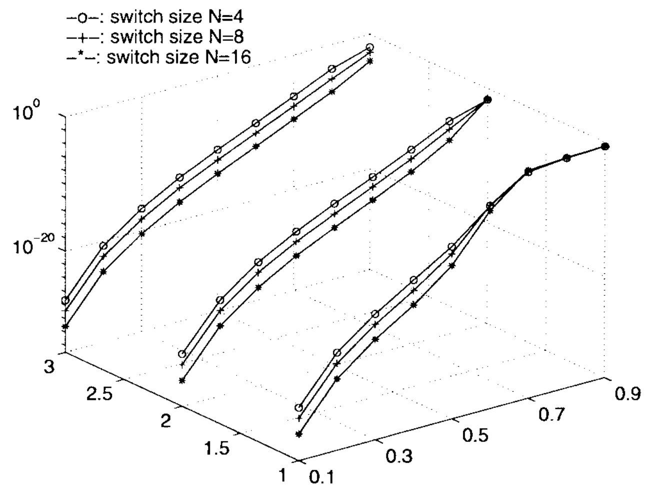


Fig. 13. The mean cell loss probability of a PIM switch, as a function of offered load, with a buffer size $b_i = 10$.

complicates the mathematical analysis, but would also lead to more practical result.

VI. Conclusion

A queueing model for the performance analysis of a multiple input queued ATM switch with PIM scheduling under random traffic loads has been presented. The queueing model provides a rather general method in analyzing such ATM switches in terms of throughput, mean queue length, mean cell delay, and mean cell loss probability given the switch size, queues' buffer size, and offered load. As seen from Section V, the analytical results of the queueing model is able to approximate the simulation results satisfactorily.

The contribution of this paper is twofold. First, the throughput of an ATM switch with multiple iteration PIM scheduling in case of saturated traffic load is analyzed mathematically. Second, a theoretical analysis for various performance parameters including throughput, mean cell delay, and mean cell loss probability of an ATM switch using a PIM scheduling scheme is presented. Such theoretical analysis is lacking in existing literature on ATM switches with PIM or variations of PIM scheduling [7], [9]–[12].

Several assumptions were made in order to make the original switch model tractable for analysis. The most important is the uniform traffic assumption, i.e., cells arrive at each input according to an *i.i.d.* Bernoulli process, and the output destinations of arriving cells are uniformly distributed over all outputs. We are currently working on extending the model for ATM switches with bursty and correlated traffic.

Appendix
Computations of the Steady-State Probabilities

Here we give the procedures to compute the steady-state probabilities of the Markov chain Z.

Given that $\pi_{(l, w_i, w_o)}$ is the steady-state probability of the state (l, w_i, w_o), where l is the length of the tagged input queue, w_i is the length of the virtual HOL input queue, and w_o is the length of the virtual HOL output queue. The steady-state probability vector of the Markov chain Z is given by $\Pi = [\pi_0, \pi_1, \pi_2, \cdots, \pi_l, \cdots, \pi_{b_i}]$, where every element $\pi_l = [\pi_{(l,1,1)}, \pi_{(l,1,2)}, \cdots, \pi_{(l,N,N)}], l > 0$ is a row vector of size N^2, and π_0 is a scalar.

From the definition of the transition probability matrix, we know that

$$\Pi T = \Pi. \tag{17}$$

By expanding (17), we have

$$\pi_0 C_1 + \pi_1 C_0 = \pi_0 \tag{18}$$

$$\pi_0 C_2 + \pi_1 A_1 + \pi_2 A_0 = \pi_1 \tag{19}$$

$$\pi_{i-1} A_2 + \pi_i A_1 + \pi_{i+1} A_0 = \pi_i, \quad \text{for } 1 < i < b_i - 1 \tag{20}$$

$$\pi_{b_i-2} A_2 + \pi_{b_i-1} A_1 + \pi_{b_i} D_0 = \pi_{b_i-1} \tag{21}$$

$$\pi_{b_i-1} A_2 + \pi_{b_i} D_1 = \pi_{b_i}. \tag{22}$$

In order to solve the linear equations above, we note that once π_0 and π_1 are known, the remaining π_i $(1 < i \le b_i)$ can be solved by substitute the known ones into (20)–(22) step by step. The key problem in solving these linear equations becomes how to find the values of π_0 and π_1. Fortunately, the special structure of Z facilitates our solving of these equations. As mentioned in Section IV-A, Z is a *QBD* process which holds an important attribute that in moving from a state to a higher-level state, the chain must visit all intermediate levels at least once. More detail attributes in our case are:

1) for any π_i $(0 < i < b_i)$, it appears in three equations;
2) for π_0 and π_{b_i}, which we notate as *boundary probabilities*, however, they appear in two equations.

Utilize the attribute of the *boundary probabilities* π_{b_i}, we can express π_{b_i} as function of π_{b_i-1} from (22)

$$\pi_{b_i} = \pi_{b_i-1} A_2 (I - D_1)^{-1}. \tag{23}$$

Substitute (23) into (21), π_{b_i-1} can be expressed as function of π_{b_i-2}

$$\pi_{b_i-1} = \pi_{b_i-2} A_2 (I - A_1 - \alpha_{b_i} D_0)^{-1}. \tag{24}$$

Similarly, π_i $(2 \le i < b_i - 2)$ can be expressed as function of π_{i-1} by further the operations recursively

$$\pi_i = \pi_{i-1} A_2 (I - A_1 - \alpha_{b_i} A_0)^{-1}. \tag{25}$$

Let α_i $(2 \le i \le b_i)$ be a $N^2 \times N^2$ matrix defined as

$$\alpha_i = \begin{cases} A_2 (I - D_1)^{-1}, & \text{for } i = b_i \\ A_2 (I - A_1 - \alpha_{b_i} D_0)^{-1}, & \text{for } i = b_i - 1 \\ A_2 (I - A_1 - \alpha_{i+1} A_0)^{-1}, & \text{for } 2 \le i \le b_i - 2. \end{cases}$$

Now π_2 can be written as

$$\pi_2 = \pi_1 \alpha_1. \tag{26}$$

To replace π_2 in (19) by (26) and rearrange the equation, we have

$$\pi_1 = \pi_0 C_2 (I - A_1 - \alpha_2 A_0)^{-1}. \tag{27}$$

To be consistent with the denotation of α_i $(2 \le i \le b_i)$, we define α_1 as

$$\alpha_1 = C_2 (I - A_1 - \alpha_2 A_0)^{-1}.$$

Note that α_1 is a row vector of length of N^2 at this time. Now we can express all π_i $(1 \le i \le b_i)$ as function of π_0

$$\pi_i = \pi_0 \prod_{j=1}^{i} \alpha_j, \quad \text{for } 1 \le i \le b_i. \tag{28}$$

By utilizing the condition that $\pi_0 + \sum_{i=1}^{b_i} \pi_i e = 1$, we get

$$\pi_0 = 1 \Bigg/ \left(1 + \sum_{i=1}^{b_i} \prod_{j=1}^{i} \alpha_j e\right). \tag{29}$$

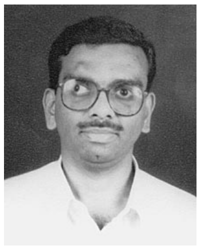
Once π_0 is known, all other state probabilities can be computed by (28).

References

[1] R. Rooholamini and V. Cherkassky, "Finding the right ATM switch for the market," *IEEE Trans. Comput.,* vol. 27, pp. 16–28, Apr. 1994.
[2] R. Y. Awdeh and H. T. Mouftah, "Survey of ATM switch architectures," *Computer Networks and ISDN Systems,* vol. 27, pp. 1567–1613, 1995.
[3] M. Hamdi and J. Muppala, "Performance evaluation of nonblocking ATM switches under various traffic and buffering schemes," *Int. J. Commun. Syst.,* vol. 9, pp. 59–79, 1996.
[4] 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.
[5] A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches," *IEEE/ACM Trans. Networking,* vol. 1, pp. 314–328, June 1993.
[6] P. Achille and B. Giacomo, "Analysis of input and output queueing for nonblocking ATM switches," *ACM Trans. Networking,* vol. 1, June 1993.
[7] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.,* vol. 11, no. 4, pp. 319–352, Nov. 1993.
[8] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "Tiny tera: A packet switch core," *IEEE Micro,* vol. 17, pp. 26–33, Jan. 1997.
[9] N. W. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, Univ. California at Berkeley, 1994.
[10] N. Mckeown, P. Varaiya, and J. Walrand, "Scheduling cells in an input-queued switch," *Electron. Lett.,* vol. 29, no. 25, pp. 2174–2175, 1993.
[11] B. Prabhakar, N. Mckeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE J. Select. Areas Commun.,* vol. 15, pp. 855–866, June 1997.
[12] R. O. LaMaire and D. N. Serpanos, "Two-dimentional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Trans. Networking,* vol. 2, pp. 471–481, Oct. 1994.
[13] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications.* Amsterdam, The Netherlands: Elsevier, 1976.
[14] M. Hamdi, J. Wang, and K. Ben Letaief, "Efficient estimation of cell loss and cell delay on nonblocking ATM switches using importance sampling," presented at GLOBECOM '97.
[15] Y. C. Jung and C. K. Un, "Analysis of backpressure-type packet switches with input and output buffering," *Proc. Inst. Elect. Eng.,* vol. 140, Aug. 1993.
[16] ——, "Performance analysis of packet switches with input and output buffers," *Computer Networks and ISDN Systems,* vol. 26, pp. 1559–1580, 1994.
[17] M. J. Lee and D. S. Ahn, "Cell loss analysis and design trade-offs of nonblocking ATM switches," *IEEE/ACM Trans. Networking,* vol. 3, pp. 199–210, Apr. 1995.
[18] M. K. Mehmet-Ali, M. Youssefi, and H. T. Nguyen, "The performance analysis and implementation of an input access scheme in a high-speed packet switch," *IEEE Trans. Commun.,* vol. 42, pp. 3189–3199, Dec. 1994.
[19] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models.* Baltimore, MD: The Johns Hopkins Press, 1981.
[20] E. Del Re and R. Fantacci, "Performance evaluation of input and output queueing techniques in ATM switching systems," *IEEE Trans. Commun.,* vol. 41, pp. 1565–1575, Oct. 1993.
[21] J. Medhi, *Stochastic Models in Queueing Theory.* New York: Academic, 1991.
[22] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York: Wiley, 1972.
[23] L. Jacob and A. Kumar, "Saturated throughput analysis of an input queueing ATM switch with multiclass bursty traffic," *IEEE Trans. Commun.,* vol. 43, pp. 757–761, Apr. 1995.
[24] X. R. Cao and D. Towsley, "A performance model for ATM switches with general packet length distributions," *IEEE/ACM Trans. Networking,* vol. 3, pp. 299–309, June 1995.
[25] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM '96,* 1996, vol. 1, pp. 296–302.

Ge Nong (S'97) received the B.E. degree from Nanling Aeronautical Institute, JiangShu, China, in 1992 and the M.E. degree from South China University of Science and Technology, GuangDong, China, in 1995, all in computer engineering. He is currently working toward the Ph.D. degree in the Department of Computer Science, The Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong.

His current research interests include performance modeling of ATM switches, architectures of high-speed packet switches, and providing QoS guarantees on high-speed packet switching networks.



Jogesh K. Muppala (S'86–M'87) received the Ph.D. degree in electrical engineering from Duke University, Durham, NC, in 1991.

He is currently an Associate Professor in the Department of Computer Science, The Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong. He was previously with Software Productivity Consortium, Herndon, VA. His research interests include performance and dependability modeling, high speed networking, distributed systems, and stochastic Petri nets. He has published more than 30 refereed journal and conference papers in these areas. He is the program cochair for the 1999 Pacific Rim International Symposium on Dependable Computing. He has also served on program committees of international conferences.

Dr. Muppala was recently given the Teaching Excellence Appreciation Award by the Dean of Engineering at HKUST.

Mounir Hamdi (S'89–M'91) received the BS.c. degree with distinction in electrical engineering (computer engineering) from the University of Southwestern Louisiana, in 1985, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Pittsburgh in 1987 and 1991, respectively.

While at the University of Pittsburgh, he was a Research Fellow involved with various research projects on interconnection networks, high-speed communication, parallel algorithms, switching theory, and computer vision. In 1991, he joined the Computer Science Department at The Hong Kong University of Science and Technology as an Assistant Professor. He is now an Associate Professor of Computer Science and the Director of the Computer Engineering Programme. In 1999, he has been named one of the Ten Best Lecturers by the University's students. His main areas of research are ATM packet switching architectures, high-speed networks, wireless networking, and parallel computing. He has published more than 80 papers on these areas in various journals and conference proceedings. He cofounded and cochairs the International Workshop on High-Speed Network Computing.

Dr. Hamdi is on the editorial board of the *IEEE Communications Magazine* and *Parallel Computing,* and has been on the Program Committee of various International Conferences. He received the Best Paper Award at the 12th International Conference on Information Networking.