Grant-Aware Scheduling Algorithm for VOQ-Based Input-Buffered Packet Switches

Kyeong-Eun Han, Jongtae Song, Dae-Ub Kim, JiWook Youn, Chansung Park, and Kwangjoon Kim

In this paper, we propose a grant-aware (GA) scheduling algorithm that can provide higher throughput and lower latency than a conventional dual round-robin matching (DRRM) method. In our proposed GA algorithm, when an output receives requests from different inputs, the output not only sends a grant to the selected input, but also sends a grant indicator to all the other inputs to share the grant information. This allows the inputs to skip the granted outputs in their input arbiters in the next iteration. Simulation results using OPNET show that the proposed algorithm provides a maximum 3% higher throughput with approximately 31% less queuing delay than DRRM.

Keywords: Grant-aware, High performance, Input-buffered switch, Scheduling, Low latency, Maximal matching algorithm.

Manuscript received Sept. 13, 2017; revised Dec. 22, 2017; accepted Mar. 5, 2018.

Kyeong-Eun Han (corresponding author, kehan@etri.re.kr), Jongtae Song (jsong@etri.re.kr), Dae-Ub Kim (artkdu@etri.re.kr), JiWook Youn (younjw@etri.re.kr), Chansung Park (chansung18@etri.re.kr), and Kwangjoon Kim (kjk@etri.re.kr) are with the Hyper-connected Communication Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

The recent exponential growth of Internet traffic and newly emerging data-intensive applications, such as streaming video, social networking, and cloud computing, requires networks that have much higher bandwidth and lower latency [1]–[4]. Because of the new traffic characteristics and the delay requirement of DC networks, control architectures [5] and scheduling algorithms [6], [7] have been widely investigated.

To accommodate these requirements, fast and high-capacity switching structures and corresponding fast control schemes have been proposed [4], [8]–[10]. High-speed switches are built around a virtual output queue (VOQ)-based input-buffered switch architecture in order to eliminate the head-of-line (HOL) blocking, and a centralized scheduler uses a fixed-size time slot as a transfer unit. Variable-length packets are framed to a fixed-size synchronized time slot and transferred across the central switching fabric [10]–[15]. Most studies on scheduling algorithms have focused on the contention resolution problem in VOQ-based input buffered switches because the problem of HOL blocking was eliminated using VOQ [4], [12]–[14], [16]–[22].

Maximum matching algorithms, such as longest queue first (LQF), oldest cell first, and longest port first (LPF), guarantee 100% throughput under uniform traffic conditions. However, it is known that under nonuniform traffic conditions, LQF and LPF can lead to instability and starvation [4], [11], [12], [14]. Furthermore, their practical implementation is difficult owing to their high complexity. Recently, much attention has been given to maximal matching algorithms as a fast, efficient, and feasible packet switching scheduler [12], [16]–[25], because their implementation complexity is significantly lower than

maximum matching algorithms, although they are sub-optimal.

Maximal matching algorithms, including parallel iterative matching (PIM) [19], iterative round-robin matching (iRRM) [20], FIRM [21], iterative round-robin with SLIP (iSLIP) [22], and dual round-robin matching (DRRM) [23], [24] were proposed as a technique to enable ease of implementation. Among them, DRRM and iSLIP are widely used for the scheduling of packet switches because they can provide fairness, higher efficiency, and a desynchronization effect. In particular, DRRM is preferred for large-scale switches owing to its faster arbitration and relative ease of implementation compared to iSLIP [25]. To maximize the number of input and output matching, the maximal matching algorithm is generally combined with multiple iterations. This achieves higher throughput and less delay than with a single iteration. Maximal matching algorithms can provide throughput values that are as high as maximum matching algorithms that use multiple iterations. It is known that a maximal match can always be found within N iterations, where N denotes the number of ports in the switch. DRRM and iSLIP can achieve a maximal matching with a reduced number of iterations, $\log_2 N$ [12], [22]–[25]. In practical terms, with large switches, the maximal matching algorithm should be applied within a limited time, and the maximum allowed number of iterations can be reduced owing to the capacity of the processing hardware. This limits the maximum size of the switch that can be implemented. If a sufficient number of iterations are not allowed, the performance may be degraded. This constraint can be alleviated using a scheduling algorithm that can apply a smaller number of iterations than existing algorithms.

In this study, we propose a grant-aware (GA) scheduling algorithm to provide better performance than existing DRRM. In the proposed scheme, we use the grant indicator so that all inputs share output-grant information. The output sends the grant to the selected input and the grant indicator to all the remaining inputs. Hence, the inputs know which outputs have already been matched for this time slot and exclude them in the next round of iteration.

The remainder of this paper is organized as follows. In Section II, an overview of DRRM scheduling and its problem are shown. In Section III, the proposed GA scheduling algorithm is described in detail. In Section IV, we evaluate its performance in terms of throughput and queuing delay by performing a simulation using OPNET followed by some concluding remarks in Section V.

II. Dual Round-Robin Matching Algorithm

As previously mentioned, DRRM [12], [23], [24] provides the least hardware complexity among the existing maximal matching algorithms by performing two-phase scheduling. Other maximal matching algorithms, such as PIM [19], iRRM [20], FIRM [21], and iSLIP [22] perform three-phase scheduling in each iteration. PIM uses random selection arbiters to perform input-output matching in both grant and accept phases. iRRM uses round-robin (RR) arbiters instead of random selection for input-output matching. Arbiters of iRRM update their RR pointers immediately after selecting one request in the grant phase or one output in the accept phase. Under uniform traffic conditions, PIM provides a throughput of approximately 65% at a single iteration and a 100% throughput with N iterations when the switch size is $N \times N$ [12]. While PIM brings unfairness and a high cost for the implementation of random functions at high speed, iRRM provides good fairness using a RR manner, but only a 50% throughput under heavy load conditions owing to the output pointer update mechanism [12]. To solve this problem of iRRM, iSLIP was proposed. In iSLIP, outputs increment their arbiter pointers only when their grants are accepted by inputs. iSLIP can achieve a 100% throughput with a single iteration under uniform and independent and identically distributed (i.i.d) traffic. Similar to iSLIP, FIRM also uses RR arbiters for input-output matching. In FIRM, if the grant is not accepted, its RR arbiter is set to the granted input. If the grant is accepted, the output RR pointer is incremented by 1 to the granted input. Using this scheme for output arbiter updates, FIRM can provide more fairness than iSLIP [12].

Similar to iSLIP, DRRM [23], [24] uses RR arbiters for input-output matching at both inputs and outputs. Each input arbiter maintains the VOQ status of its corresponding input port and requests based on a nonempty VOQ status to one corresponding output. DRRM operates in two phases, the request and grant phases, during each iteration as follows.

- Request: Each input sends an output request for the first nonempty VOQ based on an RR pointer of the input arbiter, starting from the current position of the pointer. The pointer is incremented by 1 only when the request is granted; otherwise, it remains unchanged.
- Grant: If an output receives one or more requests, it chooses one using the output arbiter, starting from the current position of the RR pointer. The output notifies

https://doi.org/10.4218/etrij.2017-0057

each requested input whether it was granted or not. The RR pointer of the output arbiter is incremented by 1 from the granted input. If there are no requests, the pointer remains unchanged.

For multiple iterations, the scheduler repeats the request and grant processes, and only those inputs and outputs that are not matched at the end of the previous iteration are eligible for participation in the next matching iteration. Here, both the input and output arbiter pointers are updated only at the first iteration. The DRRM performs arbitration faster than other algorithms with similar performance owing to its lower implementation complexity because it has less information exchange between input and output arbiters. Thus, DRRM can provide not only 100% throughput under i.i.d. and uniform traffic, but also scalability and fairness without the starvation problem [12].

Figure 1 shows an example of a DRRM algorithm operation for mapping between inputs and outputs in a 4×4 switch. Here, the maximum required number of iterations, $k_{\max}$, is 2 [$= (\log_2 N)$ with $N = 4$]. In Fig. 1(a), the inputs of I1, I2, I3, and I4, which have nonempty VOQs, send requests to their selected outputs, O2, O1, O2, and O2, respectively, during the request phase. In Fig. 1(b), the output O2 receives multiple requests from inputs I1, I3, and I4, and it grants the request from the selected input I1 based on its RR pointer. The output RR pointers of O1 and O2 are now pointing at 3 and 2, respectively. In Fig. 1(c), the inputs I3 and I4, which have a nonempty VOQ and have not yet been granted, send

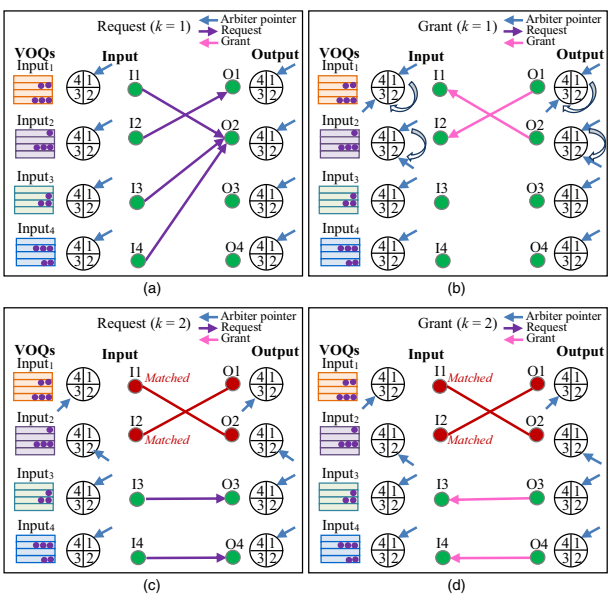

Fig. 1. Operation of DRRM over a 4×4 switch: (a), (b) the first iterations, and (c), (d) the next iteration cycle.

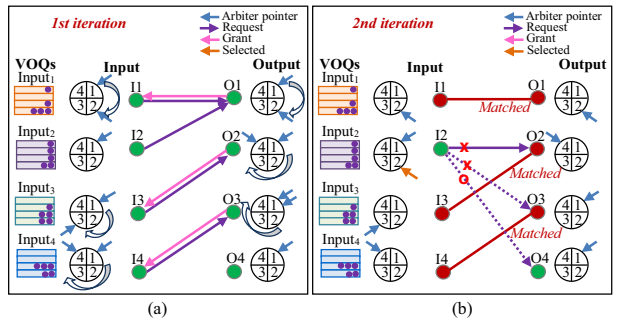

Fig. 2. Example of inefficient iterations in DRRM: (a) request-grant process in the 1st iteration and (b) request-grant process in the 2nd iteration

requests to outputs O3 and O4 based on their RR arbiters, respectively. In Fig. 1(d), I3 and I4 are granted from O3 and O4, respectively. Both input arbiter and output arbiter pointers remain the same. Note that the pointers of all arbiters are updated during the first iteration only. As described above, in DRRM, an input selects one of its nonempty VOQs using the RR manner, and sends a request to the corresponding output. If the output has been matched in a previous iteration, this request is in vain and the input wastes the opportunity. If inputs know the output matching information of the previous iterations, they can send a request only to unmatched outputs.

An example is shown in Fig. 2. In Fig. 2(a), all inputs send requests, and inputs I1, I3, and I4 are granted to outputs O1, O2, and O3, respectively, during the first iteration. However, input I2 fails to obtain the grant from output O1. The input I2 conducts the second iteration, as shown in Fig. 2(b). It selects the next nonempty VOQ and sends the request to the corresponding output O2. However, O2 has been matched with I3 in the previous iteration. It takes two more iterations for input I2 to find the available output O4 because input I2 does not know that the outputs O2 and O3 have already been matched during the first iteration. To avoid this inefficiency, the inputs need to know the grant status of all outputs in the previous iterations.

III. Grant-Aware Scheduling Algorithm

To improve the performance of existing DRRM, we introduce a grant-aware (GA) scheme, where the matching status of the outputs is shared by all inputs during each iteration.

The scheduler gets the information of all VOQs at the beginning of every time slot. If a VOQ is not empty, its VOQ status is set to one; otherwise, it is set to zero. After the VOQ status setting is completed, the scheduler

performs a GA algorithm for maximal matching for all inputs and outputs.

First, each input selects one nonempty VOQ with the input arbiter and sends a request to the corresponding output. The output selects an input among the requesting inputs and allocates a time slot to the selected input. The output sends a grant to the selected input, as well as a grant indicator to all other inputs to share the output granted information.

We define G_{ij} and I_{ij} as the grant and grant indicator from output j to input i, respectively, which indicate the following.

- G_{ij}: The request is granted. Output j will match with input i in this time slot.
- I_{ij}: Output j is matched and no further request will be accepted in this time slot.

The pointer of the output arbiter is incremented to the location next to the granted input. The granted inputs, that is, $G_{ij} = true$, update their input arbiter's pointers to the location next to the selected output during the first iteration, and they reset all of their VOQ statuses. These granted inputs are excluded from the next iteration. All the non-granted inputs reset the already matched j-th VOQ status if $I_{ij} = true$, according to the output granted information. During the next iteration, the inputs that have not yet been granted select one output with a new VOQ status and send requests. This iteration procedure is repeated up to $k_{\max}$ times. After completing the scheduling, the scheduler returns the scheduling result with a grant message to all input nodes. The input nodes transmit the data to the destined output nodes through the allocated time slot.

Figure 3 shows the operation of our GA scheduling algorithm in a 4 × 4 switch. All inputs make a request to their selected output based on the nonempty VOQ status and RR input arbiter, as shown in Fig. 3(a). During the grant phase of Fig. 3(b), outputs O1 and O2 select inputs I2 and I1, respectively. Output O1 grants input I2, and it simultaneously sends grant indicators to other inputs I1, I3, and I4. Output O2 grants input I1, and it also sends grant indicators to all other inputs I2, I3, and I4. Outputs O1 and O2 update their output arbiter pointers to 3 and 2, respectively. The granted inputs I1 and I2 update their input arbiter pointers to 3 and 2, respectively, and they reset all of their VOQ statuses. They then ignore the received grant indicators. Inputs I3 and I4 receive two grant indicators from outputs O1 and O2, and they reset their VOQ status of {VOQ_{31}, VOQ_{32}} and {VOQ_{41}, VOQ_{42}}, respectively. In the request phase of the next iteration, as shown in Fig. 3(c), inputs I3 and I4 can use the newly updated VOQ status, where the VOQs for the matched outputs are not counted on. Thus, inputs I3 and

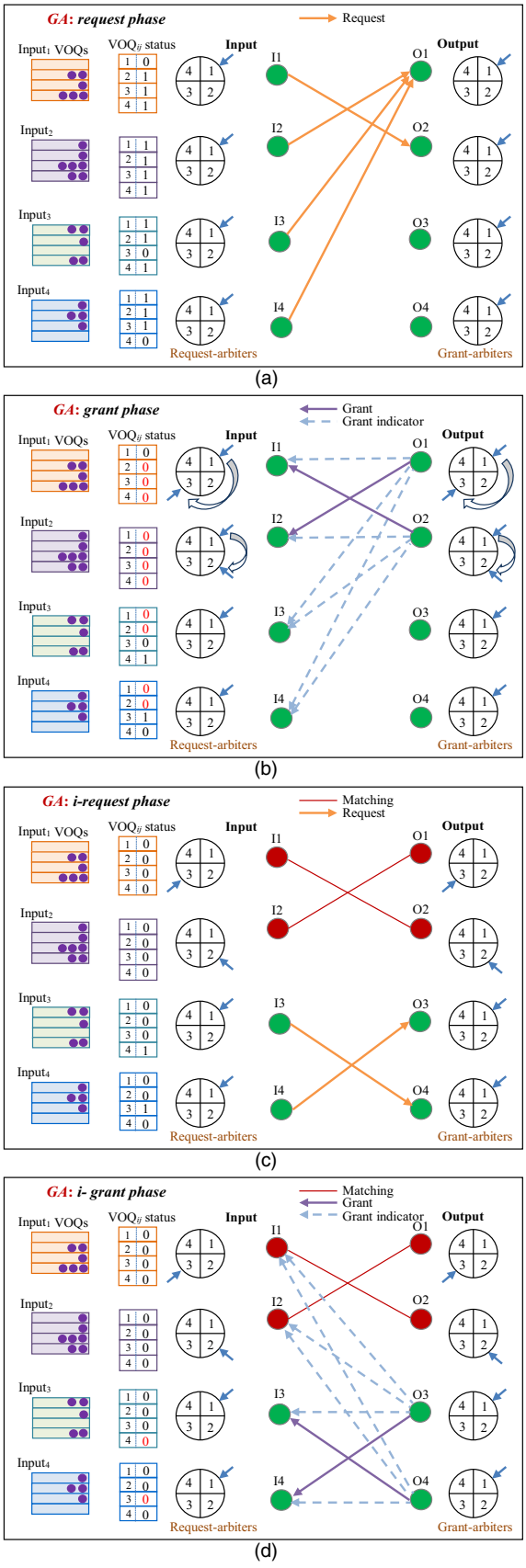

Fig. 3. Operation of GA scheduling algorithm (k = 2).

I4 can skip the process of sending a request to output O2, and it can directly make a request to outputs O4 and O3, respectively. During the grant phase of Fig. 3(d), inputs I3 and I4 receive grants from outputs O4 and O3, respectively. Here, the input arbiter pointers of I3 and I4 remain the same because it is not the first iteration. The pseudo code below shows the operation of our two-phase GA scheduling algorithm.

Pseudo code for grant-aware scheduling algorithm:

Definition of parameters

- N_q: number of VOQs (that is, number of outputs)
- N_i: number of inputs
- IArr_i: input arbiter RR pointer for input i
- OArr_j: output arbiter RR pointer for output j
- selected_i: indicates that any nonempty VOQ of input i is selected
- req_i^j: indicates that input i sends request to output j
- grant_i: indicates that input i is granted
- grant_indicator_j: indicates that output j is matched with any input.
- *iteration*: the remaining number of iterations. The value is set to the required number of iterations and decremented in each iteration.
- voq_{ij} : VOQ status corresponding output j for input i
- *first_iteration*: indicates whether the current iteration is the first iteration or not
- granted_output_i: the granted output for input i

```
REQUEST:
For input i:
If (!iteration) scheduling_end;
Else
  k = IArr_i;
  For (j = 0; j < N_q; j++)
    If (voq_k! = null) selected_i = k;
      req_i^k = 1;
      break;
    End if
    Else
      k++;
      If (k > N_q) k = k % N_q;
    End else
  End for j
  go to GRANT;
End else

GRANT:
  For output j
  k = OArr_j;
  For (i = 0; i < N_i; i++)
    If (req_k^j)
      grant_k = 1;
      grant_indicator_j = 1;
      If (first_iteration)
        OArr_j = k + 1;
        If (OArr_j > N_i) OArr_j = OArr_j% N_i;
      End if
        break;
    End if
    Else
      k++;
      If (k > N_i) k = k % N_i;
    End else
  End for i

For input i
  If ((grant_i == 1) && (selected_i! = null))
    If (first_iteration)
      IArr_i = selected_i + 1;
      If (IArr_i > N_q) IArr_i = IArr_i % N_q;
    End if
    granted_output_i = selected_i;
    selected_i = 0;
    For (j = 1; j < N_q; j++) voq_ij = 0;
End if
Else
  For (j = 1; j < N_q+1; j++)
      If (grant_indicator_j == 1) voq_ij = 0;
    End for j
  End else

iteration - -;
go to REQUEST;
```

IV. Performance Evaluation

Using the OPNET simulator, we examined the performance of the proposed GA scheduling algorithm in terms of the throughput and average queuing delay, and we compared them with those of DRRM. For the simulation, we assumed a 30 × 30 switch, whose scheduling time is within one time slot. We also assumed that a fixed-size time slot accommodates several Ethernet packets. When inputs receive grants, they send Ethernet packets to their destination for the time slot. We assigned a 1% overhead as the guard time at the boundary of the time slots. VOQ sizes of 3 TS, 10 TS, 15 TS, 30 TS, and infinite are considered. For the burst traffic, we used the on-off traffic model of which the both on and off periods are exponentially distributed. The packet length is a variable that follows the distribution of the IP packet length [26]. Packets were generated only during the on-period in the traffic model. The Ethernet packets that are generated during a single on-period had the same destination, each of which was generated with a uniform distribution. For the performance evaluation, the throughput (*Th*) and queuing delay (*d*) are defined as follows.

$$Th = \frac{D_{\text{tx}}}{D_{\text{gen}}}, d = t_{\text{a}} - t_{\text{d}},$$

where D_{tx}, D_{gen}, t_{a}, and t_{d} are the transmitted data, generated data, packet arrival time in the queue, and packet departure time from the queue, respectively.

Figure 4 shows the throughput as a function of the offered load when the VOQ size is infinite. The throughput starts to decrease as the offered load exceeds

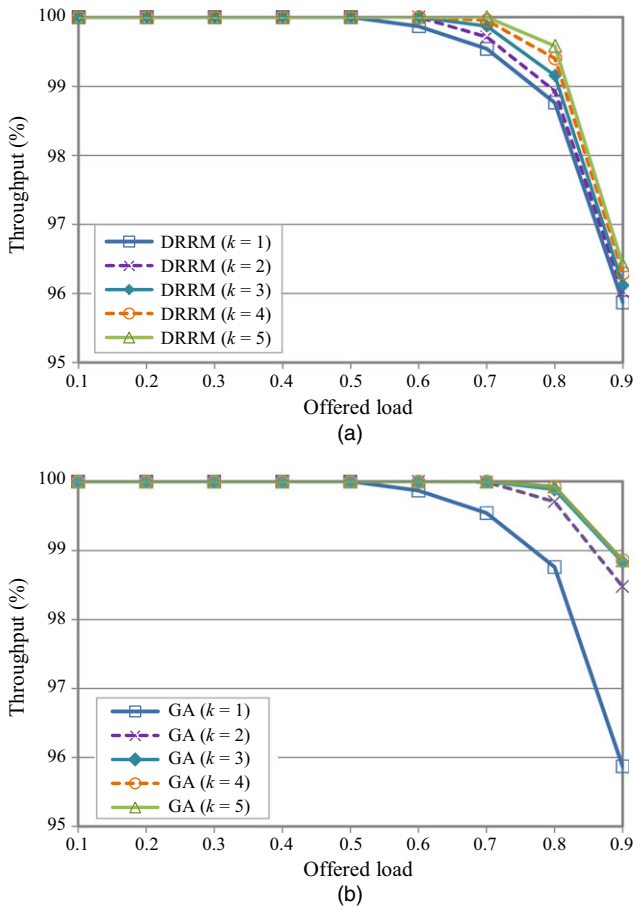

Fig. 4. Comparison of throughput for (a) DRRM and (b) GA.

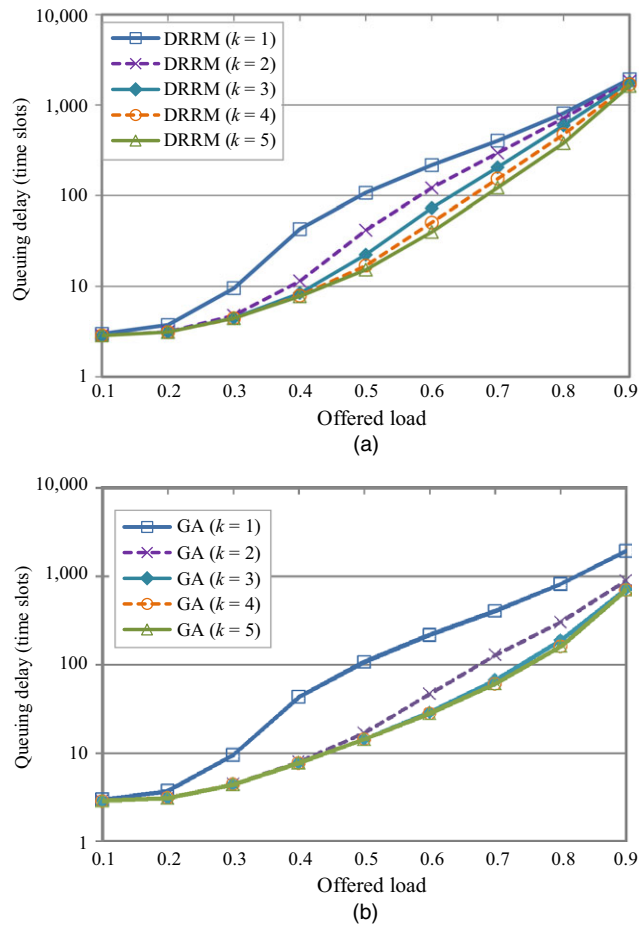

Fig. 5. Comparison of average queuing delay for (a) DRRM and (b) GA.

0.5. When the offered load exceeds 0.5, the throughput increases as the number of iterations increases. The DRRM result is improved even when the number of iterations is increased to 5. However, in the GA case, the result converges within three iterations, and outperforms the DRRM result with five iterations.

Figure 5 shows the queuing delay as a function of the offered load when the VOQ size is infinite. The queuing delay was decreased as the number of iterations was increased. The proposed GA algorithm provided a better performance than DRRM. Figure 5 also shows that GA with three iterations achieved less queuing delay than DRRM with five iterations. When the offered load was more than 0.7, the GA scheme reduced the delay to less than half compared to DRRM.

Figure 6 shows the throughput of the GA algorithm with three iterations for a VOQ size ranging from 3 TS to 30 TS. The throughput decreased as the VOQ size decreased because the packets were dropped when the VOQ for its destination was full. When the offered load was 0.9, the throughput was 65%, 85%, 92%, and 96%, for VOQ sizes of 3 TS, 10 TS, 15 TS, and 30 TS, respectively.

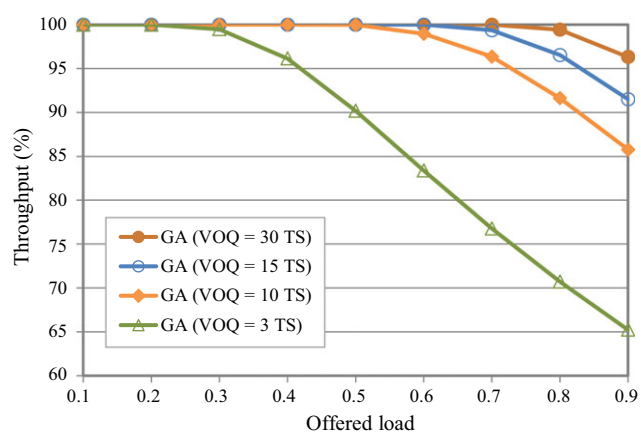

Fig. 6. VOQ size dependence of throughput with GA (k = 3).

Figure 7 shows the average queuing delay of the switched packets as a function of the offered load when the GA algorithm was applied. As the offered load increased, the queuing delay increased. The delay decreased when the size of VOQ was smaller because more packets were dropped. The effect of the VOQ size

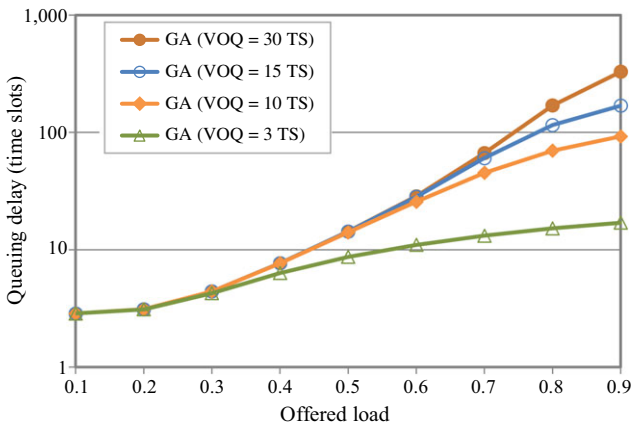

Fig. 7. VOQ size dependence of queuing delay with GA (k = 3).

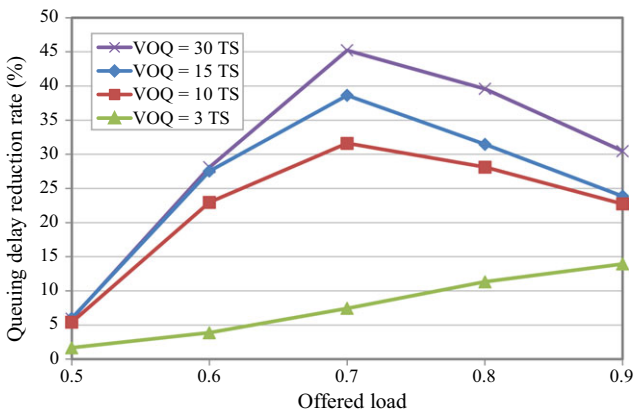

Fig. 8. VOQ size dependence of queuing delay reduction rate when DRRM was replaced by GA.

on the delay was negligible when the offered load was < 0.4.

Figure 8 shows the reduction rate of the delay of the GA algorithm over DRRM with five iterations when the offered load ranged from 0.5 to 0.9 with a finite VOQ size. When the VOQ size was 3 TS, the delay reduction rate of GA over DRRM was < 15%. However, for a VOQ size of 30 TS, the GA algorithm showed a maximum delay reduction of 45% over DRRM. For VOQ sizes of 15 TS and 10 TS, maximum delay reductions of approximately 38% and 31% were shown, respectively. When the offered load was 0.7, the GA provided approximately 7% to 45% less delay than DRRM.

Figure 9 shows the throughput of the GA and DRRM when the VOQ size was 15 TS. When the offered load was lower than 0.5, the throughput was maintained at 100%, and it decreased as the offered load increased. When the offered load was 0.6, DRRM needed five iterations to achieve a 100% throughput. However, GA reached the same throughput with two iterations.

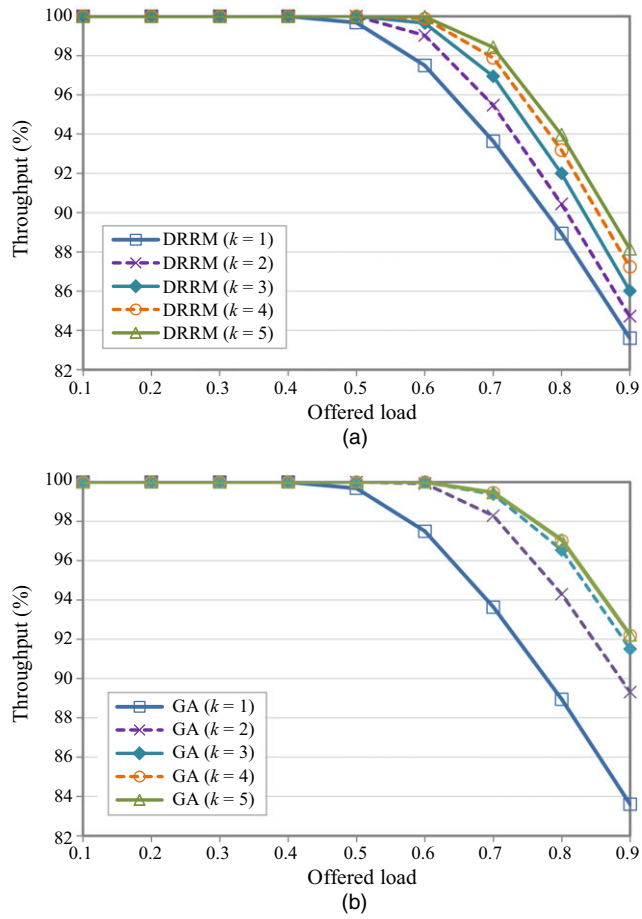

Fig. 9. Throughput of (a) DRRM and (b) GA (VOQ = 15 TS).

Figure 10 shows the average queuing delay of switched packets when the VOQ size was 15 TS. As the number of iterations increased, the queuing delay became less. When the iterations were performed five times (k = 5), both algorithms, GA and DRRM, showed a similar delay at an offered load of 0.5 or less. However, GA had a lower delay than DRRM as the offered load increased. When the offered load was 0.9, the GA showed a lower delay of about 100 TS than DRRM. Even with two iterations, GA showed a lower delay than DRRM with five iterations.

V. Conclusion

We proposed a two-phase GA scheduling algorithm that efficiently provides higher throughput and lower latency with a smaller number of iterations than DRRM. Our proposed algorithm uses the output granted indication for an efficient output selection in subsequent iterations. We compared the performance of the throughput and queuing delay while varying the offered load and VOQ size using the OPNET

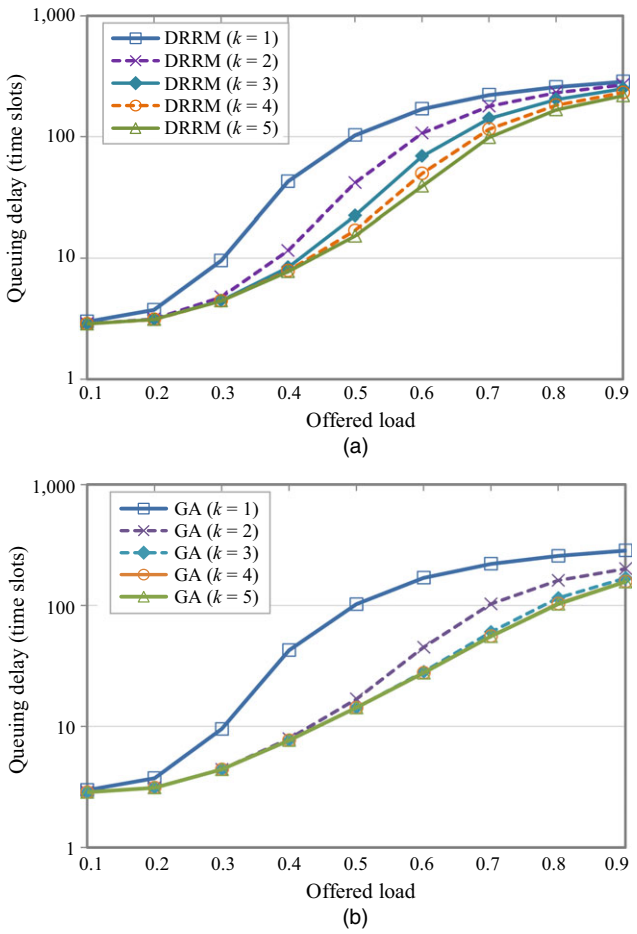

Fig. 10. Average queuing delay of (a) DRRM and (b) GA (VOQ = 15 TS).

simulator. The simulation results showed that the GA provides a maximum 3% higher throughput and 31% less queuing delay than DRRM for a 30 × 30 switch.

Acknowledgements

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00573, Development of Data Center Optical Networking Core Technologies for Photonic Frame-based Packet Switching).

References

[1] A. Bach, "High Speed Networking and the Race to Zero," in *Proc. IEEE Symp. High Perform. Interconnects*, New York, USA, Aug. 25–27, 2009, p. 15.

[2] Juniper Networks, *Network Fabrics for the Modern Data Center*, White Paper, Juniper Networks, Inc., 2010.

[3] S. Liu, Q. Cheng, M.R. Madarbux, A. Wonfor, R.V. Penty, I.H. White, and P.H. Watts, "Low Latency Optical Switch for High Performance Computing with Minimized Processor Energy Load [invited]," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 3, Mar. 2015, pp. A498–A510.

[4] K. Chen et al., "OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility," *J. IEEE/ACM Trans. Netw. (TON)*, vol. 22, no. 2, Apr. 2014, pp. 498–511.

[5] A. Singh et al., "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, Oct. 2015, pp. 183–197.

[6] L. Liu, Z. Zhang, and Y. Yang, "Packet Scheduling in a Low-Latency Optical Interconnect with Electronic Buffers," *J. Lightw. Technol.*, vol. 30, no. 12, 2012, pp. 1869–1881.

[7] I. Iliadis and C. Minkenberg, "Performance of a Speculative Transmission Scheme for Scheduling-Latency Reduction," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, Feb. 2008, pp. 182–195.

[8] N. Farrington et al., "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM 2010 Conf. Data Commun.*, New Delhi, India, Aug. 30–Sept. 3, 2010, pp. 339–350.

[9] K. Naxhimoto et al., "High-Speed PLZT Optical Switches for Burst and Packet Switching," *Proc. Int. Conf. Broadband Netw.*, Boston, MA, USA, Oct. 7, 2005, pp. 1118–1123.

[10] M. Fiorani, S. Aleksic, and M. Casoni, "Hybrid Optical Switching for Data Center Networks," *J. Electr. Comput. Eng.*, vol. 2014, Jan. 2014, pp. 1–11.

[11] D. Shah and M. Kopikare, "Delay Bounds for Approximate Maximum Weight Matching Algorithms for Input Queued Switches," in *Proc. IEEE INFOCOM'02*, New York, USA, June 23–27, 2002, pp. 1024–1031.

[12] H.J. Chao and B. Liu, *High Performance Switches and Routers*, Hoboken, NJ, USA: John Wiley & Sons Inc, 2007.

[13] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in *Proc. IEEE INFOCOM '96*, San Francisco, CA, USA, Mar. 24–28, 1996, pp. 296–302.

[14] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm for Achieving 100% Throughput in Input-Queued Switches," in *Proc. INFOCOM '98*, San Francisco, CA, USA, Mar. 29–Apr. 2, 1998, pp. 792–799.

[15] Y. Tamir and G. Frazier, "High Performance Multi-queue Buffers for VLSI Communication Switches," in *Proc. Annu. Symp. Comput. Arch.*, Honolulu, HI, USA, May 30–June 2, 1988, pp. 343–354.

[16] S. Li and N. Ansari, "Scheduling Input-Queued ATM Switches with QoS Features," *Proc. Int. Conf. Comput.*

Commun. Netw., Lafayette, LA, USA, Oct. 15, 1998, pp. 107–112.

[17] D. Cavendish, M. Goudreau, and A. Ishii, "On the Fairness of Scheduling Algorithms for Input-Queued Switches," *Teletraffic Sci. Eng.*, vol. 4, Dec. 2001, pp. 829–841.

[18] S. Li, J.-G. Chen, and N. Ansari, "Fair Queueing for Input-Buffered Switches With Back Pressure," *IEEE Int. Conf. ATM*, Colmar, France, June 22–24, 1998, pp. 252–259.

[19] T.E. Anderson, S.S. Owicki, J.B. Saxe, and C.P. Thacker, "High Speed Switch Scheduling for Local Area Networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, Nov. 1993, pp. 319–352.

[20] N. McKeown, P. Varaiya, and J. Warland, "Scheduling Cells in an Input-Queued Switch," *Electron. Lett.*, vol. 29, no. 25, Dec. 1993, pp. 2174–2175.

[21] D.N. Serpanos and P.I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Multiple Input Queues," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Isreal, Mar. 26–30, 2000, pp. 548–554.

[22] N. McKeown, "The iSlip Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, Apr. 1999, pp. 188–200.

[23] H. Chao and J. Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," in *Proc. IEEE ATM Workshop*, Fairfax, VA, USA, May 26–29, 1998, pp. 11–16.

[24] H.J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," *IEEE Commun. Mag.*, vol. 38, no. 12, Dec. 2000, pp. 78–84.

[25] C. Minkenberg, F. Abel, P. Muller, R. Krishnamurthy, M. Gusat, and B.R. Hemenway, "Control Path Implementation for a Low-Latency Optical HPC Switch," in *Proc. Symp. High Perform. Interconnects*, Stanford, CA, USA, Aug. 17–19, 2005, pp. 29–35.

[26] S. McCreary and K.C. Claffy, "Trends in Wide Area IP Traffic Patterns," Cooperative Association for Internet Data Analysis, USA, Accessed 2017. http://www.caida.org

Kyeong-Eun Han received her B.S., M.S., and Ph.D. degrees in computer engineering from Chonbuk National University, Jeonju, Rep. of Korea, in 2001, 2003, and 2008, respectively. Her Ph.D. thesis focused on WDM-PON for access networks. In 2003, she was nominated as an Up-and-Coming Young Researcher by the Korea Science and Engineering Foundation. Since 2008, she has been with the ETRI, Daejeon, Rep. of Korea, where she is currently a Senior Researcher. Her research topics include energy efficiency Ethernet, all-optical data center networks, and optical communication networks.

Jongtae Song received his B.S. degree in electronics and electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 1990, the M.S. degree in Electrical Engineering from the University of Southern California, Los Angeles, USA, in 1994, and the Ph.D. degree in electrical engineering from New York University, NJ, USA, in 1998. He worked for Bell Labs Lucent Technologies, Holmdel, New Jersey, from 1998 to 2001, and for several startup companies (including Coree Networks and Parama Networks) from 2001 to 2004. Since 2004, he has worked as a principal research staff at the ETRI, Daejeon, Rep. of Korea. His research interests include all-optical switching systems, network architectures for 5G networks, and all-optical data center networks.

Dae-Ub Kim is a principal researcher at the ETRI, Daejeon, Rep. of Korea. He earned his Ph.D. degree in information and communication engineering from Chungnam National University, Daejeon, Rep. of Korea, in 2017. He received his M.S. degree in information and communication engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 2001. Since joining the Electronics and Telecommunication Research Institute in 2001, his work has focused on next-generation networks, wireless backhaul networks, carrier class Ethernet, OTN and MPLS-TP technology, focusing especially on protection standardization activities in the ITU-T. He is currently focusing on optimal control for optical switches supporting both packet and circuit connections in intra- or inter-datacenter networks.

JiWook Youn received the B.S. and M.S. degrees in electronic engineering from KyungHee University, Seoul, Rep. of Korea, in 1997 and 1999, respectively, and the Ph.D. degree in Electronic Engineering from Chungnam University, Daejeon, Rep. of Korea, in 2007. Since 1999, he has worked at the ETRI, Daejeon, Rep. of Korea. His current research interests include all-optical switching systems for data center and optical transport networks.

Chansung Park received his B.S. degree in computer engineering from Inje University, Gimhae, Rep. of Korea, in 2012. Since 2012, he has been affiliated with the ETRI, Daejeon, Rep. of Korea, as a Researcher. He worked on OAM and Protection in OTN and PCE for multi-layer networks. Currently, he is working on software-defined networks, packet scheduling for data centers, and optical communication networks.

Kwangjoon Kim received the B.S. and M.S. degrees in physics from Seoul National University, Rep. of Korea, and the Ph.D. degree in physics from the Ohio State University, Columbus, OH, USA, in 1981, 1983, and 1993, respectively. He joined the ETRI, Daejeon, Rep. of Korea, in 1984, and worked on HF communications until he enrolled in the Ph.D. program at the Ohio State University, where he worked on various linear and nonlinear optical behaviors of conducting polymers. He rejoined ETRI and worked on optical semiconductor devices with quantum wells. His current research focus is on WDM optical communication systems and high-speed optical transmission, including all-optical switching networks.

https://doi.org/10.4218/etrij.2017-0057