IEICE TRANS. COMMUN., VOL.E83–B, NO.2 FEBRUARY 2000

PAPER IEICE/IEEE Joint Special Issue on Recent Progress in ATM Technologies

The *i*-QOCF (Iterative Quasi-Oldest-Cell-First) Scheduling Algorithm for Input-Queued ATM Switches*

Masayoshi NABESHIMA† ***and*** **Naoaki YAMANAKA**†, ***Members***

SUMMARY This paper proposes the iterative quasi-oldest-cell-first (*i*-QOCF) scheduling algorithm, a new scheduling algorithm for input-queued ATM switches with virtual output queuing (VOQ). In the *i*-QOCF scheduling algorithm, each input port and each output port maintains its own list. The length of the list can be $N, 2\times N, \ldots, B\times N$, where B is the size of the separate queue for an output port at input ports, and N is the number of output ports. The list maintained by an input port contains the identifiers for those output ports to which that input port will send a cell. The list maintained by an output port contains the identifiers for input ports that have a cell destined for that output port. If we use a list whose length is $B\times N$, then the identifiers in the list appear in the oldest order, and *i*-QOCF gives preference to cells that have been waiting for the longest time. If we use a list whose length is less than $B\times N$, then the identifiers in the list appear in the quasi-oldest order, and *i*-QOCF gives preference to cells that have been waiting for the quasi-longest time. We determine the performance of *i*-QOCF in a comparison with *i*-OCF in terms of cell delay time. We find that an input-queued ATM switch with *i*-QOCF and VOQ can achieve 100% throughput for independent arrival processes. Under uniform traffic, 3-QOCF is enough to achieve convergence during one cell time. If we use 3-QOCF, the list length is $3\times N$, then its cell delay time is almost the same as that of 4-OCF (Oldest-Cell-First).

key words: *input-queued ATM switch, scheduling algorithm, throughput, cell delay time*

1. Introduction

The tremendous popularity of the World Wide Web has dramatically increased the amount of traffic carried over the Internet [1]. In addition, the demand for multimedia applications, such as video on demand, distance learning, and distributed games, is now expanding. If Asynchronous Transfer Mode (ATM) backbone networks are to support such a computer communication network, they will require switches that offer throughputs of more than 1 Tb/s in a cost-effective manner.

Many switch architectures have been proposed to realize a high-speed ATM switch [2]–[4]. They can be classified into three types according to the location of their buffers. They include the input-queued, output-queued, and shared-queued ATM switches [5]. Among these, the input-queued ATM switch has very attractive features, suggesting that it has potential as a high-speed ATM switch; it does not require any internal speedup, which is the ratio of the switch fabric speed to the input speed. The output-queued ATM switch, on the other hand, requires a speedup of N, where N is the number of input ports. This requirement needs high-speed memory access, and so increases memory cost significantly.

Offsetting this advantage, though, the input-queued ATM switch with a First-In-First-Out (FIFO) queue per input port has the disadvantage that its maximum throughput is limited to $(2-\sqrt{2}) = 58.6\%$ [6]. However, it has been shown that an appropriate buffering policy with a scheduling algorithm enable an input-queued ATM switch to achieve 100% throughput for independent arrival processes [7]. This buffering policy is called virtual output queuing (VOQ) [8]. Several scheduling algorithms for achieving 100% throughput have been proposed [9], including *i*-SLIP, *i*-LQF (Longest-Queue-First), and *i*-OCF (Oldest-Cell-First). *i*-SLIP is an unweighted matching algorithm, and both *i*-LQF and *i*-OCF are weighted matching algorithms. It is known that weighted matching algorithms have better performance than unweighted matching algorithms [10]. *i*-LQF gives preference to queues with higher occupancy, while *i*-OCF gives preference to cells that have been waiting for the longest time. *i*-LQF can lead to the starvation of one or more input queues, while *i*-OCF does not [9]. However, *i*-OCF requires timestamp information in each cell. Moreover, the input ports have to send information indicating the waiting time in *i*-OCF to the output ports.

In this paper, we propose a new scheduling algorithm for achieving 100% throughput in an input-queued ATM switch with VOQ [11]. We call it the iterative quasi-oldest-cell-first (*i*-QOCF) scheduling algorithm. *i*-QOCF has almost the same performance as *i*-OCF, but it does not require timestamp information, and the input ports do not have to send information about the waiting time to the output ports.

We show the performance of *i*-QOCF and present results in which we compare *i*-QOCF to *i*-OCF in terms of cell delay time.

The remainder of this paper is organized as follows. Section 2 briefly reviews the conventional input-queued ATM switch architecture, and improved strategies for input-queued ATM switches. Section 3 presents

Manuscript received June 1, 1999.
Manuscript revised August 26, 1999.

†The authors are with NTT Network Service Systems Laboratories, Musashino-shi, 180-8585 Japan.

*A preliminary version of this paper appeared in IEEE ATM'99 Workshop [11].

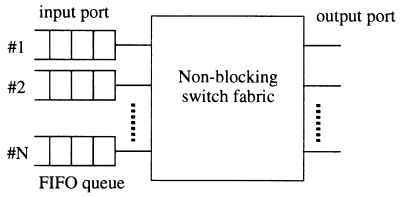

Fig. 1 $N \times N$ pure input-queued ATM switch.

the architecture and operation of the i-QOCF scheduling algorithm. Section 4 evaluates its performance using computer simulations. Finally, conclusions are provided in Sect. 5.

2. Input-Queued ATM Switch Architecture

In simple terms, an input-queued ATM switch consists of a non-blocking switch fabric of size $N \times N$ and queues that are placed at each input port of the switch fabric. Each input port can accept at most one cell during one cell time, and can deliver one cell to an appropriate output port in one cell time.

2.1 Pure Input-Queued ATM Switch

We call an input-queued ATM switch in which a single FIFO queue is placed at each input port, a pure input-queued ATM switch (Fig. 1). Each cell that arrives first enters a queue at the input port, and then when it reaches the head of the queue, it undergoes arbitration. The scheduler chooses one of the head of line (HOL) cells destined for the same destination output port.

The pure input-queued ATM switch suffers from HOL blocking [6]. In HOL blocking, if the HOL cell in a queue is not served because of contention at its destination output port, then other cells behind it cannot be served either, even if they are destined for different output ports that are not experiencing contention. This reduces the maximum switch throughput. Under the assumption of uniform traffic, the maximum throughput of a pure input-queued ATM switch is limited to 58.6%. With correlated input traffic, the throughput can be even lower.

2.2 Performance Improvement of Input-Queued ATM Switch

Various techniques for improving the low performance of the pure input-queued ATM switch have been proposed and investigated [13]–[15].

The throughput can be increased by adopting a *window policy* [16]. It reduces HOL blocking by checking the first w cells in each FIFO queue. That is, if an input port is not selected by the scheduler to transmit its HOL cell, it will try again with its second cell to access to any of the remaining idle output ports, and so on up to w cells. The throughput is improved from 58.6% ($w = 1$) to 70% with $w = 2$ or 88% with $w = 8$ under uniform traffic [12]. However, when the traffic is bursty, cells behind the HOL cell are likely to be destined for the same output port. Thus, this policy is highly sensitive to traffic arrival patterns.

HOL blocking can be eliminated completely by using the buffer policy known as VOQ. In VOQ, each input port maintains a separate queue for each output port, rather than maintaining a single FIFO queue for all cells. VOQ eliminates HOL blocking, because it ensures that a cell cannot be held up by a cell queued ahead of it that is destined for a different output port.

It has been shown that an input-queued ATM switch with VOQ can attain 100% throughput by means of sophisticated scheduling algorithms [7]. Several sophisticated scheduling algorithms have been proposed: i-SLIP, i-OCF, and i-LQF [9].

Among them, we mention i-OCF because it can prevent any queue from being starved. On the other hand, i-SLIP may became unstable under non-uniform traffic, and it is possible for i-LQF to permanently starve an input queue under inadmissible traffic [9].

i-OCF gives preference to cells that have been waiting for the longest time and is basically composed of three steps, as follows.

Step 1 Each unmatched input port sends a request word of width 2^b bits to each output port for which it has a queued cell, indicating the waiting time of the cell at the head of each unmatched input port.

Step 2 If an unmatched output port receives any requests, it chooses the request having the longest waiting time, and sends that input port an offer.

Step 3 If an unmatched input port receives one or more offers, it accepts the one to which it made the request with the longest waiting time.

The prefix i of i-OCF signifies that steps 1 through 3 are iterated i times. In other words, i-OCF tries i times to find a conflict-free match between input and output ports.

i-OCF obviously requires timestamp information, the overhead of b bits, in each cell. When a cell enters a queue, the time information must be written in the overhead of the cell.

3. Proposed Scheduling Algorithm

There are two important differences between our scheduling algorithm and i-OCF. First, it does not require timestamp information as i-OCF does. Second, the input ports do not have to send information indicating the waiting time to the output ports, as is needed in i-OCF.

In the i-QOCF scheduling algorithm, each input port and each output port maintains its own list (Fig. 2). The length of the list can be $N, 2 \times N, \ldots, B \times N$, where B is the size of the separate queue for an out-

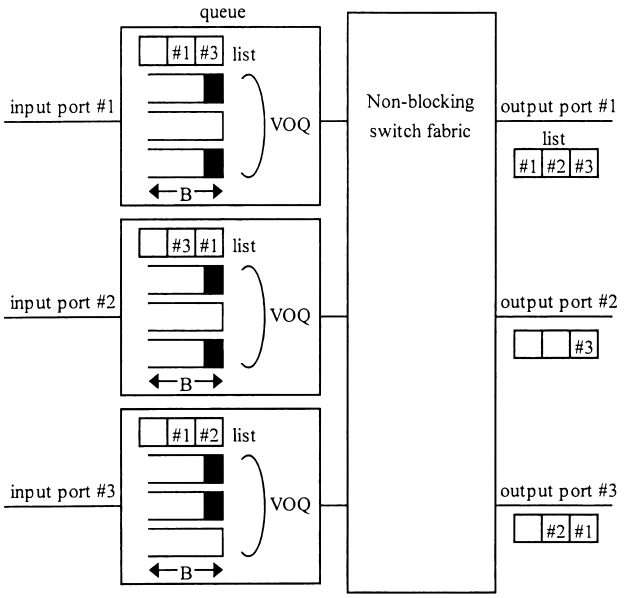

Fig. 2 $N \times N$ input-queued ATM switch with VOQ and i-QOCF ($N = 3$).

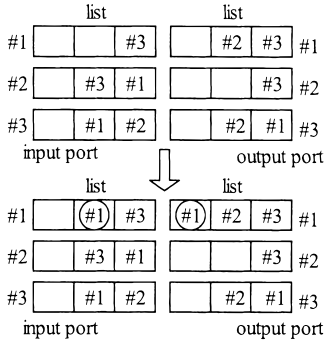

Fig. 3 The operation of i-QOCF in step 1.

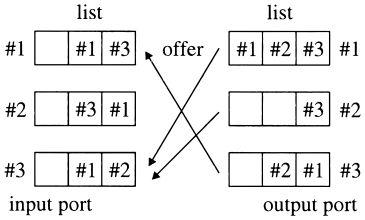

Fig. 4 The operation of i-QOCF in step 2.

put port at input ports, and N is the number of output ports.

In Fig. 2, lists of length N are shown as an example. The list maintained by an input port contains the identifiers for those output ports to which that input port will send a cell. The maximum allowed number of times that an output port identifier can appear in a list maintained by an input port is C when the list length has been set to $C \times N$. In Fig. 2, the list maintained by input port #1 contains an identifier for output port #3 and an identifier for output port #1, because input port #1 has a cell destined for each of those output ports, and $C = 1$. For the same reason, the list maintained by input port #2 contains an identifier for output port #1 and an identifier for output port #3. The list maintained by input port #3 contains an identifier for output port #2 and an identifier for output port #1.

The list maintained by an output port contains the identifiers for input ports that have a cell destined for that output port. The maximum allowed number of times that an input port identifier can appear in a list maintained by an output port is C when the length of the list has been set to $C \times N$. In Fig. 2, the list maintained by output port #1 contains an identifier for each of the input ports #3, #2, and #1, because those input ports have a cell destined for output port #1, and $C = 1$. For the same reason, the list maintained by output port #2 contains an identifier for input port #3. The list maintained by output port #3 contains an identifier for input port #1 and an identifier for input port #2.

i-QOCF is basically composed of six arbitration steps, as follows.

Step 1 When input port $\#i$ ($1 \leq i \leq N$) has k cells destined for output port $\#j$ ($1 \leq j \leq N$), if the number $N_{i,j}$ that output port $\#j's$ identifier actually appears in input port $\#i's$ list is less than the maximum allowed number of times (C), and $(k - N_{i,j})$ is greater than zero, then an identifier for output port $\#j$ is appended to the end of input port $\#i's$ list and an identifier for input port $\#i$ is appended to the end of output port $\#j's$ list.

In the example of Fig. 3, input port #1 has one (k=1) cell destined for output port #1, C is one, and $N_{1,1}$ is zero. Since $N_{1,1}$ is less than C, and $(k - N_{1,1})$ is one that is greater than zero, an identifier for output port #1 is appended to the end of input port #1's list and an identifier for input port #1 is appended to the end of output port #1's list.

Step 2 Output port $\#j$ sends an offer to the input port whose identifier is at the head of its list.

In the example of Fig. 4, The identifiers for input ports #3, #3, and #1 are at the head of the lists maintained by output ports #1, #2, and #3, respectively. Thus, output ports #1, #2, and #3 send offers to input ports #3, #3, and #1, respectively (Fig. 4).

Step 3 If input port $\#i$ receives any offers, it accepts the one that appears first in its list, starting from the head of the list.

In the example of Fig. 5, input port #1 accepts the offer from output port #3 because it received an offer only from output port #3. Input port #2 did not receive any offers. Input port #3 received two offers, one from output port #1 and one from output port #2. It accepts the offer from output

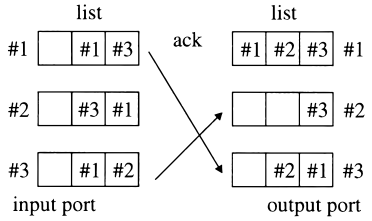

Fig. 5 The operation of i-QOCF in step 3.

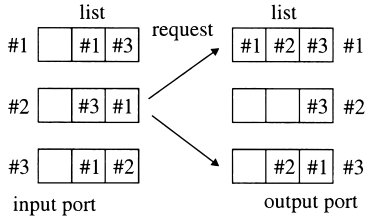

Fig. 6 The operation of i-QOCF in step 4.

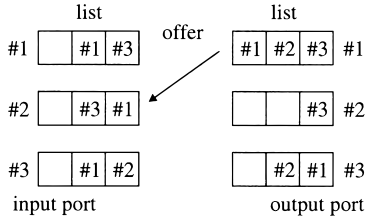

Fig. 7 The operation of i-QOCF in step 5.

port #2 because the identifier for output port #2 appears first in its list, starting from the head of the list (Fig. 5).

Step 4 Each unmatched input port which received no offers in step 3 sends a request to every output port for which it has a cell destined.

In the example of Fig. 6, input port #2 sends requests to output ports #1, and #3 because it has a cell destined for those output ports (Fig. 6).

Step 5 If unmatched output port #j (which was not accepted in step 3) receives any requests, it chooses the one that appears first in its list, starting from the head of the list. Output port #j notifies the input ports that sent a request to it as to whether or not their request was granted.

In the example of Fig. 7, output port #1 chooses input port #2 because it received a request only from input port #2 (Fig. 7).

Step 6 If unmatched input port #i (which received no offers in step 3) receives any offers, it accepts the one that appears first in its list, starting from the head of the list.

In the example of Fig. 8, input port #2 accepts the offer from output port #1 because it received an offer only from output port #1 (Fig. 8).

If input port #i is allowed to send a cell to output port #j, it does so; then the identifier for output port

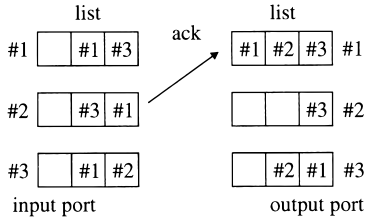

Fig. 8 The operation of i-QOCF in step 6.

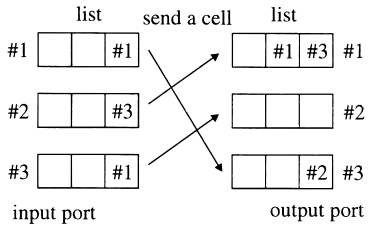

Fig. 9 The operation of i-QOCF when cells are sent.

#j is removed from its list, and the identifier for input port #i is removed from output port #j's list.

In the example of Fig. 9, input ports #1, #2, and #3 are allowed to send a cell to output ports #3, #1, and #2, respectively. The identifiers for output ports #3, #1, and #2 are removed from the lists maintained by input ports #1, #2, and #3, respectively. The identifiers for input ports #2, #3, and #1 are removed from the lists maintained by output ports #1, #2, and #3, respectively.

When the list length is $B \times N$, C equals B. Thus, the identifiers in the list appear in the oldest order, and the i-QOCF gives preference to cells that have been waiting for the longest time. When the list length is less than $B \times N$, the identifiers in the list appear in the quasi-oldest order, and the i-QOCF gives preference to cells that have been waiting for the quasi-longest time.

That is, if we use the lists whose length are $B \times N$, then the operation of i-QOCF is same as that of i-OCF. This reason is as follows. When list length is $B \times N$, in step 2, the identifier that is at the head of list maintained by output port #j is the one of the input port that has the oldest-cell among the input ports that have cells destined for output port #j. Thus, output port #j sends an offer to the input port that has the oldest-cell among the input ports that have cells destined for output port #j.

In step 3 and 6, since the identifiers in the list appear in the oldest order, input port #i accepts the offer from the output port to which it has the oldest-cell.

In step 5, since the identifiers in the list appear in the oldest order, output port #j chooses the request from the input port which has the oldest-cell destined for output port #j.

These operations are the same as those of i-OCF.

The i-QOCF iterates steps 4 through 6 i times.

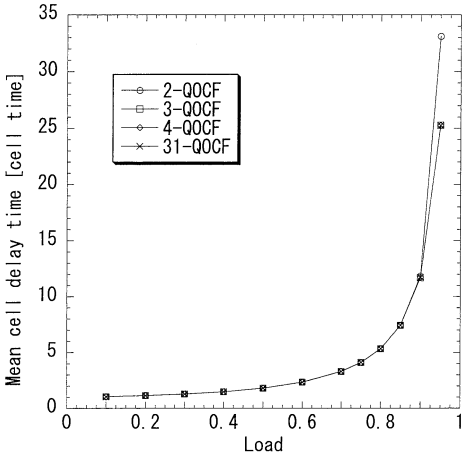

Fig. 10 Mean cell delay time performance when the number of iterations is changed.

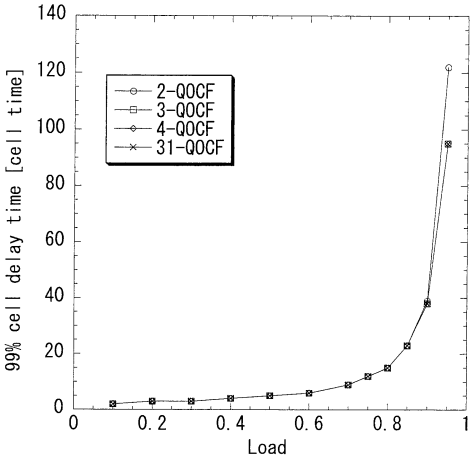

Fig. 11 99% cell delay time performance when the number of iterations is changed.

That is, 4-QOCF means that it iterates steps 1 through 3 once, and iterates steps 4 through 6 four times. Thus, i-QOCF tries $(i+1)$ times to find a conflict-free match between input and output ports.

4. Simulated Performance of i-QOCF Scheduling Algorithm

In this section, we show computer simulation results for i-QOCF and compare them to those for i-OCF. Note that we ignore any hardware design and implementation-specific details because we are mainly concerned with a basic performance evaluation. These other issues are for further study.

We consider the case where cell arrivals at N input ports follow a Bernoulli process. When the input traffic load is ρ, an incoming cell arrives with probability ρ in a cell time, and the probability of no cells arriving is 1-ρ. The incoming cells are distributed uniformly to all output ports. The input traffic is assumed to be homogeneous, and it is distributed uniformly to all input ports. We assume that N is equal to 32.

First of all, we need to clarify how many times i-QOCF should be iterated during one cell time. Ideally, the number of iterations should be N. In other words, i-QOCF should iterate steps 1 through 3 once, and steps 4 through 6 (N-1) times. In practice, there may be insufficient time for N iterations. However, in fact i-QOCF will usually converge in fewer than N iterations.

Figure 10 shows the mean cell delay time when we changed the number of iterations for i-QOCF. We assumed that the length of the list was N. In Fig. 10, if ρ is less than or equal to 0.90 and i is greater than or equal to 2, then the mean cell delay time is almost the same. In the same way, if ρ is less than or equal to 0.95 and i is greater than or equal to 3, then the mean cell delay time is almost the same.

Figure 11 shows the 99% cell delay time when we changed the number of iterations for i-QOCF. In Fig. 11, if ρ is less than or equal to 0.90 and i is greater than or equal to 2, then the 99% cell delay time is almost the same. In the same way, if ρ is less than or equal to 0.95 and i is greater than or equal to 3, then the 99% cell delay time is almost the same.

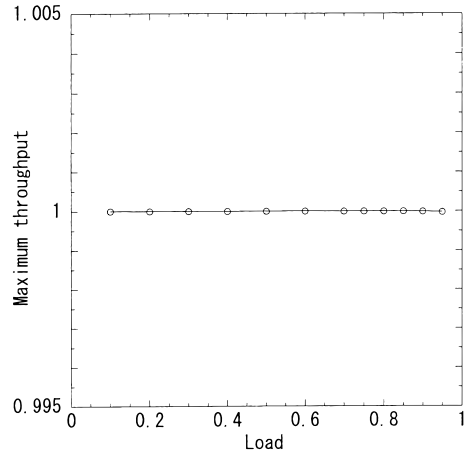

Fig. 12 Maximum throughput performance.

We consider that $i = 3$ is large enough to obtain almost the same cell delay time performance as i-QOCF in which the number of iterations is N. Thus, we will use 3-QOCF in the following performance evaluation of i-QOCF.

We determined the maximum throughput (which is the ratio of the total number of cells transmitted to output ports to the total number of input cells that arrived) to confirm that an input-queued ATM switch with 3-QOCF and VOQ can achieve 100% throughput.

Figure 12 shows the maximum throughput versus the offered load. We can see that the maximum throughput is almost 1.0. That is, we can confirm an input-queued ATM switch with 3-QOCF and VOQ achieve 100% throughput for independent arrival pro-

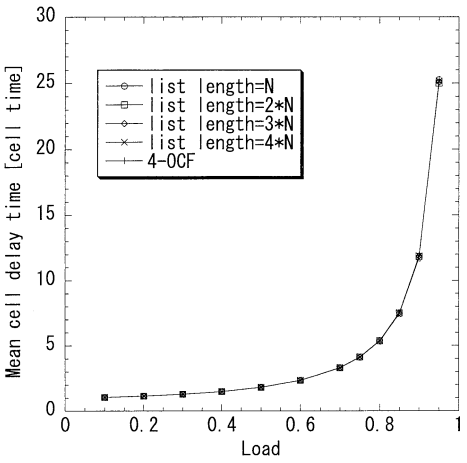

Fig. 13 Mean cell delay time performance when the list length is changed.

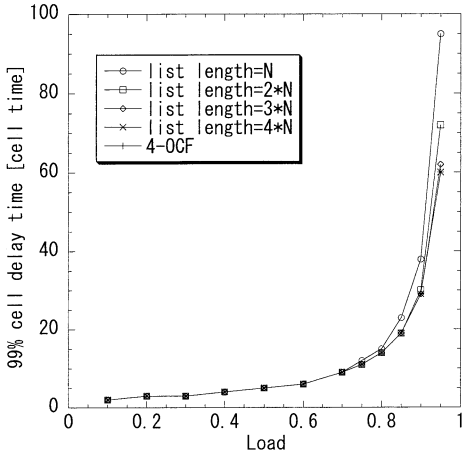

Fig. 14 99% cell delay time performance when the list length is changed.

cesses.

Next, we clarify the effect of the list length. The length of the list can be $N, 2 \times N, \ldots, B \times N$, where B is the size of the buffer that queues cells destined for an output port, and N is the number of output ports. If we use a list whose length is $B \times N$, then the operation of i-QOCF is the same as that of i-OCF as we mentioned in Sect. 3. However, it is expected that list length $B \times N$ is not necessary when i-QOCF has almost the same cell delay time performance as i-OCF.

Figures 13 and 14 show the mean cell delay time and the 99% cell delay time, respectively, for various list lengths. We also show the simulated performance results of 4-OCF in Figs. 13 and 14. Since 3-QOCF tries four times to find a conflict-free match between input and output ports, we compare 3-QOCF with 4-OCF.

In Fig. 13 , we can see that there is no clear distinction among 3-QOCFs with different list lengths. This is because they all achieve virtually the same throughput. For the same reason, there is no clear distinction between 3-QOCF and 4-OCF.

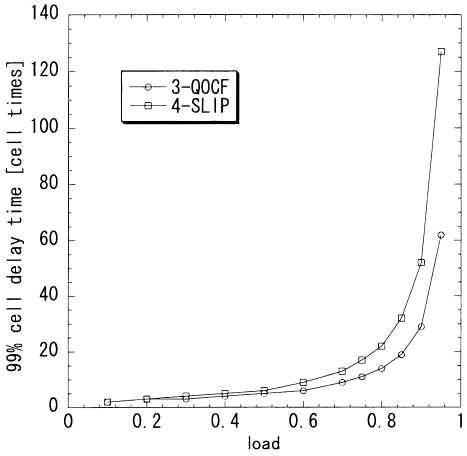

Fig. 15 99% cell delay time of 3-QOCF and 4-SLIP.

In Fig. 14, if ρ is less than or equal to 0.90 and the list length is greater than or equal to $2 \times N$, then the 99% cell delay time is almost the same. In the same way, if ρ is less than or equal to 0.95 and the list length is greater than or equal to $3 \times N$, then the 99% cell delay time is almost the same.

Even though the operation of 3-QOCF, in which the list length is less than $B \times N$, is different from that of 4-OCF, it can be seen that the cell delay time performance of 3-QOCF, in which the list length is $3 \times N$, is almost the same as that of 4-OCF under uniform traffic. In other words, if we want almost the same cell delay time performance as 4-OCF under uniform traffic, we can use 3-QOCF with list length of $3 \times N$.

Next, we compare 3-QOCF, in which the list length is $3 \times N$, to the well-known conventional algorithms 4-SLIP and 4-LQF.

Figure 15 shows the 99% cell delay time of 3-QOCF and 4-SLIP. We do not show the mean delay time of 3-QOCF and 4-SLIP. This is because there is no clear distinction between them. In Fig. 15, we can see that 3-QOCF has better 99% cell delay time performance than 4-SLIP.

Figure 16 shows the relationship between maximum queue length and time of 3-QOCF and 4-LQF under non-uniform traffic. We assume that under non-uniform traffic, all the incoming cells at input port #1 are destined for output port #1, and the incoming cells at other ports are distributed uniformly to all output ports. Maximum queue length is assumed to be the maximum queue length among the queue lengths at input ports except input port #1. In Fig 16, we can see that maximum queue length of 4-LQF is much higher than that of 3-QOCF. That is, 3-QOCF is superior to 4-LQF.

Switch size dependency of the cell delay time is shown in Fig. 17. We assume $\rho = 0.95$ in Fig. 17. Figure 17 shows that the mean cell delay time of 3-QOCF

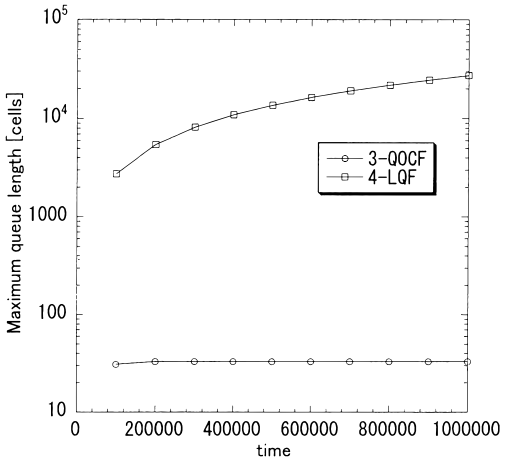

Fig. 16 Relationship between maximum queue length and time of 3-QOCF and 4-LQF under non-uniform traffic.

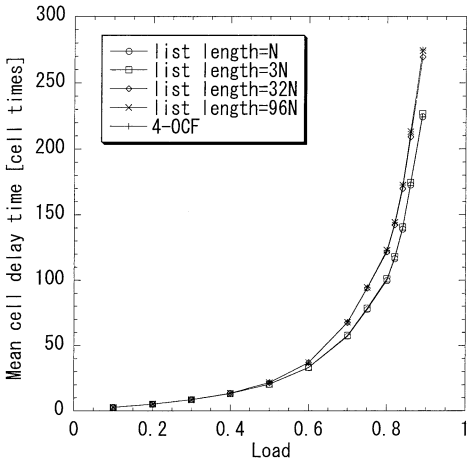

Fig. 18 Mean cell delay time under bursty traffic.

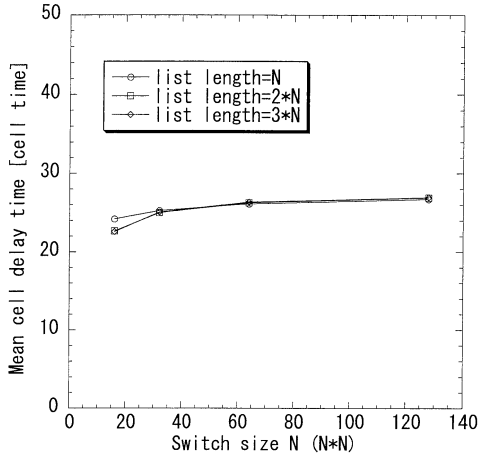

Fig. 17 Mean cell delay time performance versus switch size.

with different list lengths does not increase with switch size. Thus, i-QOCF offers scalability in terms of switch size.

Finally, we show the mean cell delay time of 3-QOCF under bursty traffic. We assume that under bursty traffic, the source alternately produces a burst of full cells (all with the same destination) followed by an idle period of empty cells. The bursts contain a fixed number of cells (32). The idle periods contain a geometrically distributed number of cells and are chosen to adjust the load.

Figure 18 shows the mean cell delay time for various list lengths. We also show the simulated performance result of 4-OCF in Fig. 18. We can see that 3-QOCF has almost the same cell delay time performance as 4-OCF under bursty traffic.

5. Conclusions

In this paper we proposed a new scheduling algorithm for input-queued ATM switches with VOQ, called iterative quasi-oldest-cell-first (i-QOCF) scheduling algorithm.

In the i-QOCF scheduling algorithm, each input port and each output port maintains its own list. The length of the list can be $N, 2 \times N, \ldots, B \times N$, where B is the size of the separate queue for an output port at input ports and N is the number of output ports. The list maintained by an input port contains the identifiers for those output ports to which that input port will send a cell. The list maintained by an output port contains the identifiers for input ports that have a cell destined for that output port. If we use a list whose length is $B \times N$, then the identifiers in the list appear in the oldest order, and i-QOCF gives preference to cells that have been waiting for the longest time. If we use a list whose length is less than $B \times N$, then the identifiers in the list appear in the quasi-oldest order, and i-QOCF gives preference to cells that have been waiting for the quasi-longest time. i-QOCF has almost the same cell delay performance as i-OCF. However, it does not need timestamp information, unlike i-OCF. Moreover, i-QOCF does not require the input ports to send information indicating the waiting time to the output ports, as is needed in i-OCF.

Numerical results have confirmed the excellent performance of the i-QOCF scheduling algorithm. Under uniform traffic, 3-QOCF achieves convergence during one cell time. An input-queued ATM switch with 3-QOCF and VOQ can achieve 100% throughput for independent arrival processes. There is no clear distinction among the mean cell delay time of 3-QOCF with different list lengths. This is because they all achieve almost the same throughput. For the same reason, there is no clear distinction between the mean cell delay time of 3-QOCF and that of 4-OCF. 3-QOCF in which the list length is $3 \times N$, and its 99% cell delay time performance is almost the same as that of 4-OCF. 3-QOCF has some advantages over 4-SLIP and 4-LQF. 3-QOCF offers scalability in terms of switch size. Un-

der bursty traffic, 3-QOCF has almost the same cell delay time performance as 4-OCF. Therefore, the proposed scheduling algorithm will be suitable for future high-speed input-queued ATM switches.

References

[1] K. Thompson, G.J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," IEEE Network, Nov./Dec. 1997.
[2] J. Turner and N. Yamanaka, "Architectural choices in large-scale ATM switches," IEICE Trans. Commun., vol.E81-B, no.2, pp.120–137, Feb. 1998.
[3] H. Ahmadi and W. Denzel, "A survey of modern high-performance switching techniques," IEEE JSAC, vol.7, no.7, pp.1091–1103, Sept. 1989.
[4] Y. Oie, T. Suda, M. Murata, D. Kolson, and H. Miyahara, "Survey of switching techniques in high-speed networks and their performance," Proc. IEEE INFOCOM'90, pp.1242–1251, June 1990.
[5] M. Murata, "Requirements on ATM switch architectures for quality-of-service guarantees," IEICE Trans. Commun., vol.E81-B, no.2, pp.138–151, Feb. 1998.
[6] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input versus output queuing on a space-division packet switch," IEEE Trans. Commun., vol.COM-35, pp.1347–1356, Dec. 1987.
[7] N. McKeown, V. Anantharan, and J. Walrand, "Achieving 100% throughput in an input-queued switch," Proc. IEEE INFOCOM'96, March 1996.
[8] M. Ali, M. Youssefi, and H. Nguyen, "The performance analysis and implementation of an input access scheme in a high-speed packet switch," IEEE Trans. Commun., vol.42, no.12, pp.3189–3199, Dec. 1994.
[9] N. McKeown, Scheduling algorithms for input-queued cell switches, Ph.D. Thesis, University of California at Berkeley, May 1995.
[10] R. Schoenen, G. Post, and G. Sander, "Prioritized arbitration for input-queued switches with 100% throughput," Proc. IEEE ATM '99 Workshop, pp.253–258, May 1999.
[11] M. Nabeshima and N. Yamanaka, "The i-QOCF (iterative quasi-oldest-cell-first) algorithm for input-queued ATM switches," Proc. IEEE ATM'99 Workshop, pp.25–30, May 1999.
[12] M.G. Hluchyj and M.J. Karol, "Queuing in high-performance packet switching," IEEE JSAC, vol.6, no.9, pp.1587–1597, Dec. 1988.
[13] H. Obara, "Optimum architecture for input queuing ATM switches," IEE Electron. Lett., pp.555–557, March 1991.
[14] H. Matsunaga and H. Uematsu, "A 1.5 Gb/s 8×8 cross-connect switch using a time reservation algorithm," IEEE JSAC, vol.9, no.8, pp.1308–1317, Oct. 1991.
[15] M. Ali and H. Nguyen, "A neural network implementation of an input access scheme in a high-speed packet switch," Proc. IEEE GLOBECOM'89, pp.1192–1196, 1989.
[16] R. Bubenik and J. Turner, "Performance of a broadcast packet switch," IEEE Trans. Commun., vol.37, no.1, pp.60–69, Jan. 1989.

Masayoshi Nabeshima received B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 1992 and 1994, respectively. He joined Nippon Telegraph and Telephone Corporation (NTT), Tokyo, Japan, in 1994. He is currently researching high-speed ATM switching systems and traffic management and scheduling mechanisms for ATM networks in the NTT Network Service Systems Laboratories. He is a member of the IEEE Communication Society.

Naoaki Yamanaka was born in Sendai City, Miyagi Prefecture, Japan in 1958. He graduated from Keio University, Tokyo, Japan, where he received B.E., M.E. and Ph.D. degrees in engineering in 1981, 1983 and 1991, respectively. In 1983 he joined Nippon Telegraph and Telephone Corporation's (NTT's) Communication Switching Laboratories, Tokyo, Japan, where he researched and developed high-speed switching systems and high-speed switching technologies, such as ultra-high-speed switching LSI chips/devices, packaging techniques, and interconnection techniques, for broadband ISDN services. Since 1989 he has been developing broadband ISDN things based on ATM techniques. He is now researching ATM-based broadband ISDN architectures and is engaged in traffic management and performance analysis of ATM networks. He is currently a senior research scientist, supervisor, and distinguished technical member in the Broadband Network System Laboratory at NTT. Dr. Yamanaka received the Best of Conference Award at the 40th and 44th IEEE Electronic Components and Technology Conferences, the TELECOM System Technology Prize from the Telecommunications Advancement Foundation, the IEICE Switching System Research Award (twice), and the IEEE CPMT Transactions Part B: Best Transactions Paper Award in 1990, 1994 ,1994, 1996, 1998, and 1996, respectively. Dr. Yamanaka is the Broadband Network Area Editor of the IEEE Communication Surveys, Associate Editor of the IEICE Transactions, and the IEICE Communication Society International Affairs Director, as well as the Secretary of the Asia Pacific Board of the IEEE Communications Society. Dr. Yamanaka is a senior member of the IEEE.