Desynchronized Two-Dimensional Round-Robin Scheduler for Input Buffered Switches[†]

Seung Yeob Nam* and Dan Keun Sung*
*Dept. of EECS, KAIST, KOREA
E-mail: synam@cnr.kaist.ac.kr, dksung@ee.kaist.ac.kr

Abstract—**We propose a new arbitration algorithm, called the desynchronized two-dimensional round-robin (D2DRR), for input queued packet switches in which each input port maintains a separate logical queue for each output. D2DRR is an enhanced version of 2DRR, and thus improves fairness over 2DRR without a starvation problem. Iterative maximal matching schemes including iSLIP yield better throughput with more iterations. However, since many iterations require much time for arbitration, it is not desirable for high speed switching. Thus, D2DRR attempts to resolve contentions with less iterations, while yielding high throughput. The proposed arbitration algorithm is compared in terms of average cell latency with conventional algorithms, especially iSLIP and wrapped wave front arbitration (WWFA).**

I. Introduction

Explosively increasing internet traffic has led to a greater demand for high-speed switches and routers with a throughput of higher than 1-Tbit/s [1]. Crossbar switching fabrics are widely adopted because of their non-blocking capability, simplicity, and market availability. Since it has been shown possible to increase the throughput of input-buffered switches from 58.6% to 100% using a virtual output queuing (VOQ) scheme [2], input queuing schemes are currently being adopted in many high-speed switching systems [3][4].

Even if input buffered switches maintain a virtual output queue at each input port, there remain contention problems at the input and output ports. In order to solve these contentions and to increase the utilization of the switch, proper arbitration algorithms are used. Maximum matching algorithms have been proposed to achieve 100% throughput. However, such a high complexity [2] makes implementation difficult for high-speed systems. Maximal matching schemes, such as parallel iterative matching (PIM) [5], round-robin matching (RRM), iSLIP [6], FCFS in round-robin matching (FIRM) [7], wave front arbitration (WFA) [8], and two-dimensional round-robin (2DRR) [9], have been considered as an alternative to maximum matching schemes.

iSLIP provides 100% throughput for uniform, independent traffic and yields better performance as the number of iterations increases until it converges in about $\log_2 N$ iterations for an $N \times N$ switch. However, since it takes more time to perform multiple iterations, the number of iterations becomes a bottleneck as the switching speed increases.

In this paper we propose a new arbitration algorithm, called the desynchronized two-dimensional round-robin (D2DRR), which yields good performance with just a single iteration. D2DRR is similar to the 2DRR algorithm [9]. However, it provides a better fairness property than 2DRR. We propose two types of D2DRR algorithms. The first one is D2DRR with a single matching which does not allow a nonempty VOQ to remain unserved indefinitely. The second one is D2DRR with double matching which inherits the *starvation-free* property of a single matching algorithm and improves delay performance by using the bandwidth released free after the first matching.

This paper is organized as follows: In Section II, we present a basic D2DRR algorithm with a single matching and describe its properties. In Section III, we describe an enhanced D2DRR algorithm with double matching. In Section IV, we evaluate the performance of the proposed arbitration algorithms by simulation. Finally, we present conclusions in Section V.

II. Basic D2DRR Arbitration Algorithm

The 2DRR algorithm was introduced as a two-dimensional generalization of the one-dimensional round-robin scheme [9]. We enhance the 2DRR from the aspect of fairness. In this section, we describe a basic D2DRR scheduling algorithm, while in a later section, we will describe an enhanced version of the D2DRR.

In this paper we consider an $N \times N$ input queued switch which uses virtual output queueing (VOQ), where a separate queue is maintained at each input port for each output. $Q_{i,j}$ denotes the virtual output queue that stores the cells passing from the i-th input port to the j-th output port.

D2DRR with a single matching consists of three steps: Request, Matching, and Pointer update. All the three steps are performed within a time slot, where a time slot is the time between packet arrivals at input ports.

Step 1: Request

Let R denote the request matrix for an $N \times N$ input buffered switch, then it can be expressed as

$$R = \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,N} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,N} \\ \vdots & & \ddots & \vdots \\ R_{N,1} & R_{N,2} & \cdots & R_{N,N} \end{bmatrix} \quad (1)$$

where $R_{i,j}$ takes one value among 0, 2, and 4. If $Q_{i,j}$ is non-empty, the value of $R_{i,j}$ becomes 2 or 4. Otherwise, it becomes 0 or 2.

In the D2DRR algorithm, time is logically partitioned into variable-length intervals called the frames. Each frame is determined independently for each column of request matrix. During a frame, all non-empty VOQs belonging to the same column of the request matrix are served just once. At the beginning of a frame, the values of all $R_{i,j}{}'s$ corresponding to non-empty VOQs are 4 and the values of the other $R_{i,j}{}'s$ are zero. When $Q_{i,j}$ acquires a grant to transmit a cell into the switch fabric, the value of $R_{i,j}$ is changed into 2 and the VOQ can not obtain any more grant until the current frame is over since the grant is given to the VOQ whose request value is equal to 4. If a new cell arrives at an empty VOQ $Q_{i,j}$ during a frame, the value of $R_{i,j}$ is changed from 0 to

[†] This study was supported in part by the Ministry of Information and Communications.

4. However, if a new cell arrives at the $Q_{i',j'}$ whose request value is 2, the value of $R_{i',j'}$ is not changed. If all non-empty VOQs for column j are served once, all the request values are reset since the current frame is over and the next frame begins with new request values.

Step 2: Matching

The D2DRR scheduler allocates one pointer for each column in order to resolve output contentions for each output. This is a major difference from the WFA [8] or the 2DRR [9] which uses a fixed set of pointer array. However, the pointer array for D2DRR can have any pattern. Let $CP[j]$ be the pointer allocated for column j. Thus, there are N pointers in total. Initially, all the values of N pointers are different from each other, and the values are ever kept different thereafter, i.e., the pointers move desynchronized with each other. Thus, there is only one pointer per each row and there is only one pointer per each column. An output j chooses the request that has a value equal to 4 and appears next in a round-robin schedule starting from the position of $CP[j]$.

Step 3: Pointer Update

Pointer update is very important because it can affect the throughput of the switch and it also determines whether the algorithm suffers from a starvation problem or not. In the D2DRR algorithm, the pointer update stage is further divided into three steps.

First, if there is at least one request for a column, but no matching is made for the column, a pointer value is incremented (modulo N) by one for the column.

Second, pointers for the columns where at least one matching is made are updated. Pointers for these columns are incremented (modulo N) to one location beyond the granted input. However, if another pointer from other column has already occupied the input, the later one can not be fixed there. Such a pointer should be incremented (modulo N) one by one until it is allocated to the free input (or row) where no previous pointer allocation is made. Since the initial points (i.e., the granted inputs) of the pointers are different, all pointers can be incremented concurrently.

Third, pointers for the columns where there is no request are updated. The procedure is almost the same as the second case. Each pointer is incremented (modulo N) one by one from the initial position until it is allocated to the free input.

We compare the D2DRR with the WFA and the 2DRR from the aspect of fairness. Fig. 1 shows a vertical line-shaped traffic pattern. We assume that requests are always present for the three input/output pairs that are indicated with 1's in Fig. 1. If the line-shaped pattern includes all $N = 8$ of the input/output pairs in the third column, then each one has a throughput of $1/N = 0.125$ for any arbitration algorithm. However, in the reduced length line of Fig. 1 WFA yields a throughput of $(0.125, 0.125, 0.75)$ or $(0.75, 0.125, 0.125)$ for the pairs $((4,3),(5,3),(6,3))$, respectively. Even the basic 2DRR or the enhanced 2DRR algorithm does not show a throughput of $(1/3, 1/3, 1/3)$ because they use only fixed pointer patterns [9].

On the other hand, in D2DRR the pattern of pointer array is not fixed and the pointer for each column can move more freely. D2DRR partitions time into variable-length intervals called the frames. During a frame, all non-empty VOQs belonging to the same column of the request matrix are served just once. Thus, all three input/output pairs in Fig. 1 have the same throughput of $1/3$, and furthermore, all non-empty VOQs contending for the same output always receive fair services.

Output

Input	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	0	0	1	0	0	0	0	0
6	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Fig. 1. Request pattern for line-shaped traffic case

We now consider a *starvation-free* property of the D2DRR algorithm.

Lemma 1 If there is any request for a column, at least one request is granted for transmission for N time slots.

Proof) Let us assume that no request is granted from a column with at least one request for N time slots. Then, according to the *pointer update* rule this column is given the highest priority and the pointer for the column is incremented (modulo N) by one for each time slot. Since no request is granted during N time slots, the pointer has passed through every input. The condition there is at least one request for the column implies at least one input has a request at the column and at least one match should have occurred for N time slots. Therefore, proof is done by contradiction. □

Theorem 1 The duration of one frame does not exceed N^2 time slots.

Proof) Since we consider the maximum value of the duration of a frame, it is sufficient to consider only one generic frame. Fig. 2 shows the sample of a frame. As shown in Fig. 2, time is counted from the beginning of a frame in time slots.

Let $R(t)$ be the number of requests coming to the head of the VOQs corresponding to the selected column (or output) during t time slots of a frame. $R(t)$ does not count the requests coming to the head of the VOQ that is already served in the frame, i.e., $R(t)$ counts the requests for each VOQ at most once in a frame. Let $G(t)$ be the number of grants given to the selected column of the request matrix during t time slots of a frame. Then, the number of the requests remaining in the column without being served during the time interval $[0, t]$, $U(t)$ can be expressed as

$$U(t) = R(t) - G(t). \tag{2}$$

We can easily know the following relation from the definition of $R(t)$:

$$R(t) \leq N. \tag{3}$$

If we define S_i as the time interval between the $(i-1)$-th and i-th grants given to the column, $G(t)$ can be expressed as

$$G(t) = \max\left\{ i : \sum_{j=1}^{i} S_j \leq t \right\}. \tag{4}$$

If the frame ends before N^2 time slots, the proof is over. Thus, we assume the frame does not end before N^2 time

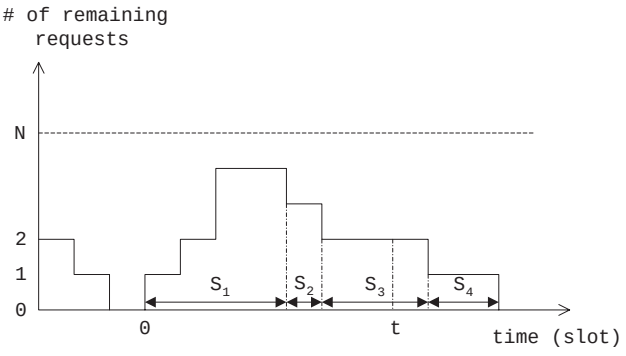

Fig. 2. Change of the remaining requests over time

slots. Then, since $S_i \leq N$ holds for each $i = 1, 2, \ldots, N$ by lemma 1, the following relation also holds:

$$\sum_{j=1}^{N} S_j \leq N^2. \tag{5}$$

From (4) and (5), the following relation can be obtained:

$$G(N^2) = \max\left\{ i : \sum_{j=1}^{i} S_j \leq N^2 \right\} = N. \tag{6}$$

Combining (2), (3), and (6) yields the following equation:

$$U(N^2) = R(N^2) - G(N^2) \leq N - N = 0 \tag{7}$$

Q.E.D. □

This theorem states that any frame does not last longer than N^2 time slots. Since a frame ends only after every non-empty VOQ for the column is served once, the following result can be obtained easily.

Corollary 1 Any head-of-line request can be served within $2N^2$ time slots.

Proof) trivial □

This corollary implies that no request is starved by the D2DRR algorithm.

III. Enhanced D2DRR Arbitration Algorithm

Thus far, we have considered the operation of D2DRR with a single matching (1-D2DRR). We now examine an enhanced D2DRR algorithm that performs two matchings within a time slot (2-D2DRR).

The performance of iSLIP improves as the number of iterations increases (up to about $\log_2 N$, for an $N \times N$ switch). The performance of D2DRR improves significantly with one more matching. A single iteration of 1-D2DRR consists of 3 steps: Request, Matching, and Pointer Update. The operation of 2-D2DRR requires one more step, and thus, a single iteration of 2-D2DRR consists of 4 steps: Request, first Matching, second Matching, and Pointer Update.

The operation of 2-D2DRR follows the basic operation of 1-D2DRR. We examine the 2-D2DRR algorithm based on the difference with 1-D2DRR.

Step 1: Request

The structure of the request matrix is the same as (1), but the value of its element is different. $R_{i,j}$ is set to 0, 1, 2, or 4 depending on the situation. If $Q_{i,j}$ is non-empty, the value of $R_{i,j}$ is 2 or 4. Otherwise, it is 0 or 1. The definition of a frame is similar. At the beginning of a frame, the values of all ${R_{i,j}}'s$ corresponding to non-empty VOQs are set to 4 and the values of the other ${R_{i,j}}'s$ are set to zero. When $Q_{i,j}$ acquires a grant to transmit a cell into the switch fabric, the value of $R_{i,j}$ is changed into 2 if $Q_{i,j}$ has any remaining cell and it is changed to 1 if $Q_{i,j}$ has no cell to transmit. If a new cell arrives at the $Q_{i',j'}$ whose request value is 1 or 2, the value of $R_{i',j'}$ is changed into 2. If all non-empty VOQs for column j are served at least once, all the request values are reset since the current frame is over and the next frame begins with new request values.

Step 2: First Matching

The second step of 2-D2DRR is the same as the second step of 1-iteration algorithm, i.e., a grant is given to the request whose value is equal to 4.

Step 3: Second Matching

At the first matching stage, a grant is given only to the request with a value of 4 in order to support fairness and to avoid a starvation problem. At the step of second matching, the grant is given only to the request whose value is equal to 2 in order to increase the throughput irrespective of the constraints of fairness and starvation-free.

Step 4: Pointer Update

Step 4 neglects the matching made at Step 3, and the pointer update is performed, as shown in the third step of 1-D2DRR. The pointer is updated by considering the match made at the first matching step in order to support fairness and to avoid a starvation problem.

Since the pointer is updated in the same way, the starvation-free property obtained for the 1-D2DRR algorithm also holds for 2-D2DRR. From the aspect of fairness, the throughput provided to each input by the 1-D2DRR is also guaranteed. Thus, minimum throughput is provided fairly to each input. Since the remaining bandwidth is utilized to increase total throughput, the remaining bandwidth may not be distributed evenly.

Since the second matching of 2-D2DRR does not affect the pointer update step, the second matching and the pointer update steps can be performed concurrently. Thus, the arbitration time of 2-D2DRR is the same as that of 1-D2DRR.

IV. Simulation Results

The performance of the proposed arbitration algorithm is evaluated through simulation for an 8×8 input buffered switch. The input and output link rates are 622 Mbps and the length of packets is fixed to that of ATM cell, i.e., 53 bytes. One traffic source is connected to each input port and the destined output ports of generated cells are randomly selected among 8 output ports.

Input traffic models for simulation include random and bursty traffic. For random traffic, cells arrive at each input port according to a Bernoulli process with parameter λ, where λ is the offered load per each input port. Bursty traffic is modeled by an *on-off* arrival process where the *on* and *off* interval lengths are exponentially distributed with different parameters. The source alternately generates a burst of cells followed by an idle period of no cells. During the *on* period cells are generated at the link rate and the destined output ports of the cells belonging to the same *on* period are all identical. The average burst length is set to 32 cells.

The performance of the basic D2DRR (1-D2DRR) and the enhanced D2DRR (2-D2DRR) algorithms are compared with other arbitration algorithms including the iSLIP and the

wrapped wave-front arbitration (WWFA) arbitration algorithms, and are also compared with that of an output buffered switch.

Fig. 3 compares the average delay distributions obtained from several different algorithms under a Bernoulli traffic load. We can observe that the 1-D2DRR algorithm yields low latency compared with the SLIP algorithm with a single iteration. This is because in case of SLIP grant arbiters suffer from blocking. The pointer of grant arbiter of iSLIP is incremented if, and only if, the grant is accept. Thus, if the highest priority element at the output is not accepted, the grant can not be allowed to other lower priority element because the position of the pointer is not changed. However, in the D2DRR algorithm, if the highest priority element can not be granted because the input has already selected another element for different output, the grant can be given to other lower priority element. And fairness is guaranteed by another mechanism. Since D2DRR can give more grants than iSLIP, it yields higher throughput.

Since one frame ends only after all non-empty VOQs receive service just once, any non-empty VOQ that is served should wait for the frame to end in order to be served again. The throughput of 1-D2DRR is limited by the above constraint. 2-D2DRR relaxes this constraint by introducing a second matching step which allows the bandwidth left free after the first matching to be used freely. Fig. 3 shows that 2-D2DRR yields slightly better performance than WWFA and 2-SLIP and its performance is close to that of 3-SLIP.

Fig. 4 compares the average cell latency of the proposed D2DRR algorithms with that of the conventional algorithms under a bursty traffic load. The differences among different algorithms seem clear under the worse scenario. However, we can observe a similar trend to Fig. 3. The performance of 2-D2DRR is better than for 2-SLIP and is very close to that of 3-SLIP.

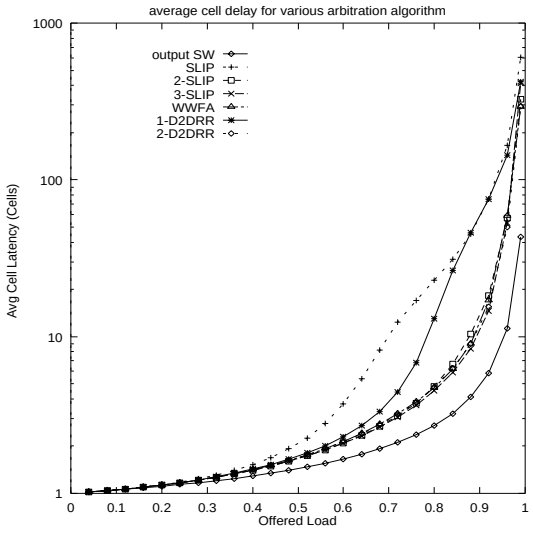

Fig. 3. Comparison of the conventional and the proposed D2DRR arbitration algorithms under random traffic load

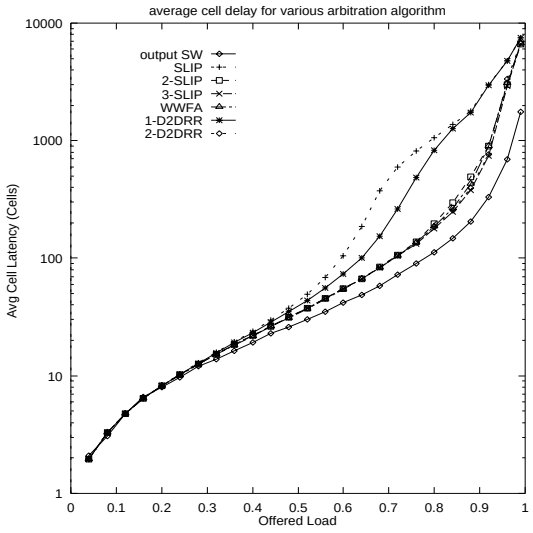

Fig. 4. Comparison of the conventional and the proposed D2DRR arbitration algorithms under bursty traffic load

V. Conclusions

In this paper we proposed two new arbitration algorithms for input buffered switches; a basic D2DRR and an enhanced D2DRR algorithms. The D2DRR algorithm yields high throughput with just a single iteration. The single matching scheme (1-D2DRR) guarantees fairness and avoids the starvation problem, and the double matching scheme (2-D2DRR) improves the throughput by utilizing the bandwidth released after the first matching while preserving fairness and starvation-free properties.

Simulation results show that the basic D2DRR algorithm yields better performance than the SLIP with a single iteration and the enhanced D2DRR algorithm yields the performance close to the maximum throughput of conventional iterative algorithms. Thus, it is more competitive than other iterative algorithms requiring many iterations from the viewpoint of high speed implementation.

References

[1] N. Yamanaka, E. Oki, S. Yasukawa, R. Kawano, and K. Okazaki, "OPTIMA: Scalable, Multi-Stage, 640-Gbit/s ATM Switching System Based on Advanced Electronic and Optical WDM Technologies," *IEICE Trans. Commun.*, vol. E83-B, no. 7, pp. 1488-1496, 2000.

[2] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, pp. 296-302.

[3] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "The tiny tera: A small high-bandwidth packet switch core," *IEEE Micro*, vol. 17, pp. 26-33, Jan.-Feb. 1997.

[4] C. Partridge *et al*, "A 50-Gb/s IP router," *IEEE/ACM Trans. Networking*, vol. 6, pp. 237-248, June 1998.

[5] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[6] Nick McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-201, Apr. 1999.

[7] 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 2000*, vol. 2, pp. 548-555, 2000

[8] Hsin-Chou Chi and Yuval Tamir, "Decomposed Arbiters for Large Crossbars with Multi-Queue Input Buffers," *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 233-238, 1991.

[9] Richard O. LaMaire and N. Serpanos, "Two-Dimensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues," *IEEE/ACM Trans. Networking*, vol. 2, no. 5, Oct. 1994.