Lookahead Scheduling Algorithm for Input-Buffered Packet Switches

Kwan L. Yeung N. H. Liu Hai Shi

Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Hong Kong
{kyeung,nhliu}@ee.cityu.edu.hk

Abstract

An analytical model for evaluating the performance of a packet scheduling algorithm, called lookahead scheduling, is proposed in this paper. Using lookahead scheduling, each input port of a switch has B packet buffers. A packet arrives at an input port is scheduled for conflict-free transmission for up to B time slots in advance. If it cannot be scheduled for transmission in the next B slots, the packet is immediately discarded for having more room for the packets arrived later on. Based on a set of recursive equations for obtaining buffer occupancy and probability that a packet cannot be placed into a buffer, analytical expressions for switch throughput, packet loss probability and mean packet delay are derived. Analytical results are then compared with the simulation results and good agreement is found.

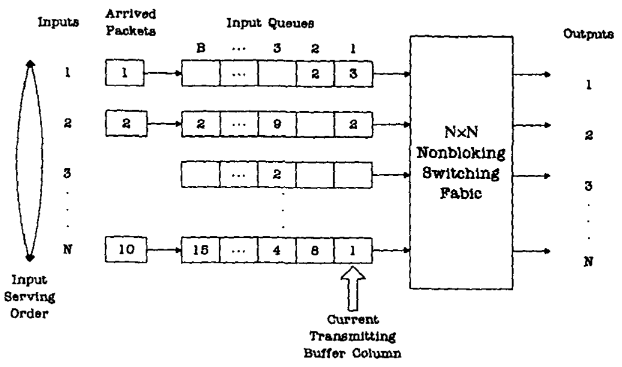


Fig. 1. An $N\times N$ input-buffered nonblocking switch with B buffers for each queue at the beginning of a time slot. The number inside a buffer denotes the destination of the packet it stores.

I. Introduction

Asynchronous Transfer Mode (ATM) is an international networking standard designed for cost-effective transfer of multimedia traffic, such as video-on-demand and video conferencing. Various ATM switch architectures have been proposed and studied extensively in order to provide high performance packet switching for integrated ATM transport. In this paper we focus on input-buffered nonblocking switches with N input ports and N output ports.

It has been found [1] that the maximum throughput of an input-buffered packet switch is limited to 58.6% under uniformly distributed traffic condition. This is because of the Head-of-Line (HOL) blocking phenomenon. To solve this problem, many queueing and scheduling techniques have been proposed aiming at maximizing switch throughput and minimizing mean packet delay[2]-[6]. In this paper, we focus on a scheduling algorithm [7] for switches with a single buffer queue per input port. We call it the lookahead scheduling algorithm. The throughput of the lookahead scheduling is found to be comparable to that of using the SDR algorithm [8] by simulations [7]. SDR is a maximum cardinality algorithm with extremely high time complexity of $O(N^4)$. As far as we know, SDR offers the best throughput and delay performance among various scheduling algorithms [6], [8], [9].

This work was supported by Research Grant Council Earmarked Grant 9040264.

In the lookahead scheduling algorithm, each input port has a single queue with B buffers and each buffer can accommodate one packet (as shown in Fig. 1). Upon the arrival of a packet, it is immediately scheduled for conflict-free transmission for up to B time slots in advance. If the scheduling effort fails, the packet is immediately discarded (instead of storing in the buffer) for having more room for packets arrived later on. It is shown [7] that the computational complexity of the lookahead scheduling algorithm is lower than that of the SDR [8], MRS [6] and RS [9] algorithms. However, the packet loss performance of the lookahead scheduling has not been considered [7]. Unlike conventional scheduling algorithms [6], [8], [9], the switch using lookahead scheduling has fixed buffer size at each input port. Therefore packets will be lost due to buffer overflow even under light input traffic loading.

In this paper, an analytical model for performance evaluation of the lookahead scheduling algorithm is constructed. Analytical expressions for the switch throughput, packet loss probability and mean packet delay are derived. In the next section, the lookahead algorithm is summarized. In Section III, a pipeline implementation of the lookahead scheduling is described. In Section IV, the analytical model for evaluating the switch performance is constructed. Then the analytical results are compared with the simulation results in Section V. Finally the paper is concluded in Section VI.

II. Lookahead Scheduling Algorithm

A. Data Structures

Fig. 1 shows an $N \times N$ input-buffered nonblocking switch with B buffers (numbered from 1 to B) at each input port. The following data structures are used in the lookahead scheduling algorithm:

1. *Buffer State Matrix S:* $S = [s_{ij}]_{N\times B}$ denotes the occupancy state of the j-th buffer at input i, or buffer (i,j), where

$$s_{ij} = \begin{cases} 0 & \text{if buffer } (i,j) \text{ is empty;} \\ 1 & \text{if buffer } (i,j) \text{ is occupied.} \end{cases}$$

2. *Destination Map M:* $M = [m_{ij}]_{N\times B}$ is the packet destination map, where

$$m_{ij} = \begin{cases} 0 & \text{if no packet in buffer column } j \\ & \text{destined for output } i; \\ 1 & \text{if there is a packet stored in} \\ & \text{column } j \text{ destined for output } i. \end{cases}$$

   A buffer *column* is referred to the buffers at the same relative position within each queue across all the input ports.

B. Packet Transmission and Packet Scheduling

In each time slot, the lookahead scheduling algorithm consists of two steps, packet transmission and packet scheduling. These two steps are summarized by the following pesudo-codes.

Lookahead Scheduling Algorithm

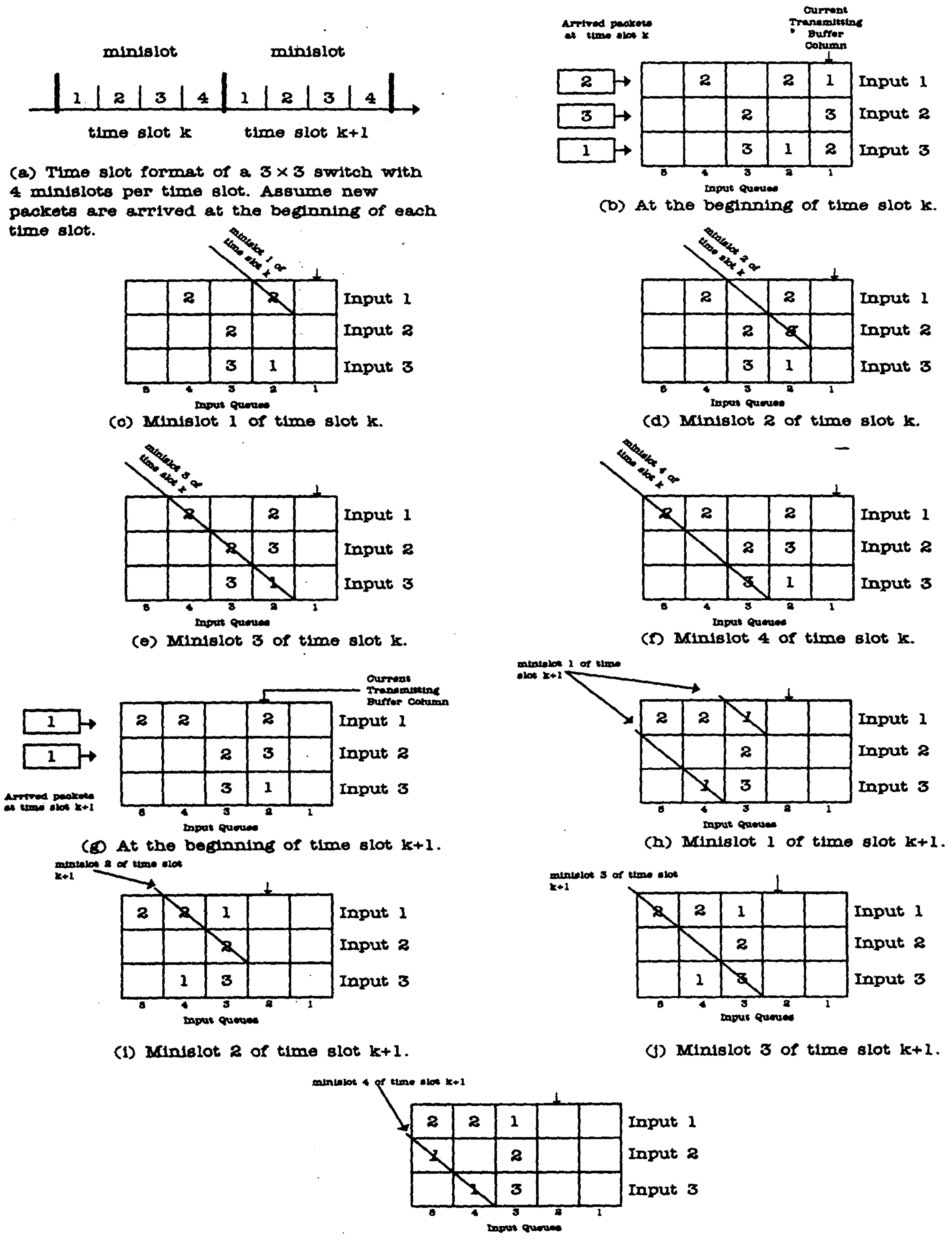
```
/*Packet Transmission*/
Step 1.  For $i = 1$ to $N$ do
             If $s_{ij} = 1$, send the packet and clear
             buffer $(i,j)$;
             $s_{ij} = 0$; $m_{ij} = 0$.    /* Reset buffer state
             matrix and destination map of the
                 transmitting column. */

/*Packet Scheduling*/
Step 2.  For $k = j+1, j+2, \cdots, B, 1, 2, \cdots, j$ do
             If $s_{ik} = 0$ and $m_{nk} = 0$,
             /* Buffer $(i,k)$ is available */
                 Place the packet into buffer $(i,k)$;
                 $m_{nk} = 1$; $s_{ik} = 1$; Exit.
             The packet is dropped and Quit.
```

Packet transmission takes place column-wise in a cyclic fashion. If column j is the transmitting column in current time slot, then column ($j+1$ mod B) becomes the transmitting column in the next time slot. For packet scheduling, the key is to make sure the packets stored in the same buffer column must destine to *different* output ports. This task is carried out by each input port processor. Let the current transmitting column be j. To schedule a packet arrived at input i with destination n, the input port processor searches a buffer for packet placement and the searching follows a column sequence of $\{j+1, j+2, \cdots, B, 1, 2, \cdots, j\}$. In the worst case, the last column to be searched is the transmitting column of the current slot. (We assume that all buffers in the transmitting column are available for storing packets at the end of a searching pass.)

The packets arrived at the same input port may not be served on a FCFS (first-come-first-serve) basis. But it will not cause the well-known packet out-of-sequence problem for each end-to-end connection. This is because for packets destined to the same output port (thus they belong to the same end-to-end connection) are served on a FCFS basis. Therefore the packet integrity is maintained. To maintain input port access fairness, the first port to be served in each time slot is cyclically rotated. That is if in the current slot we start to serve from input port i, in the next slot we start from input port ($i+1$ mod N).

III. Pipeline Implementation

Each input port has an input port processor to carry out the buffer checking function. Each input port processor should have a speed of at least $N+1$ buffer checkings per time slot. This ensures that the buffers which are checked in the same minislot must be at different buffer columns. Otherwise packet output conflict may occur. Without loss of generality, let each time slot be divided into $N+1$ minislots and a buffer checking can be completed within a minislot. An example of a time slot consists of 4 minislots is shown in Figure 2(a). At the beginning of a minislot, each input port processor checks a particular buffer for possible packet placement if there is a waiting packet.


minislot
minislot
1 2 3 4 1 2 3 4
time slot k
time slot k+1
Arrived packets at time slot k
Current Transmitting Buffer Column
Input 1
Input 2
Input 3
Input Queues
minislot 1 of time slot k
minislot 2 of time slot k
minislot 3 of time slot k
minislot 4 of time slot k
Arrived packets at time slot k+1
minislot 1 of time slot k+1
minislot 2 of time slot k+1
minislot 3 of time slot k+1
minislot 4 of time slot k+1

(a) Time slot format of a 3 × 3 switch with 4 minislots per time slot. Assume new packets are arrived at the beginning of each time slot.

(b) At the beginning of time slot k.

(c) Minislot 1 of time slot k.

(d) Minislot 2 of time slot k.

(e) Minislot 3 of time slot k.

(f) Minislot 4 of time slot k.

(g) At the beginning of time slot k+1.

(h) Minislot 1 of time slot k+1.

(i) Minislot 2 of time slot k+1.

(j) Minislot 3 of time slot k+1.

(k) Minislot 4 of time slot k+1.

Fig. 2. A 3 × 3 switch with buffer size $B = 5$. Let each time slot be divided into 4 minislots and each minislot is long enough for a packet placement/checking. New packets arrive at the beginning of each time slot and the packet placement order is fixed to be (input 1, input 2, input 3).

An example is shown in Figures 2(b)-(k) for a 3×3 switch. The current transmitting buffer column which marked by an arrow is column 1 in time slot k. At the beginning of time slot k, all packets in the transmitting column are switched to their respective outputs. Let the order for packet scheduling be (input 1, input 2, input 3). In the first minislot, the packet at input 1 is checked for placement at buffer (1,2) as indicated by the tilt line in Figure 2(c). Since this buffer is occupied, it is passed to buffer (1,3) at the end of minislot 1. In minislot 2 (Figure 2(d)), packet at input 1 is checked for placement at buffer (1,3) and packet at input 2 is checked for placement at buffer (2,2) simultaneously (i.e. pipeline operation). Packet at input 2 with output 3 is successfully placed (as indicated by the italic number). In minislot 3 (Figure 2(e)), packet at input 1 is checked for placement at buffer (1,4) and packet at input 3 is checked for placement at buffer (3,2). Both requests are rejected because the two associated buffers are occupied. Then in minislot 4 (Figure 2(f)), packet at input 1 is successfully placed into buffer (1,5).

Let the same order of (input 1, input 2, input 3) be followed for packet scheduling in time slot $k + 1$. At the beginning of time slot $k + 1$ (Figure 2(g)), two new packets arrive at inputs 1 and 2 and the transmitting column is rotated to column 2. The packets in column 2 are switched to their respective outputs. The packet that has not been placed in time slot k and those arrived in time slot $k+1$ are then scheduled. The packet arrived at input 1 with output 1 is placed into buffer (1,3) in minislot 1. Meanwhile the only packet that has not been placed in time slot k, i.e. the packet at input 3 with output 1, is placed into buffer (3,4) in the same minislot. Finally, the packet arrived at input 2 with output 1 is placed into buffer (2,5) in minislot 4 as shown in Figure 2(k).

For input port access fairness, the input port serving order for different time slots should be shifted. We can show that the pipeline operation is still valid. From the above example, we can see that with the proper synchronization among input port processors, a pipeline operation of packet placement can be carried out. The scheduling speed of this pipeline implementation is independent of the buffer size (B) at each input port. That means the scheduling complexity will not scale up with the buffer size if the input port processor can perform at least $N + 1$ packet placements in each time slot.

IV. Analytical Model

Assume the packets arrived at each input in each time slot follow an independent Bernoulli process with probability λ that a packet arrives. Let the destination of a packet be uniformly distributed to all outputs. An exact analysis of the lookahead scheduling algorithm is intractable because of the large state random variables involved. An approximate model which assumes a fixed serving priority is adopted in this paper. Under the fixed serving priority assumption, input i is always served before input j if $i < j$. For a homogeneous traffic system as we considered here, the packet scheduling priority has no bearing on the switch throughput. Without loss of generality, let input 1 always have the highest scheduling priority. For simplicity, let the transmitting column be always represented by column 1[1]. Let f_{ij} be the occupancy of buffer (i,j) and F_j be the *average* buffer occupancy of all column j buffers. We have

$$F_j = \frac{\sum_{i=1}^{N} f_{ij}}{N}. \tag{1}$$

When $j = 1$, F_1 is the switch throughput. F_j at the beginning of the next time slot is equal to F_{j+1} in the current time slot plus the contributions of the packets placed into column $j+1$ in the current slot. During each time slot, packets arrived at different inputs are served one by one. Let f'_{ij} be the occupancy of buffer (i,j) immediately *after* being considered for possible packet placement. Let F_{ij} be the average buffer occupancy in column j *before* buffer (i,j) being considered for possible packet placement. Thus $F_j = F_{1j}$ in the same time slot.

Let b_{ij} be the probability that a packet cannot be placed into buffer (i,j), we have

$$\begin{aligned} f'_{ij} &= f_{ij} + \lambda \prod_{k=2}^{j-1} b_{ik}(1 - b_{ij}) \\ F_{i+1,j} &= F_{ij} + \lambda \prod_{k=2}^{j-1} b_{ik}(1 - b_{ij})/N, \end{aligned} \tag{2}$$

where $i = 1, 2, \cdots, N$ and $j = 2, 3, \cdots, B$. Note that we have assumed that column 1 is always the transmitting column. In other words, f_{ij} in the next time slot is given by $f_{ij} = f'_{i,j+1}$.

A packet cannot be placed into a buffer if (i) that buffer is occupied, or (ii) the buffer is not occupied but there is another packet in the same buffer column destines to the same output. When the second situation occurs, the destination of the new packet depends on the outputs of the packets in the current buffer column. It is important to model this dependency of packet output addresses among different buffer columns. Two quantities are defined for this purpose:

[1]One can think that at the beginning of each time slot, we relabel column $j + 1$ to column j for $j < B$ and column 1 to column B.

- $F_j^{(k)}$: the average buffer occupancy in column j for storing packets with the same output addresses as those in column k, where $k < j$. Similarly, we can define $F_{ij}^{(k)}$ as the average buffer occupancy in column j for storing packets with the same output addresses as those in column k *before* buffer (i,j) being considered for packet placement.

- $b_{ij}^{(j)}$: the probability that a packet being considered for placement at buffer (i,j) has an output conflict with a stored packet in column j (independent of if buffer (i,j) is empty or not).

Therefore, we can have

$$\begin{aligned}
b_{i2} &= f_{i2} + (1-f_{i2})F_{i2} \\
b_{i3} &= \frac{f_{i2}}{b_{i2}}\left[f_{i3} + (1-f_{i3})F_{i3}\right] \\
&\quad + \frac{(1-f_{i2})F_2}{b_{i2}}\left[f_{i3} + (1-f_{i3})\frac{F_{i3}^{(2)}}{F_2}\right] \\
&= f_{i3} + \frac{(1-f_{i3})\left[f_{i2}F_{i3} + (1-f_{i2})F_{i3}^{(2)}\right]}{b_{i2}} \\
b_{i4} &= \frac{f_{i2}f_{i3}}{b_{i2}b_{i3}}\left[f_{i4} + (1-f_{i4})F_{i4}\right] \\
&\quad + \frac{f_{i2}(1-f_{i3})F_3}{b_{i2}b_{i3}}\left[f_{i4} + (1-f_{i4})\frac{F_{i4}^{(3)}}{F_3}\right] \\
&\quad + \frac{(1-f_{i2})f_{i3}F_2}{b_{i2}b_{i3}}\left[f_{i4} + (1-f_{i4})\frac{F_{i4}^{(2)}}{F_2}\right] \\
&\quad + \frac{(1-f_{i2})(1-f_{i3})F_3^{(2)}}{b_{i2}b_{i3}} \\
&\quad \cdot \left[f_{i4} + (1-f_{i4})\frac{F_{i4}^{(3)(2)}}{F_3^{(2)}}\right]
\end{aligned}$$

where $F_{i4}^{(3)(2)}$ is the average buffer occupancy in column 4 for storing packets with the same output addresses as those in columns 3 and 2, and before the packet (if available) arrived at input i is considered for placement.

Let us focus on several typical cases to explain the above set of equations. Consider b_{i2} the probability that a packet cannot enter buffer $(i,2)$. The first term on the right hand side of the equation is the probability that buffer $(i,2)$ is occupied. The second term is the probability that the packet will cause an output contention with one of the existing packet in column 2 given that buffer $(i,2)$ is empty.

Consider b_{i3}. A packet cannot be placed into buffer $(i,3)$ under two situations. First, buffer $(i,2)$ is occupied. Second, buffer $(i,2)$ is empty but the packet has an output contention with another packet in column 2. In case 1 (with probability f_{i2}/b_{i2}), the packet's output address is uniformly distributed to *all* N outputs. The probability that the packet will have an output contention with a packet in column 3 is F_{i3}. In case 2 (with probability $\frac{(1-f_{i2})F_2}{b_{i2}}$), the packet's output address is uniformly distributed to the set of output addresses occupied by packets in column 2. Thus the probability that the packet will have an output contention with some packet in column 3 is $F_{i3}^{(2)}/F_2$.

The expressions for b_{ij} where $j \geq 4$ can be similarly defined but the complexity involved in solving them is very high. To simplify the analysis, we substitute F_{i4} and $F_{i4}^{(3)}$ for $F_{i4}^{(2)}$ and $F_{i4}^{(3)(2)}$ respectively in b_{i4}. After rearranging, we get

$$b_{i4} \approx f_{i4} + \frac{(1-f_{i4})\left[f_{i3}F_{i4} + (1-f_{i3})F_{i4}^{(3)}\right]}{b_{i2}b_{i3}}.$$

Following this approximation method, we can have the following general expression for b_{ij}

$$b_{ij} \approx f_{ij} + \frac{(1-f_{ij})\left[f_{i,j-1}F_{ij} + (1-f_{i,j-1})F_{ij}^{(j-1)}\right]}{\prod_{k=2}^{j-1} b_{ik}}, \quad (3)$$

where $i = 1,2,\cdots,N$, $j = 2,3,\cdots,B,B+1$ and $f_{i,B+1} = 0$ for all i's. Notice that column $B+1$ represents the transmitting column (i.e. column 1) after packet transmission.

If a packet cannot be placed into buffer $(i,B+1)$, i.e. buffer $(i,1)$, it will be discarded[2].

By the definition of b_{ij}^j and making use of Eqn. (3), we have

$$b_{ij}^j \approx f_{ij}F_j + \frac{(1-f_{ij})\left[f_{i,j-1}F_{ij} + (1-f_{i,j-1})F_{ij}^{(j-1)}\right]}{\prod_{k=2}^{j-1} b_{ik}^k}, \quad (4)$$

From Eqns. (3) and (4), we have the following recursive equations for $F_{ij}^{(j-1)}$:

$$F_{ij}^{(j-1)} = F_{i-1,j}^{(j-1)} + \lambda \prod_{k=2}^{j-1} b_{i-1,k}^k (1-b_{i-1,j})/N, \quad (5)$$

where $i = 1,2,\cdots,N$ and $j = 2,3,\cdots,B$. Then the mean packet delay is given by

$$D = \sum_{j=1}^{B} j(F_j - F_{j+1})/F_1, \quad (6)$$

[2]From our assumption that input 1 is always served with the highest priority, buffer (1,1) will always be empty and $b_{11} = 0$. In other words, packets arrived at input 1 will never be discarded. This however will not affect our subsequent derivations as we are only interested in the average switch performance, not a particular port.

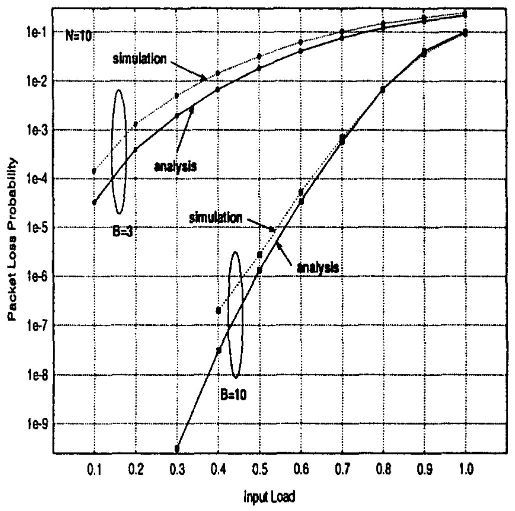


Fig. 3. Packet loss probability vs input load λ for a 10×10 with buffer sizes $B = 3$ and 10.

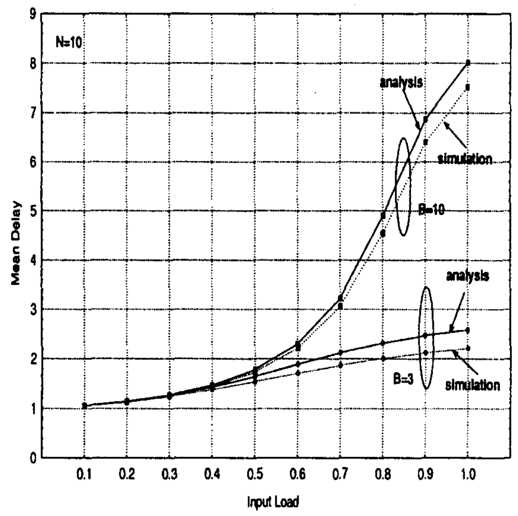


Fig. 4. Mean packet delay vs input load λ for a 10×10 switch with buffer sizes $B = 3$ and 10.

where $(F_j - F_{j+1})/F_1$ is the probability that a packet needs to wait for j time slots. Using the lookahead scheduling, a packet can be delayed for at most B slots. The packet loss probability of the switch is given by

$$P_{loss} = 1 - F_1/\lambda. \tag{7}$$

The above set of equations can then be solved iteratively with an initial set of values, say, $f_{ij} = 0, F_{ij} = 0, F_j = 0, F_{ij}^{(k)} = 0$ and $F_j^{(k)} = 0$ for all i and j's.

V. Performance Evaluations

The packet loss probability, switch throughput and mean packet delay are studied by both simulations and analysis in this section. We focus on a 10×10 switch with buffer sizes $B = 3$ and 10 respectively. Figs. 3 shows the packet loss probability P_{loss} against input load λ. When input load is light, we can see that the analytical model only slightly underestimates the packet loss performance of the switch. At $\lambda = 0.6$ and $B = 10$, $P_{loss} = 5.22 \times 10^{-5}$ from simulation and 3.41×10^{-5} from analysis. At the same load with $B = 3$, $P_{loss} = 6.4 \times 10^{-2}$ from simulation and 4.07×10^{-2} from analysis.

Fig. 4 shows the mean packet delay versus input load. Both simulation results and analytical results match very well. At $\lambda = 0.8$, mean packet delay is 4.55 time slots from simulation and 4.91 slots from analysis.

VI. Conclusions

In this paper, an analytical model was proposed for the performance evaluation of an efficient packet scheduling algorithm, called lookahead scheduling algorithm. Analytical expressions for packet loss probability, throughput and mean packet delay were derived. Analytical results were compared with the simulation results and we found that the proposed model is very accurate in predicting the performance of the lookahead scheduling algorithm.

REFERENCES

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

[2] M. Chen, N. Georganas, and O. Yang, "A fast algorithm for multi-channel/port traffic assignment," *Proceedings of IEEE ICC '94*, pp. 96-100 1994.

[3] J. Choi and C. Un, "A nonblocking ATM switch with a single plane or multiple planes combined with a window scheme," *Proc. of IEEE ICC' 96*, pp. 1680-1684, Nov. 1996 1996.

[4] M. Mehmet-Ali, M. Youssefi, and H. Nguyen, "The performance analysis and implementation of an input access scheme in a high-speed packet switch," *IEEE Trans. on Commun.*, vol. 42.

[5] Y. Leung, "Neural scheduling algorithms for time-multiplex switches," *IEEE J. Select. Areas Commun.*, vol. 12, No. 9, pp. 1481-1487, Dec. 1994.

[6] M. Chen and T. S. Yum, "A conflict-free protocol for optical WDMA networks," *Proceedings of IEEE GLOBECOM '91*, pp. 1276-1281 1991.

[7] V. Yau and K. Pawlikowski, "A conflict-free traffic assignment algorithm using forward planning," *Proceedings of IEEE INFOCOM '96*, pp. 1277-1284 1996.

[8] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. on Commun.*, Vol. COM-27, No. 10, pp. 1449-1455, Oct. 1979.

[9] R. Chipalkatti, Z. Zhang, and A. S. Acampora, "High speed communication protocols for optical star coupler using WDM," *Proceedings of IEEE INFOCOM '92*, pp. 2124-2133 1992.