A Performance-Enhanced Parallel Scheduling Algorithm for MIQ Switches Providing a QoS Guarantee

Hakyong Kim, Hyunho Yoon, Kiseon Kim, and Yongtak Lee

Dept. Information and Communications,
Kwang-Ju Institute of Science and Technology (K-JIST)
1 Oryong-dong, Puk-gu, Kwangju, 500-712, S.Korea
E-mail: hykim@ieee.org, Web URL: http://charly.kjist.ac.kr/~hykim/

Abstract

A novel parallel scheduling algorithm, namely a parallel solitary-request-first (PSRF) algorithm, is proposed to improve the performance of the multiple input-queued (MIQ) switch. The proposed algorithm is basically based on the three-phase scheme consisting of Request, Grant, and Accept phases. The essential idea of PSRF is to select the solitary requests with preference. By doing so, PSRF enhances the throughput of the MIQ switch whose input has less number of queues than the switch size. Simulation results for the i.i.d. Bernoulli traffic demonstrate that 2 or 4 queues are appropriate in terms of simplicity, efficiency, and operation speed. Further, to provide a QoS guarantee in the MIQ switch, we developed an enhanced PSRF scheduler which is a kind of the hierarchical or hybrid scheduling algorithm. Hierarchical scheduling is a mixture of a static scheduling with a dynamic scheduling. That is, after finding out an optimal set of input-output matchings, the scheduler chooses one VC connection or a cell in the corresponding virtual queue so as to provide high throughput as well as the QoS guarantee.

1 Introduction

In order to overcome the throughput limit of the single input-queued (SIQ) packet switch, a number of buffering strategies and scheduling algorithms have been put forward in recent years. Among them, the virtual output-queued (VOQ) switch is receiving considerable attention, since it can yield 100% throughput depending on the scheduling algorithm employed. As a special case of the multiple input-queued (MIQ) switches which is the idea of deploying multiple queues in each input, the VOQ switch equips each input with the same number of queues as the switch size and each queue stores the cells for the same destination. It implies that the VOQ switch eliminates the HOL blocking observed in the traditional SIQ switch and, as the result, leads to the improved performance.

Shortcomings of the VOQ switch are (i) that the switch requires N^2 queues when the switch size is N, and (ii) that the controller has to manage those queues simultaneously in a very short time. These cost the implementation of the VOQ switch prohibitively and make the switch infeasible, especially for large dimensions[†]. In addition, the performance enhancement by increasing the number of queues in the MIQ switch is not significant when the number

[†] As a commercialized example using the VOQ switch, the Gigabit Switch Router (GSR) 12000 series of CISCO, designed for the core backbone, has the limited number of line interfaces even though they could provide the broad bandwidth as a whole [1].

of queues is larger than 4 [2]-[5]. In references [2] and [3], the authors found that the hardware implementation and the performance of the MIQ switch is most reasonable when the number of queues (m) is equal to 2 or 4.

Although the downside in hardware implementation of the VOQ switch, many scheduling algorithms have been developed, resulting in satisfactory switching performance. However, most algorithms are either sequential in their operation [6, 7] or require multiple iterations for better performance [8]-[12]. In a very high-speed switch, however, long control time or multiple iteration is not acceptable. In this paper, therefore, we first aim at designing a parallel scheduling algorithm which improves the switch performance with a moderate number of queues.

The scheduling algorithms developed for performance enhancement usually do not consider the QoS of connections, even though the QoS guarantee is one of the most important requirements in the current and future communications environments. The second part of this paper concerns itself with the development of an enhanced PSRF algorithm with QoS provisioning. In the algorithm, the scheduling is performed in a hierarchical manner.

The rest of this paper is organized as follows: In Section 2, we discuss a problem of the randomness of the parallel scheduling algorithm, which degrades the performance of the MIQ switch. Followed is an introduction of the PSRF algorithm, which resolves the randomness problem in a simple manner, and a simple example explaining the operation of the algorithm. In Section 3, simulation results of the PSRF algorithm for the i.i.d. Bernoulli traffic are presented in terms of the saturation throughput, average queue length, and average delay time. Section 4 deals with the QoS problem in the MIQ switch. At first, we newly introduce the concept of the hierarchical or hybrid scheduling in the MIQ switch after categorizing the existing scheduling algorithms. Secondly, we propose an enhanced PSRF (EPSRF) algorithm which improves on the basic PSRF in terms of QoS guarantee. Next, we demonstrate that the EPSRF algorithm can guarantee a QoS requirement for non-bursty and bursty traffics. Finally, Section 5 concludes this paper with remarks on the proposed algorithms.

2 The PSRF Scheduler

The parallel solitary-request-first (PSRF) algorithm introduced in this paper is a parallel algorithm based on our previous sequential algorithm, called Chessboard [7], and the three-phase scheme, represented by PIM [8], which is consisted of the Request, Grant, and Accept phases.

Before describing the proposed algorithm, let's define the *solitary request* be a sole request for a specific output in Request phase, the *solitary output* be an output receiving the solitary request, and the *solitary input* be an input receiving a grant from the solitary output. In Grant phase of the three-phase scheme, then, the solitary output always selects the solitary request due to no output contention there and, as the result, grants the corresponding input (solitary input). The grant from the solitary output, however, may or may not be selected at the input in Accept phase, when the input receives other grants as well and select one randomly among them. It renders the solitary outputs idle during the time slot, resulting in a poor performance.

To relieve such solitary requests which were lost in Accept phase, conventional schemes such as PIM iterate the three phases several times. In the PSRF scheme, however, it adopts a priority instead of multiple iteration. Two differences of the basic PSRF algorithm from the usual three-phase algorithm are (i) that the PSRF scheduler grants higher priority to the solitary request and (ii) that the scheduler ignores the other requests from the solitary input when it grants. The three phases of the basic PSRF algorithm are as follows:

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

Grant phase When an unmatched output receives the solitary request, it chooses the request and marks the solitary input and solitary output. When an output receives two or more requests,

```
/* i:input port number, j:output port number */
for (all i) Request(i) { /* Request phase */
/* Request to every output for which it has a queued cell */
}

for (all j) { /* Grant phase */
    if(num_request[j] == 1) { /* if output j is the solitary output */
        solitary_input[i] = YES;
        solitary_output[j] = YES;
}}

for (all j) Grant(j)
    if(solitary_output[j] == YES)
        Grant_request(i,j);     /* grant the solitary request */
    else if(solitary_output[j] == NO && num_request[j]>1) {
        i = Select(j);     /* select one request for output j */
        if(solitary_input[i] == NO) Grant_request(i,j);
}}

for (all i) Accept(i) { /* Accept phase */
    if(grant[i] == 1)
        Accept_grant(i,j);   /* accept the solitary grant */
    else if(grant[i] > 1) {
        j = Select(i);     /* select one grant for input i */
        if(solitary_output[j] == NO) Accept_grant(i,j);
}}
```

Fig. 1 Pseudo-code for the basic PSRF algorithm

it chooses one randomly among the requests from the non-solitary input. The inputs corresponding to the selected requests are granted.

Accept phase When an input receives only one grant, it accepts the grant. When an input receives more grants than one, it chooses one randomly among the grants from the solitary outputs. If all grants are from the non-solitary outputs, the input accepts one by randomly selecting an output. The outputs corresponding to the selected grants are notified.

Figure 1 shows the pseudo-code for the basic PSRF algorithm. Above three-phase algorithm may be iterated several times.

The idea behind PSRF, similar to that for Chessboard, is well explained via a simple example shown in Fig. 2, based on a 2×2 MIQ switch where each input has 2 separate queues. In input 1, there is only one cell for output 2 (its corresponding request is r_{12}), while, in input 2, there are two cells, each for output 1 (r_{21}) and output 2 (r_{22}), respectively. In this example, request r_{21}, output 1, and input 2 correspond to the solitary request, solitary output, and solitary input, respectively. In Grant phase, therefore, the PSRF scheduler will select the solitary request r_{21} first and the request r_{12} from the non-solitary input. From this example, it is manifest that the selection of r_{12} and r_{21} gives us the highest performance. In this example, there might be a starvation problem when a specific traffic situation continues. Considering that a queue is usually shared by multiple virtual connections, however, it does not matter any more.

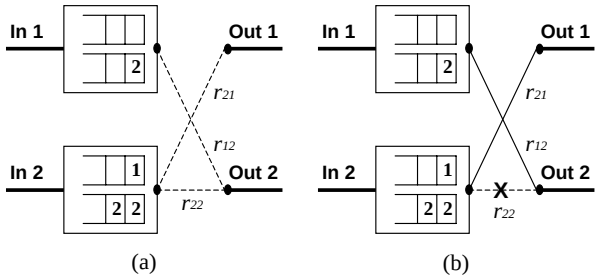

Fig. 2 An example of the 2×2 MIQ switch where each input has 2 separate queues

3 Performance Evaluation

In this study, we performed extensive computer simulations under different circumstances. For the simulations, we assumed that time is slotted into a fixed size, called time slot, and that each input receives an arrival traffic distributed based on the i.i.d Bernoulli distribution.

Fig. 3 plots the curves of the saturation throughputs (viz. the offered load ρ is equal to 1.0) versus number of queues in an input port (m). The solid line and the dashed line represent the result for PSRF and for PIM, respectively, when we iterate both algorithms only once. N designates the switch size and m ranges from 1 to N. As shown in the figure, PSRF outperforms the PIM algorithm except for the bound conditions of $m = 1$ and $m = N$. It is an interesting result that PSRF gives better performance when m takes on a moderate value between 1 and N, while PIM provides its maximum throughput when m is equal to N. In the figure, the throughput for PSRF is maximum when m is equal to 4 except for the case

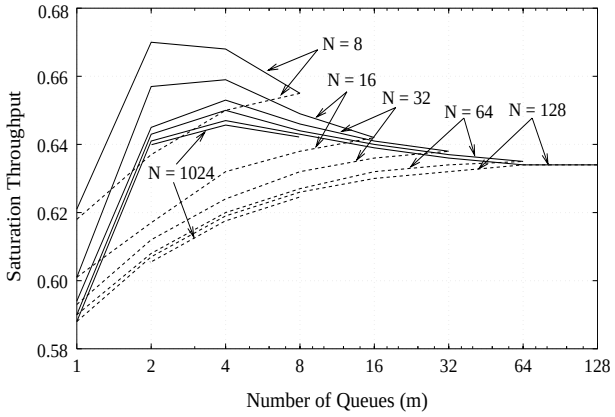

Fig. 3 Saturation throughput of PSRF and PIM for a single iteration.

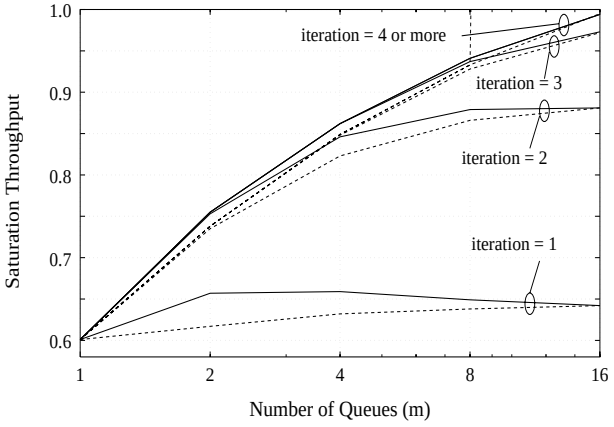

Fig. 4 Saturation throughput of of a 16×16 MIQ switch for varying number of queues and iteration times.

Table 1 Saturation throughput gain obtained by PSRF to PIM. (The bold-faced numbers represent the maximum gains.) (%)

m\N	8	16	32	64	128
1	0.00	0.00	0.00	0.00	0.00
2	**5.85**	**6.48**	**5.39**	**5.92**	**5.60**
4	2.77	4.27	4.65	4.84	4.35
8	0.00	1.72	2.22	2.71	2.72
16	-	0.00	0.78	1.27	1.43
32	-	-	0.00	0.47	0.63
64	-	-	-	0.00	0.00
128	-	-	-	-	0.00

of $N = 8$.

Table 1 tabulates the throughput gain obtained by PSRF to PIM. When m is equal to 2, the saturation throughput for PSRF is higher than that for PIM by about 6% and higher than that of the SIQ switch by 8 to 10%. Notice from Table 1 that the throughput gain is maximum when m is equal to 2. It is also noticeable that the results for PSRF and PIM are equal to each other when m is equal to N, i.e., the VOQ switch. This stems from the fact that, as the number of queues approaches to the switch size, the probability that an output receives the solitary request becomes smaller and, as the result, PSRF becomes indiscernible from PIM.

Fig. 4 displays the saturation throughput of a 16×16 MIQ switch for different number of queues and different number of iteration times. When the number of queues is equal to the switch size of 16, it converges to 1.0 as the iteration times increase. For the switch with 4 queues in each input, however, the performance gain obtained by PSRF to PIM decreases as the iteration times increases. This implies that the PSRF algorithm is appropriate to the switching system having short time for control.

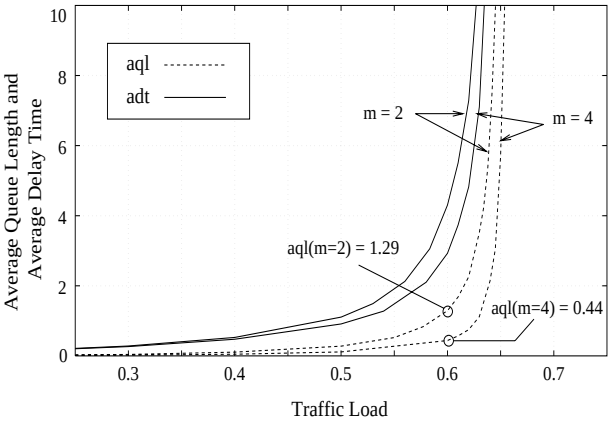

Fig. 5 Average queue length and average delay time for the MIQ switch when the switch size is 16×16 and the number of queues in an input is equal to 2 or 4. (1 iteration)

Fig. 5 plots the average queue length (aql) and average delay time (adt) of the PSRF algorithm for a 16×16 switch. We here considered only two cases of 2 queues and 4 queues. For the offered load of 0.6,

the average delay time of 4-queue case (2.9 cell slots) is about two third of that of the 2-queue case (4.3 cell slots). For the same offered load, the average queue length of 4-queue case (0.44 cells) is about one third of that of the 2-queue case (1.29 cells). It implies that doubling or increasing the number of queues requires less memory space. Therefore, 4-queue case is optimal in terms of both memory space and performance, when we consider a single iteration of the PSRF algorithm. The more times we iterate the PSRF algorithm, the steeper the elbow of the curves becomes.

4 QoS Guarantee with PSRF

In this section, we newly introduce the concept of the hierarchical scheduling to consider the QoS problem in the MIQ switch. Existing work in the MIQ switches has largely been focusing on improving the switch's throughput, relatively less attention has been given to the development of proper scheduling algorithm which can support diverse traffic type and satisfy the potentially different QoS of different streams, which is the focus of the second part of this paper. We named the scheduling prototype as the hierarchical scheduling algorithm. As an example of the hierarchical scheduling algorithm, we propose the enhanced PSRF (EPSRF) at the end of Subsection 4.1. In Subsection 4.2, we demonstrate that EPSRF can provide the QoS guarantee via computer simulation.

4.1 Hierarchical Scheduling Concept

The basic PSRF algorithm proposed in Section 2, in fact, is a dynamic scheduling algorithm which makes scheduling decisions every time slot. In general, dynamic schedulers such as PIM [8] and *i*SLIP [10] including basic PSRF may not guarantee the connections' requested QoS, even though they provide better throughput than their static counterparts. This fact implies that the dynamic scheduler can be used to switch cells belonging to best-effort service category rather than real-time services. To overcome the aforementioned problem, we developed a hierarchical scheduling algorithm for MIQ switches, which based on the basic PSRF algorithm described in Section 2 and 3. We named it enhanced-PSRF (EPSRF) algorithm.

Before describing the EPSRF algorithm in detail, we explain the general concept of hierarchical scheduling in the MIQ switch. In the output-queued switch, scheduling is performed between VC queues in each output port, since cells are switched to appropriate output ports on arriving at input ports. The mechanism supplies the valuable QoS guarantee to the inherent good throughput and delay characteristics. In the MIQ switch, meanwhile, scheduling is performed to just find an optimal set of input-output matching. It makes it possible to improve the switching throughput, but couldn't guarantee the QoS requirements of various applications. To provide the QoS guarantee in the MIQ switch, scheduling should be performed between VC queues at input ports after finding out input-output pairs. In the VOQ switch, for example, the scheduler chooses one queue in an input at first, which corresponds to the process of finding out input-output pair, followed by scheduling between VC queues within the selected queue. Such scheduling algorithm used in the MIQ switch is called *the hierarchical scheduling* or *the hybrid scheduling*, since the scheduler uses both the dynamic algorithm (in finding out input-output pairs) and the static algorithm (in choosing a specific VC connection in the selected queue). A challenge is that we have to consider connections' priority in finding out input-output pairs, which can be resolved by assigning different priorities to each traffic class. This problem is an important issue in designing the hierarchical scheduler.

The EPSRF algorithm, as an example of the hierarchical scheduling algorithm, uses the basic PSRF algorithm in finding out an optimal set of input-output matchings. For providing QoS guarantees, it gives higher priority to the real-time (RT) traffic over the non-real-time (NRT) traffic[‡], since it is the minimum requirement to prioritize RT traffic over NRT traffic or best-effort services [13]. That is, if there

[‡] The services with diversified QoS requirements is also classified into the *delay sensitive* class and the *loss sensitive* class, each of which is corresponding to RT traffic and NRT traffic, respectively [14].

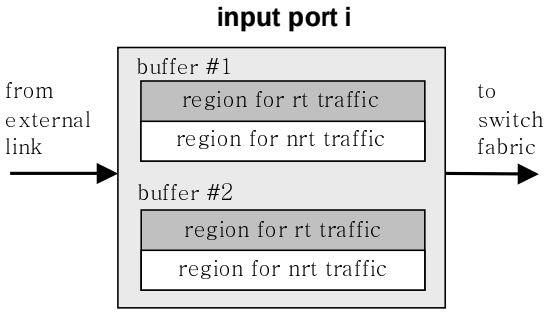

Fig. 6 Buffer architecture used in simulation.

exist both RT traffic and NRT traffic for a specific output, the scheduler chooses the RT traffic as long as the queue length of the NRT traffic is less than a preset threshold value. When the queue length of the NRT traffic becomes larger than the predefined threshold value, the scheduler chooses one of the NRT traffics. By doing so, we could guarantee a minimum delay for the NRT traffic and a minimum cell loss ratio for the NRT traffic.

4.2 Performance Evaluation of the EPSRF Algorithm

In order to evaluate the EPSRF algorithm, we performed computer simulation. The assumptions used in the simulation are as follows: The switch size used is 16×16 and each input port has 2 queues separated logically or physically. That is, each queue is dedicated to 8 output ports as in the Odd-Even switch [15, 16]. Each queue is further divided into two sections logically, regions for RT traffic and NRT traffic as shown in Figure 6 which depicts the input port i and its queue architecture. Although multiple VC connections can arrive at an input and share each region, we assume for simplicity that there exists only two VC connections, one for RT connection and the other for NRT connection. Both the RT traffic and NRT traffic are assumed to follow the ON-OFF traffic pattern. The length of ON and OFF is determined by the geometric distribution with the mean of λ_{ON} and λ_{OFF}. During the ON period, cells arrive continuously, while during the OFF period, no cell arrives. ON and OFF traffics are independent and identical to each other in terms of statistical viewpoint.

Table 2 Cell loss ratio (crl), average queue length (aql), and average delay time (adt) for each type of traffic. (switch size = 16×16, m=2, 1 iteration, BS_{RT} = 20, BS_{NRT} = 1000, Th_{NRT} = 950 cell units)

load	RT traffic			NRT traffic		
	crl	aql	adt	crl	aql	adt
0.57	$2.2e^{-7}$	0.56	3.9	N/A	0.7	4.96
0.63	$2.6e^{-4}$	1.85	11.9	N/A	2.3	14.7
0.69	$1.1e^{-2}$	5.06	29.1	$1.0e^{-1}$	132	818

Table 3 Cell loss ratio (crl), average queue length (aql), and average delay time (adt) for each type of traffic. (switch size = 16×16, m=2, 2 iterations, BS_{RT} = 20, BS_{NRT} = 1000, Th_{NRT} = 950 cell units)

load	RT traffic			NRT traffic		
	crl	aql	adt	crl	aql	adt
0.57	N/A	0.33	2.3	N/A	0.38	2.7
0.63	N/A	0.51	3.2	N/A	0.62	3.9
0.69	$1.0e^{-7}$	0.78	4.5	N/A	1.03	5.9

Table 2 and 3 show the simulation result for a non-bursty ON-OFF traffic. Cells arriving in the ON period destine for output with the uniform distribution. For those connections, we fixed that λ_{OFF} is equal to 15 cell units, but λ_{ON} takes on either 6, 7, or 8 cell units. It corresponds to the offered load of 0.57, 0.63, and 0.69, since there are two connections for an input, one RT and one NRT connection. Buffer space of 20 cells and 1000 cells are allotted for RT traffic NRT traffic, respectively. If the length of the NRT-traffic queue becomes larger than 950 (Th_{NRT}), NRT traffic has higher priority in Request phase over RT traffic. The running time was 10^8 cell times.

As shown in Table 2, for moderate traffic load such as 0.57 or 0.63, NRT traffic has the cell loss ratio (clr) less than $1.0e^{-8}$. However, for traffic load of 0.69, it shows a considerable clr due to buffer overflow. In fact, for the given assumption of 16×16 switch with 2 distinct queues, its saturation throughput or maximum admissible load is 0.657 as shown

Table 4 Cell loss ratio (crl), average queue length (aql), and average delay time (adt) for each type of traffic in bursty case. (switch size = 16×16, $m = 2$, 1 iteration, $BS_{RT} = 50$, $BS_{NRT} = 1000$, $Th_{NRT} =$ 950 cell units)

load	RT traffic			NRT traffic		
	crl	aql	adt	crl	aql	adt
0.57	$2.95e^{-3}$	1.71	3.91	N/A	2.68	18.7
0.63	$5.87e^{-2}$	5.49	36.7	N/A	35.4	222
0.69	$1.28e^{-1}$	7.68	50.7	$7.19e^{-2}$	74.6	449

Table 5 Cell loss ratio (crl), average queue length (aql), and average delay time (adt) for each type of traffic in bursty case. (switch size = 16×16, m=2, 2 iterations, $BS_{RT} = 50$, $BS_{NRT} = 1000$, $Th_{NRT} =$ 950 cell units)

load	RT traffic			NRT traffic		
	crl	aql	adt	crl	aql	adt
0.57	$1.93e^{-4}$	0.84	5.91	N/A	1.06	7.44
0.63	$2.04e^{-3}$	1.58	9.96	N/A	2.31	14.5
0.69	$1.18e^{-2}$	2.87	16.7	N/A	5.51	31.7

in Fig. 3, when we iterate the basic PSRF algorithm once in a cell time slot. Therefore, the offered load of 0.69 makes the performance measures worse as simulation time goes. For RT traffic, it guarantees a minimum delay time and queue length even though it suffers from cell loss due to a limited buffer space. The normalized throughput, the ratio of obtained throughput to the offered throughput, is 1.0 both for RT and NRT traffic when the offered load is 0.63.

Table 3 shows the result when we iterate the EPSRF algorithm twice in a cell time slot. In this case, we can say that crl values both for RT traffic and NRT traffic is less than $1.0e^{-8}$. aql and adt are much less than those in Table 2.

Table 4 and 5 show the result for ON-OFF traffic in which cells arriving during an ON period destine for the same output port. That is, the connections have the bursty characteristics. The other conditions were assumed same as the previous one except the buffer space for RT traffic. In this case, the performance becomes worse than those in Table 2 and 3 since the traffic has the bursty property. Nevertheless, RT traffic still experiences short delay time and NRT traffic suffers less from the cell loss. The normalized throughput is 0.984 and 0.971 for RT and NRT traffic, respectively, when the offered load is 0.63.

In this section, we have proposed and evaluated the EPSRF scheduling algorithm which is a hierarchical or hybrid scheduling algorithm. Even though EPSRF is very simple in its operation, we could provide both high throughput and QoS guarantee with different type of connections. Although we performed a simple simulation to validate the proposed algorithm, it is straightforward to extend the hierarchical concept in scheduling into multiple or variable number of VC connections.

5 Conclusions and Remarks

In this paper, we have proposed a parallel scheduling algorithm for the MIQ switch. The proposed PSRF algorithm improves the performance of MIQ switches with moderate number of queues by selecting the solitary request with preference. Computer simulation results show that the throughput of PSRF is optimized when the number of queues are equal to 2 or 4 for a single iteration, while the throughput is proportional to the number of queues for multiple iterations. The result implies that switches using the PSRF algorithm as well as a moderate number of queues, need lower hardware complexity and shorter control time, which leads to a high-performance and high-speed switch.

In order to provide the QoS guarantee in the MIQ switch, we improves on the basic PSRF algorithm to the enhanced PSRF (EPSRF) which is a kind of hierarchical scheduling algorithm. In the scheme, the scheduler chooses one among multiple VC connections within a queue for the same output in a static way, after finding out input-output pairs dynamically by using the basic PSRF. Through the computer simulation, we showed that the EPSRF algorithm can guarantee the QoS of each connection in a simple manner as well as provide higher throughput.

In this study, we assumed that only two connections, one for RT traffic and the other for NRT traffic, are allowed for an input, which simplified our performance evaluation of the EPSRF algorithm. However, multiple VC connections with diversified QoS requirements are expected to exist in current and future communications environments, and thus, it becomes a research topic to develop an optimal hierarchical scheduling algorithm for those VC connections.

References

[1] "Cisco 12000 Series - Gigabit Switch Routers," http://www.cisco.com/univercd/cc/td/doc/pcat/12000.pdf

[2] G. Thomas, "Bifurcated queueing for throughput enhancement in input-queued switches," *IEEE Commun., Letters*, March 1997, 1 (2), pp. 56-57.

[3] G. Thomas and V. Veludandi, "ATM switches with bifurcated input queueing," *Proc. ICCCN '97*, Las Vegas, NV, USA, 22-25 Sept. 1997, pp. 504-507.

[4] Hakyong Kim, C.H. Oh and K.S. Kim, "Throughput analysis of the bifurcated input-queued switches with restricted contention," *IEE Electronics Letters*, 20th August 1998, 34 (17), pp. 1651-1652.

[5] Hakyong Kim, C. Oh, Y. Lee, and K. Kim, "Throughput analysis of the bifurcated input-queued ATM switch," *IEICE Tr. Communi*, May 1999, E82-B (5), pp.768-772.

[6] R. O. LaMaire and D. N. Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Tr. Networking*, Oct. 1994, 2 (5), pp. 471-482.

[7] Hakyong. Kim, K. Kim, Y. Lee, H. Yoon, and C. Oh, "A simple and efficient cell selection algorithm for the multiple input-queued ATM switch," *Proc. IEEE ATM Workshop'99*, Kochi, Japan, 24-27 May 1999, pp.259-264.

[8] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Tr. Computer Systems*, Nov. 1993, 11 (4), pp.319-352.

[9] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. thesis, University of California at Berkeley, May 1995.

[10] N. McKeown, "The *i*SLIP scheduling algorithm for input-queued switches," *IEEE/ACM Tr. Networking*, April 1999, 7 (2), pp. 188-201.

[11] M. Nabeshima and N. Yamanaka, "The *i*-QOCF (iterative quasi-oldest-cell-first) Algorithm for input-queued ATM switches," *Proc. IEEE ATM Workshop'99*, Kochi, Japan, 24-27 May 1999, pp.25-30.

[12] S.Y. Liew, S.W. Cheng, and T.T. Lee, "An enhanced iterative scheduling algorithm for ATM input-buffered switch," *Proc. IEEE ATM Workshop'99*, Kochi, Japan, 24-27 May 1999, pp.103-108.

[13] K. Rothermel, "Priority mechanisms in ATM networks," *Proc. IEEE GLOBECOM '90*, Dec. 1990, vol. 2, pp. 847-851.

[14] S.G. Jong and Y.O. Chin, "Algorithms on dynamic priority scheduling for heterogeneous traffic in ATM," Proc. of 18th Conference on Local Computer Networks, Sept. 1993, pp.380-387.

[15] C. Kolias and L. Kleinrock, "The Odd-Even input-queueing ATM switch: performance evaluation," *Proc. IEEE ICC '96*, June 1996, vol. 3, pp. 1674-1679.

[16] C. Kolias and L. Kleinrock, "The Odd-Even ATM switch," *IEICE Tr. Communications*, Feb. 1998, E-81-B (2), pp. 244-250.