Scheduling Algorithms for Input-queued Switches with Virtual Output Queueing

N. H. Liu* Kwan L. Yeung† Derek C. W. Pao‡

Abstract

A set of packet scheduling algorithms are proposed for improving the performance of Iterative Longest Port First (iLPF) algorithm in [1] for Virtual Output Queueing Switch. In our proposed algorithms, scheduling priority is given according to different criteria that include input port occupancy, output port occupancy and critical port in VOQ. One of our proposed algorithm, called Longest Input Port First with Throughput Maximization (LIPF with TM), gives significant performance improvement in mean packet delay and throughput when compared with iLPF. We found that for a 16×16 switch with input load $p = 0.85$, the mean packet delay is 7.08 slots for iLPF and 3.21 slots for LIPF with TM. This represents a 55% cut in mean packet delay.

I. Introduction

With the explosive growth of the Internet, it is no doubt that switches and routers play an increasingly important role in data communications. Traditionally, switch and routers employ output-queueing where each input can transfer up to the maximum of N (where N is switch size) packets to an output port in each time slot. Such output-queued switches face the scalability problem. For an $N \times N$ switch, the switch fabric must operate at N times of the input link speed. As link speed increases to the Gbit/s range and as the switches have a larger number of input ports, the required fabric speed becomes infeasible unless very expensive technologies are used.

The input-queued switches, on the other hand, can overcome the scalability problem. Since buffers are placed at the input side of the switches, the incoming packets are first stored in buffers at the input, the switch fabric would transfer some of them to the output and the blocked packets are queued for further transmission in the subsequent time slots. In input-queued switches, the switch fabric can run at the same speed as each input link. Owing to its scalability, input-queueing has always been an attractive alternative for high speed switching systems. It has been found [2] that the maximum throughput of the input-queued switch with a single queue per input port is limited to 58.6% under uniformly distributed traffic condition. This is because of the Head-of-Line (HOL) blocking phenomenon. That is a packet can be held up by another packet ahead of it in the same queue and is destined to a different output. In [3, 4, 5, 6, 7], Virtual Output Queueing (VOQ) switch where each input port maintains a separate queue for each output is proposed, as shown in Fig. 1. In this case, HOL blocking is eliminated.

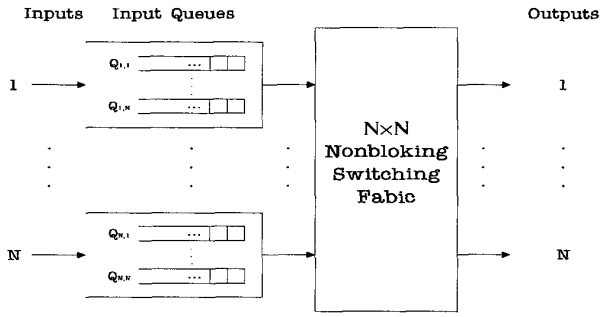

Fig. 1.: An $N \times N$ input-queued switch with Virtual Output Queueing (VOQ)

Maximizing throughput in VOQ switch is equivalent to the matching problem in a bipartite graph. Among various proposed maximum-size bipartite matching algorithms, the most efficient one converges in $O(N^{5/2})$ time [8]. However, because of the backtracking nature, such algorithms are not suitable for hardware implementation. For practical high-performance switching systems, an iterative algorithm called iSLIP is proposed in [9, 10]. It has been shown that it can achieve asymptotically 100% throughput for uniform traffic. But the iSLIP algorithm cannot achieve 100% throughput when the input traffic is non-uniform. In [11, 12], the Oldest Cell First (OCF) and Longest Queue First (LQF) algorithms, which can achieve 100% throughput for non-uniform traffic, are then introduced. Since both OCF and LQF are difficult to be implemented in hardware at high speed, an iterative algorithm called Iterative Longest Port First (iLPF) is proposed in [1] by the same authors. iLPF is a

* The author is with the Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Hong Kong. (e-mail: nhliu@ee.cityu.edu.hk)

† The author is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong. (e-mail: kyeung@eee.hku.hk)

‡ The author is with the Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Hong Kong. (e-mail: eedpao@cityu.edu.hk)

0-7803-7097-1/01/$10.00 ©2001 IEEE

practical approximation to the Longest Port First (LPF) algorithm and can be adapted to run at high speed. To the best of our knowledge, the performance of iLPF is not well-addressed in [1].

In this paper, we propose and study a set of new scheduling algorithms for input-queued switches with virtual output queueing. Like iLPF, they all can be efficiently implemented in hardware. They differ with each other in giving scheduling priority according to different criteria, which include input port occupancy, output port occupancy/popularity and critical port. We leave the detailed descriptions of those algorithms to the next section. We compare the performance of those new algorithms with iLPF in Section III. We show that one of our proposed algorithms, called LIPF with TM, gives significant performance improvement in mean packet delay and throughput performance. Finally, we conclude the paper in Section IV.

II. Scheduling Algorithms for VOQ Switch

A. Data Structure

Consider an $N \times N$ VOQ switch as shown in Fig. 1. Let packets waited at all inputs at the beginning of time slot k be represented by a traffic matrix $D(k) = [d_{ij}]$, where d_{ij} is the number of packets that are currently waiting to be transmitted from input i to output j. Let new packets arrived at the beginning of slot k be represented by an arrival matrix $A(k) = [a_{ij}]$, where $a_{ij} = 1$ if a packet arrives from input i destined to output j in the current slot. Further let the packets to be transmitted in slot k be represented by a transmission matrix $T(k) = [t_{ij}]$, where $t_{ij} = 1$ if a packet is scheduled to transmit from input i to output j; otherwise t_{ij} =0. In each time slot, each input can send at most one packet and each output can receive at most one packet. For a conflict-free transmission, each row sum and each column sum of $T(k)$ can have at most one 1. At the start of time slot k, traffic matrix is updated as follows: $D(k) = D(k-1) - T(k-1) + A(k)$.

B. Terminology

1. *Input (Port) Occupancy $R_i(k)$:* $R_i(k)$ is the total number of packets that are currently waiting at input i for transmission to their respective outputs at time slot k. In other words, $R_i(k)$ is the row sum of row i in traffic matrix $D(k)$, i.e. $R_i(k) = \sum_{j=1}^{N} d_{ij}$. In the rest of the paper, row sum and input port occupancy will be used interchangeably.
2. *Output (Port) Occupancy $C_j(k)$:* $C_j(k)$ is the total number of packets that are destined to output j but are currently waiting at some input ports at time slot k. In fact, it is an indicator for output port popularity. From traffic matrix $D(k)$, $C_j(k) = \sum_{i=1}^{N} d_{ij}$. In the rest of the paper, column sum and output port occupancy will be used interchangeably.
3. *Critical Port:* A critical port is the port (either an input port or an output port) that has the maximum occupancy, i.e. the port with the maximum row sum or column sum.
4. *Throughout Maximization Procedure:* Throughput maximization refers to the process of finding a port (either input or output port) which has the minimum number of non-zero entries in the current traffic matrix $D(k)$. Then scheduling/transmission priority is given to packets at that port. The idea is that, in general, the port that has the minimum number of non-zero entries has a smaller chance of being able to have a packet scheduled for transmission. If we can schedule such packets first, we then have the potential of packing more packets to be transmitted in the same time slot/transmission matrix. As a result, throughput maximization can be achieved.

$$D(k) = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix} \begin{matrix} 4 \\ 5 \\ 7 \\ 3 \end{matrix}$$
$$\begin{matrix} 5 & 4 & 4 & 6 \end{matrix}$$

Fig. 2.: A 4×4 traffic matrix $D(k)$ with input and output occupancy shown.

Consider a traffic matrix $D(k)$ of a 4×4 VOQ switch in time slot k in Fig. 2. The row sums and column sums of the matrix are shown at the right hand side and the bottom of D respectively. From our definition above, we can see that row 3 is the critical port. For the same traffic matrix, the corresponding number of non-zero entries for each input and output port is shown in Fig. 3. In this case, row 4 (i.e. input 4) has the least transmission/scheduling feasibility because it only has packets destined to output 4.

C. Scheduling Algorithms

Based on the definitions in the previous section, we first review the Iterative Longest Port First (iLPF) algorithm proposed in [1]. Then three new scheduling algorithms are proposed.

$$D(k) = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix} \begin{matrix} 3 \\ 3 \\ 4 \\ 1 \end{matrix}$$
$$\begin{matrix} 3 & 2 & 3 & 3 \end{matrix}$$

Fig. 3.: A 4×4 traffic matrix with the number of non-zero entries for each port shown.

1) Iterative Longest Port First(iLPF) : Let $I(k) = \{1, 2, .., N\}$ and $O(k) = \{1, 2, .., N\}$ denote the set of all inputs and the set of all outputs at the beginning of time slot k. For scheduling the traffic at time slot k, iLPF can be summarized by the following pseudo-codes:

iLPF algorithm

1. Find $R_i(k)$ and $C_j(k)$ in traffic matrix $D(k)$;
2. $O(k) = \{1, 2, .., N\}$;
3. If $O(k) = \emptyset$ then Exit;
4. Find a column q from $O(k)$ such that $C_q(k) = max_{j \in O(k)}\{C_j(k)\}$; Remove q from $O(k)$;
5. If $C_q(k)$ equals to 0 then Exit;
6. $I(k) = \{1, 2, .., N\}$;
7. If $I(k) = \emptyset$ then go to Step 3;
8. In a column q, find a row p from $I(k)$ such that $R_p(k) = max_{i \in I(k)}\{R_i(k)\}$; Remove p from $I(k)$;
9. If $R_p(k)$ equals to 0, go to Step 3;
10. If $(d_{pq} \neq 0)$, set $t_{pq} = 1$ and set $R_p(k)$ and $C_q(k) = 0$ and go to Step 3;
11. go to Step 7;

In iLPF, the packet scheduling priority is given to the output port with the highest occupancy.

2) Longest Input Port First with Throughput Maximization (LIPF with TM): In algorithm LIPF with TM, we first find row p which has the maximum input occupancy from the traffic matrix and then we apply throughput maximization procedure in Section II.2 to find the corresponding column.

LIPF with TM algorithm

1. Find $R_i(k)$ and $C_j(k)$ in traffic matrix $D(k)$;
2. $I(k) = \{1, 2, .., N\}$;
3. If $I(k) = \emptyset$ then Exit;
4. Find a row p from $I(k)$ such that $R_p(k) = max_{i \in I(k)}\{R_i(k)\}$; Remove p from $I(k)$;
5. If $R_p(k)$ equals to 0 then Exit;
6. $O(k) = \{1, 2, .., N\}$;
7. If $O(k) = \emptyset$ then go to Step 3;
8. In row p, find a column q from $O(k)$ with minimum number of non-zero entries Remove q from $O(k)$;
9. If the minimum number of non-zero entries of a column q equals to 0 then go to Step 7;
10. If $(d_{pq} \neq 0)$, set $t_{pq} = 1$ and set $R_p(k) = 0$ and set all entries in row p and column q of $D(k)$ to 0 and go to Step 3;
11. go to Step 7;

3) Longest Output Port First with Throughput Maximization (LOPF with TM): In LOPF with TM algorithm, we first find column q which has the maximum output occupancy and then we apply throughput maximization procedure to find the corresponding row. In other words, we reverse the order of selecting a row first as that of LIPF with TM.

LOPF with TM algorithm

1. Find $R_i(k)$ and $C_j(k)$ in traffic matrix $D(k)$;
2. $O(k) = \{1, 2, .., N\}$;
3. If $O(k) = \emptyset$ then Exit;
4. Find a column q from $O(k)$ such that $C_q(k) = max_{j \in O(k)}\{C_j(k)\}$; Remove q from $O(k)$;
5. If $C_q(k)$ equals to 0 then Exit;
6. $I(k) = \{1, 2, .., N\}$;
7. If $I(k) = \emptyset$ then go to Step 3;
8. In column q, find a row p from $I(k)$ with minimum number of non-zero entries Remove p from $I(k)$;
9. If the minimum number of non-zero entries of a row p equals to 0 then go to Step 7;
10. If $(d_{pq} \neq 0)$, set $t_{pq} = 1$ and set $R_p(k) = 0$ and set all entries in row p and column q of $D(k)$ to 0 and go to Step 3;
11. go to Step 7;

4) Critical Port First with Throughput Maximization (CPF with TM): In CPF with TM algorithm, we first find the critical port. If the critical port is a row, we apply throughput maximization procedure to find the corresponding column. Otherwise, we apply throughput maximization procedure to find the corresponding row. CPF with TM algorithm is summarized below.

CPF with TM algorithm

1. Find $R_i(k)$ and $C_j(k)$ in traffic matrix $D(k)$;
2. $I(k) = O(k) = \{1, 2, .., N\}$;
3. If $O(k)$ or $I(k) = \emptyset$ then Exit;
4. Find critical port in $I(k)$ and $O(k)$;
5. If the critical port is column q then go to Step 11

6. Remove p from $I(k)$; $h = 0$;
   If $R_p(k)$ equals to 0 then Exit;
7. In row p, find a column q from $O(k)$ with minimum number of non-zero entries;
8. If the minimum number of non-zero entries of a column q equals to 0 then go to Step 3;
9. $h = h + 1$; If $(d_{pq} \neq 0)$,
   set $t_{pq} = 1$ and set $R_p(k) = 0$
   and set all entries in row p and column q of $D(k)$ to 0, remove q from $O(k)$ and go to step 3;
10. If $h < N$ then go to Step 7;
    else go to Step 3;
11. Remove q from $O(k)$; $g = 0$;
    If $C_q(k)$ equals to 0 then Exit;
12. In column q, find a row p from $I(k)$ with minimum number of non-zero entries;
13. If the minimum number of non-zero entries of a row p equals to 0 then go to Step 3;
14. $g = g + 1$; If $(d_{pq} \neq 0)$,
    set $t_{pq} = 1$ and set $R_p(k) = 0$
    and set all entries in row p and column q of $D(k)$ to 0, remove p from $I(k)$ and go to step 3;
15. If $g < N$ then go to Step 12;
    else go to Step 11;

D. An Example

If we use iLPF algorithm to schedule the traffic matrix $D(k)$ in Fig. 2, the first packet scheduled for transmission is a packet from input 3 to output 4. And the next packet is from input 2 to output 1. The corresponding transmission matrix $T(k)$ for iLPF algorithm is

$$T(k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Refer to Figs. 2 and 3, if we use LIPF with TM algorithm to schedule this same traffic matrix, the first packet scheduled for transmission is a packet from input 3 to output 2. And the next packet is from input 2 to output 1. In this case, the corresponding transmission matrix $T(k)$ for LIPF with TM algorithm is

$$T(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If LOPF with TM algorithm is used, the first scheduled packet is from input 4 to output 4. And the next packet is from input 2 to output 1. The resulting transmission matrix T is

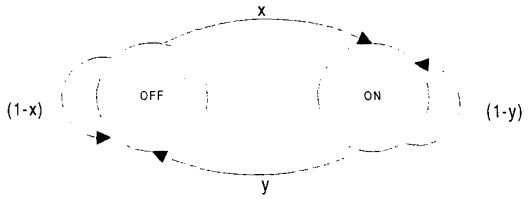

Fig. 4.: On-Off traffic model

$$T(k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If CPF with TM algorithm is used, the first scheduled packet is from input 3 to output 2. And the next packet is from input 4 to output 4. The resulting transmission matrix T is happened to be the same as that of LIPF with TM.

III. Performance Evaluations

In this section, the delay performance of the proposed algorithms is compared with iLPF by simulations. Two input traffic models are adopted. In the uniform traffic model, packets arriving at each input at each time slot follow the same independent Bernoulli process with probability p of having a new packet. We call p as the input load. In the bursty traffic model, the traffic at each input is modelled as an ON-OFF source in discrete domain as shown in Fig. 4. At ON state, the source generates packets. At OFF state, the source does not generate any packet. A state change occurs only at the end of a time slot. Packets of the same burst (i.e. packets arrived in consecutive time slots) will have the same destination. Each source is characterized by the peak packet rate (i.e. 1 packet per time slot), the average packet rate (i.e. the input load p) and the burst size s. Given these parameters, the state transition probabilities in Fig. 4 can be computed as $x = \frac{p}{s(1-p)}$ and $y = \frac{1}{s}$.

Fig. 5 shows the mean packet delay in log scale against the input load p for independent Bernoulli process. The lower bound for mean packet delay is obtained from an output-queued switch. From Fig. 5, we can see all the proposed algorithms achieve better performance than the iLPF algorithm. At input load $p = 0.85$, the mean packet delay is 7.08 slots for iLPF and 3.95 slots for LOPF with TM. A 44% cut in mean packet delay is achieved. For LIPF with TM, the mean packet delay is 3.21 slots. This gives a 55% cut in mean packet delay as compared to

0-7803-7097-1/01/$10.00 ©2001 IEEE

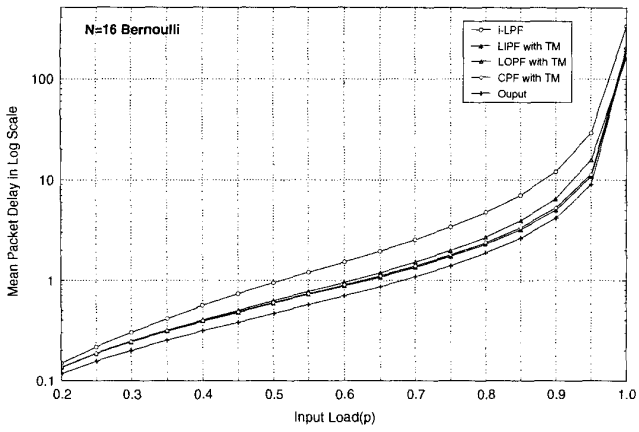

Fig. 5.: Mean packet delay vs input load p for a 16×16 switch using uniform traffic model.

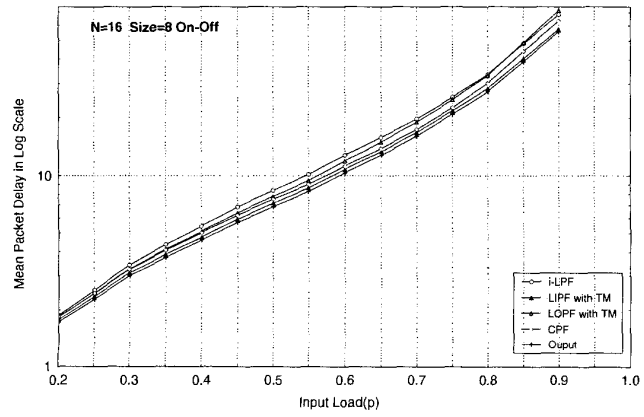

Fig. 6.: Mean packet delay vs input load p for a 16×16 switch using On-Off traffic model with burst size 8.

that of iLPF. The performance of CPF with TM and LIPF with TM algorithm are better than the LOPF with TM algorithm. Besides, both CPF with TM and LIPF with TM algorithms outperform the LOPF with TM. The performance of CPF with TM is comparable to that of LIPF with TM, and both are better than the LOPF with TM algorithm.

Fig. 6 shows the mean packet delay in log scale against the input load p using the bursty On-Off traffic model. The average burst size s is set to 8. The lower bound is again obtained from an output-queued switch. From Fig. 6, we can see all algorithms give a quite close performance. This is due to the relatively large burst size as compared with the switch size, whcih makes the throughput maximization procedure less effective. Nevertheless, at $p = 0.85$, the delay is 48.4 slots for iLPF and 40.5 slots for LIPF with TM. A 16.3% cut in mean packet delay is obtained.

IV. Conclusions

In this paper, three new packet scheduling algorithms were proposed for input-queued switch with Virtual Output Queueing. They all suitable for efficient high-speed hardware implementation. They differ with each other in giving packet scheduling priority by following different criteria including input port occupancy, output port occupancy and critical port. Combining with the heuristic procedure for throughput maximization, we have shown that the three proposed algorithms outperform the Iterative Longest Port (iLPF) algorithm. We found that one of our proposed algorithms, called Longest Input Port First with Throughput Maximization, always achieves the best delay performance and is very close to the performance lower bound.

References

[1] N. McKeown and A. Mekkittikul, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *Proceedings of IEEE INFOCOM '98*, 1998.

[2] 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.

[3] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans, on Computer Systems*, pp. 319-352 Nov 1993.

[4] M. Karol, K. Eng, and H. Obara, "Improving the performance of input-queued ATM packet switches," *Proceedings of IEEE INFOCOM '92*, pp. 110-115 1992.

[5] H. Obara, "Optimum architecture for input queueing ATM switches," *IEE Elect. Lett.*, pp. 555-557 28th March 1991.

[6] H. Obara, S. Okamoto, and Y. Hamazumi, "Input and output queueing ATM switch archecture with spatial and temporal slot reservation control," *IEE Elect. Lett.*, pp. 22-24 2nd Jan 1992.

[7] Y. Tamir and G. Frazier, "High performance multi-queue buffers for VLSI communication switches," *Proceedings of 15th Ann. symp. on Comp. Arch.*, pp. 343-354 June 1988.

[8] J. E. Hopcroft and R. M. Karp, "An algorithms for maximum matching in biparite graphics," *Soc. Ind. Math. J. Computation*, Vol. 2, pp. 225-231 1973.

[9] N. McKeown, "Scheduling algorithms for input-queued cell switches," *PhD. Thesis, University of California at Berkeley*, 1995.

[10] N. McKeown, "The iSLIP scheduling algorithm for input-queues switches," *IEEE Trans. on Networking*, Vol. 7, No. 2, pp. 188-201, April 1999.

[11] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Proceedings of IEEE INFOCOM '96*, pp. 296-302 1996.

[12] N. McKeown and A. Mekkittikul, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," *Proceedings of IEEE ICCCN '96*, 1996.