The Dual Round Robin Pseudo-grant Matching for high-speed packet Switches

R. Manivasakan, Mounir Hamdi and Danny H.K. Tsang

Abstract—**The Virtual Output Queueing (VoQ) in conjunction with matching algorithms have been proposed to overcome the HOL problem in input queued (IQ) high-speed switches. The Dual Round-Robin Matching (DRRM) scheme, has been shown to achieve good performance while being easy to build to high-speed and scalable switches. In this paper, we present a variant and improved version of the DRRM scheme, termed the Pseudo-grant Dual Round-Robin Matching (PDRRM) algorithm. The PDRRM gives a Pseudo-grant to a needy input to increase the number of matches per time slot. We have evaluated the PDRRM using extensive simulations. Our results will demonstrate that using the "Pseudo-grant" concept results in significant improvement in the performance of DRRM under different networking scenarios. In addition, PDRRM is conjectured to achieve an asymptotic 100% throughput for any arrival pattern.**

I. Introduction

The ubiquitous presence of the multimedia applications and the tremendous popularity of the World Wide Web has dramatically increased the amount of traffic carried over the Internet. As a result, there is a great demand for high-performance switches/routers that are scalable and can be interfaced to very high-speed links (e.g., OC 192). Three different architectures have been identified for these high-speed switches, depending on where packet buffering takes place: (i) output queueing (OQ), (ii) input queueing (IQ) and (iii) combined input-output queueing (CIOQ). The Virtual Output Queueing (VoQ) in conjunction with matching algorithms have been proposed to overcome the head-of-line (HOL) problem in IQ switches. It has been shown that *maximum* matching algorithms can achieve a 100% throughput. However, these algorithms are not practical to implement in hardware due to their high complexity and may not guarantee fairness and quality of service. To overcome this problem, matching algorithms that approximate maximum matching algorithms, known as *maximal* matching algorithms have been proposed instead namely, Parallel Iterative Matching (PIM) [1], Round Robin Matching (RRM) [2], Dual Round Robin Matching (DRRM) [3], iSLIP [2] and FIRM [4]. The operation of PIM requires random choices which are costly in hardware and 'slow' in terms of performance. DRRM and iSLIP provide more attractive solutions, since they replace the random choices with round-robin schemes and can be thus easily implemented efficiently. However, among these algorithms, only iSLIP achieves a 100 % throughput for a 100 % load; RRM suffers from blocking which limits its saturation throughput to less than 65 %. On the other hand, DRRM achieves a saturation throughput of 100 % with lower delay and provides better fairness than iSLIP. In this paper, we introduce a new distributed scheduling algorithm called Pseudo-grant - DRRM (PDRRM) which incorporates the notion of *pseudo-grant* in addition to the Exhaustive service feature of Exhaustive service DRRM (EDRRM) scheduling algorithm. As a result, it improves upon the performance of DRRM and EDRRM, while still maintaining their simplicity of ease of hardware implementation. The rest of the paper is organized as follows. In Section 2, we describe the DRRM class of algorithms. In Section 3 the Pseudo-grant DRRM (PDRRM) scheme is described in detail. In Section 4 we present the simulation results pertaining to the throughput, mean cell delay performance of the PDRRM scheme. Section 5 concludes the paper.

This work was supported in part by the post-doctoral matching fund under Prof. Mounir Hamdi. Corresponding author: Prof. Mounir Hamdi email: hamdi@cs.ust.hk. Prof. Hamdi is with the Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong. Dr. R. Manivasakan and Prof. Danny H. K. Tsang are with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong email: eemani@ee.ust.hk, eetsang@ust.hk

II. DRRM Class of Algorithms

Notation: We use the following notation which would make the description of our algorithm convenient in later sections. We consider an N x N switch with N inputs, $I[0]...I[N-1]$ and N outputs, $O[0]...O[N-1]$. Every input $I[i]$ maintains the following state information: (i) The matrix Rq with entry $Rq_i[j]$ where $Rq_i[j] = 1$ if $I[i]$ has a request for $O[j]$ (and 0 otherwise). (ii) The matrix Gd where $Gd_{i,j} = 1$ if $I[i]$ receives a grant from $O[j]$ (0 otherwise). (iii) A vector A where $A_i = j$ if $I[i]$ accepts the Grant from $O[j]$ (-1 if no output is accepted).

Similarly, each output $O[j]$ maintains the following state information: (i) The matrix Rd with entry $Rd_{i,j}$ where $Rd_{i,j} = 1$ if $O[j]$ receives a request from $I[i]$ (and 0 otherwise). (ii) a vector G where $G_j = i$ if $O[j]$ accepts the request from $I[i]$ (-1 if no input is granted). (iii) The vector Ad_j where $Ad_j = 1$ if the Grant from $O[j]$ is accepted (0 otherwise). We use two variables (per output) to describe the process of examining the candidate inputs in some way (e.g. randomly or by round-robin) while issuing Grant by an output to input: g_k denotes which input will be examined first (to issue a Grant) during a step, while $g_k^{'}$ shows the input to be examined first in the consecutive step. Let a_i and $a_i^{'}$ denote the choice of which grant to accept at an input is made with a similar process. Similarly, we need also h_i and $h_i^{'}$ to denote the choice of which request to be issued (corresponding to more than one VoQ nonempty in the two-step algorithms like DRRM). Also in our algorithm we need one more variable RJ_k which denotes the input (tracked by the reject pointer at the output arbiter), to which an output k will issue a *pseudo-grant* under certain conditions (see Section 3).

The DRRM algorithm: Most of the proposed scheduling algorithms for high-speed networks are three step algorithms which perform the following operations in each iteration: (i) Inputs broadcast their requests to the outputs, (ii) Each output selects one request independently and issues a Grant to it and (iii) Each input selects one Grant to accept, since it may simultaneously receive several Grant signals. In sharp contrast, the DRRM algorithm (and its variant EDRRM) has only two steps in an iteration: *Step 1: Request.* Each input sends an output request corresponding to the first nonempty VoQ in a fixed round-robin order starting from the current position of the pointer. The pointer remains at the nonempty VoQ if the selected output is not granted in step 2. The pointer of the input arbiter is incremented by one location beyond the selected output if and only if the request is granted in step 2. *Step 2: Grant.* If an output receives one or more requests it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the pointer. The output notifies each requesting input whether or not its request was granted. The pointer of the output arbiter is incremented to one location beyond the granted input. If there are no requests, the pointer remains where it is.

In the DRRM scheme described above, when an input and an output are matched, only one cell will be transferred from the input to the matched output. After that both the input and the output will increment their pointers by one and in the next time slot this input-output pair will have the lowest priority to get matched to each other. This behavior is similar to the *limited service policy* in a polling system. In order to improve on DRRM's performance under non-uniform traffic, the following modification is done [5]: whenever an input is matched to an output, all the cells in the corresponding VoQ will be transferred in the following time slots before any other VoQ of the same input can be served. This is similar to the *exhaustive service policy* in polling systems and is implemented in the Exhaustive service DRRM (EDRRM) scheme proposed in [5].

The EDRRM algorithm: A detailed description of the two-step EDRRM algorithm follows: *Step 1: Request.* Each input moves its pointer to the first nonempty VoQ in a fixed round-robin order starting from the current position of the pointer and sends a request to the output corresponding to the VoQ. The pointer of the input arbiter is incremented by one location beyond the selected output if the request is not granted in step 2. The pointer is also incremented by one location if the request is granted and after this cell is served the corresponding VoQ becomes empty. Otherwise, the pointer remains at that (nonempty) VoQ. *Step 2: Grant.* If an output receives one or more requests it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the pointer. The output notifies each requesting input whether or not its request was granted. The pointer of the output arbiter remains at the granted input. If there are no requests, the pointer remains where it is. In this paper, we further modify the EDRRM algorithm by incorporating the pseudo-grant facility.

III. Pseudo-Grant DRRM Algorithm

DRRM essentially reduces the number of steps in an iterative cycle in the parallel iterative algorithm by avoiding the multiple request issue from a *single* input (whenever more than one VoQ is backlogged) by selecting a request according to a pointer in the input arbiter. The disadvantage incurred due to this approach is that the final number of matchings achieved in a *single* slot may be less as compared to that in PIM and iSLIP class of algorithms (which are basically three-step algorithms). Moreover, the multiple iterations in DRRM does not increase the number of matching achieved due to its *limited service* feature.

In this paper, we attempt to improve the number of matching in a slot. Before discussing the PDRRM, consider the following scenario: In EDRRM scheme, it might so happen that an output (say, an output i) might go with no requests at all. Also, there might be too

many requests for a *single* output (say output j with $j \neq i$). It might also happen that the input whose request was rejected by output j, has a cell to the above output (output i) which did not get any requests. In an attempt to overcome this limitation (of not being able to achieve the maximum number of matches), we propose the Pseudo-grant DRRM algorithm (PDRRM) wherein the number of matches in a time slot is increased by the *pseudo-grant* mechanism. In Pseudo-grant DRRM algorithm (PDRRM), an output which did not get any request would issue a *pseudo-grant* in the grant phase to that input whose request got rejected in the previous slot by output i. In case if *this* input got a (conventional) grant also, then only the 'grant' gets the priority. If its request is rejected then the *pseudo-grant* will take care of the match. Hence, the two-step algorithm in EDRRM is modified as given below: *Step 1: Request.* Each input sends an output request corresponding to the first nonempty VoQ in a fixed round-robin order, starting from the current position of the pointer. The pointer remains at *that* nonempty VoQ if the selected output is granted in step 2 and the queue *does not* become empty after the cell transfer. The pointer of the input arbiter is incremented by one location beyond the selected output if the request is not granted in *Step 2*. The pointer is also incremented if it is granted *and* the VoQ becomes empty after the cell transfer. *Step 2: Grant.* If an output receives one or more requests, it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the pointer. The output notifies each requesting input whether or not its request was granted. The pointer of the output arbiter is in the *same* location. The output arbiter also maintains another pointer (called 'reject' pointer) which points to the input which was rejected in the previous time slot. This input to which the 'reject' pointer points is the one that appears next in a fixed round-robin schedule starting from the granted input [1]. If there are no requests, then this output throws a *pseudo-grant* to the input pointed by the 'reject' pointer. If the request from *this* input was rejected by the (other) output (to which it had issued request), then the pseudo-grant will be active and the cell will be transferred. In case if the input gets a grant, then this *pseudo-grant* will be ignored. Note that, it is not possible for an input to receive multiple *pseudo-grants* simultaneously. The Pseudo-grant-DRRM algorithm is given below in terms of the notation given in Section II,

[1] In case there is only one request then both pointers will coincide.

/* Main algorithm */

$$
\begin{aligned}
h_i &= h_i^{'} \\
Rq_i[k] &= 1 \quad (if(B_i[k] = 1 \text{ and } h_i = k) \\
&\quad OR \quad (B_i[k] = 1 \text{ and} \\
&\quad B_i[j] = 0 \quad \text{for } h_i \leq j < k) \\
Rd_{i,k} &= Rq_i[k] \\
g_k &= g_k^{'} \\
G_k &= i, \quad if(Rd_{i,k} = 1 \text{ and } g_k = i) \\
&\quad OR \quad (Rd_{i,k} = 1 \text{ and } Rd_{j,k} = 0), \\
&\quad \text{where } g_k \leq j < i(modN) \\
Gd_{i,k} &= 1, if(G_k = i) \\
if(G_k == i) &\quad if(q_i(k) == 1) h_i^{'} = (k+1)(modN); \\
&\quad else \ h_i^{'} = k \\
else &\quad h_i^{'} = (k+1)(modN); \\
g_k^{'} &= G_k;
\end{aligned}
$$

/* Update reject pointer */

$$
\begin{aligned}
&\text{flag} = 0; \quad RJ_k = -1; \\
&\text{for} \quad j = i+1 : N \\
&\qquad if(Rq_j[k] == 1) \\
&\qquad RJ_k = j; \\
&\qquad \text{flag} = 1; \text{break}; \\
&\qquad endif \\
&\text{endfor} \\
if((\text{flag} == 0) \&\& (i \neq 0)) &\quad \text{for } j = 0 : i-1 \\
&\quad if(Rq_j[k] == 1) \\
&\quad RJ_k = j; \\
&\quad endif \\
&\quad \text{break}; \\
&\quad \text{endfor} \\
&\text{endif}
\end{aligned}
$$

/* pseudo-grant loop */

$$
\begin{aligned}
if(Rq_l[k] = 0) &\quad \forall \ l, \quad \text{then} \\
&\quad if(RJ_k = m) \quad PG_k = m \\
&\quad endif \\
&\quad if((G_n \ != \ m) \&\& (PG_k == m)) \quad \forall \ n, \\
&\quad if(q_m(k) == 1) \\
&\quad h_m^{'} = (k+1) \ (modN); \\
&\quad \text{else} \quad h_i^{'} = k; \\
&\quad endif
\end{aligned}
$$

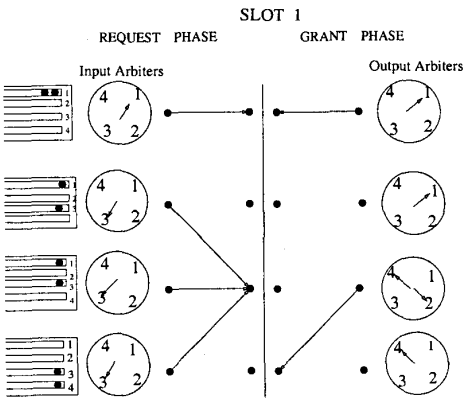

Fig. 1. PDRRM scheme

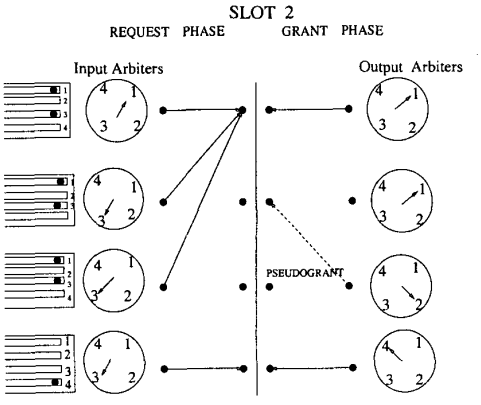

Fig. 2. PDRRM scheme

$endif$

$g_k^{'} = PG_k;$

endif

where, $B_i[k] = 1$ if there is a packet arriving in the input port i destined to output port k and PG_k is the *pseudo-grant* from the kth output. RJ_k is the 'reject' pointer in the output arbiter k. $q_i(k)$ is the occupancy of the kth VoQ at input i.

Figures 1 and 2 show an example of PDRRM arbitration algorithm. In the request phase of iteration 1 (in time slot 1), output 3 gets requests from inputs 2, 3 and 4 out of which the output 3 gives Grant to input 4. The corresponding arbiter updates its 'reject' pointer to 2, while granting to input 4. In the next slot, the inputs 2 and 3 increment their pointers to issue a grant to output 1. In the grant phase in iteration 1 (of slot 2), output 3 issues a *pseudo-grant* to 2 since output 3 did not receive any request and moreover it rejected the request from input 2 in the previous time slot. Since in this iteration also the request from input 2 is rejected by output 1, the *pseudo-grant* takes care of the matching. Thus the *pseudo-grant* facility reduces the delay and increases the throughput. Intuitively, the part of the algorithm which takes care of the output node which didn't receive any requests in the grant phase is the main difference between the EDRRM and the Pseudo-grant-DRRM (PDRRM). This part of the algorithm tries to establish additional matchings in the same iteration. We will see in the next section that a slight overhead (in maintaining reject pointer at the output arbiter) results in significant improvement in the cell delay performance.

IV. Performance of PDRRM Scheme

In this section, the throughput and the mean cell delay performance are studied using simulations and compared with that of EDRRM, DRRM and iSLIP scheduling algorithms. We consider a 16 x 16 switch for our simulations. Simulations have been run for sufficiently long time (200000 cell times) in order to get a good estimate of the performance measure of interest.

Throughput: In this subsection we study the throughput performance of PDRRM and compare its throughput performance with other switches under nonuniform traffic. The nonuniform case we consider is same as given in [3] and is modeled as follows. The arrival rate for each input is 100 % while loading for each output is also 100 %; each input has a 'hot' output (which is different from any other input's hot output) where a fraction p of cells from an input are destined to its hot output and other cells are uniformly distributed to other outputs. $p = 1/N$ corresponds to the uniform case. When $p = 1$, all the arriving cells are destined to their respective hot outputs. In this case since the hot output for each input is different from others, each input arbiter points to a different output in any time slot and the throughput for any scheme discussed in this section is 100 %. We will therefore consider only the case $1/N \leq p < 1$. Figure 3 compares the throughput of PDRRM, EDRRM and DRRM switches under the nonuniform traffic described above. The figure shows that PDRRM has certainly higher throughput than DRRM. The PDRRM switch exhibits higher throughput for values of $p > 0.4$ with EDRRM and has comparable performance for $p < 0.4$. This behavior is not surprising since EDRRM and PDRRM are optimized for nonuniform traffic. **Mean Cell Delay:** We simulated other algorithms like DRRM and EDRRM with same

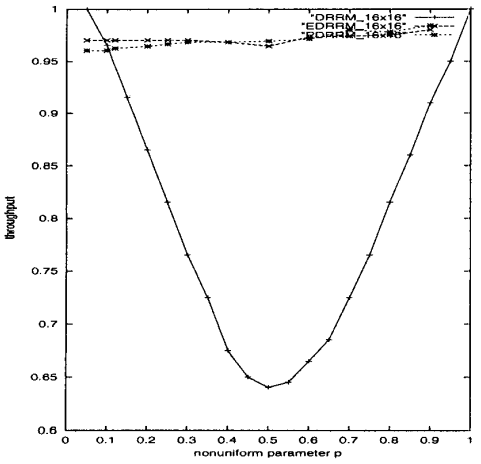

Fig. 3. Throughput in PDRRM and other schemes proposed in the literature

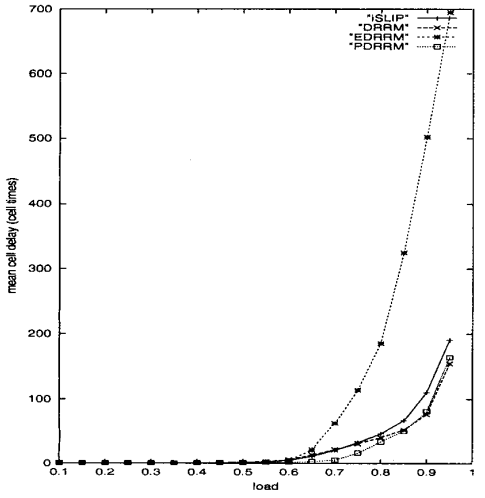

Fig. 4. Total latency in PDRRM and other schemes proposed in the literature

switch size for comparison. We estimated the total latency of cells when each of the input VoQ is fed with an *iid* Bernoulli uniform traffic. Figure 4 shows our results. From the figure, we see that PDRRM exhibits the lowest delay among all the other algorithms for all load. This demonstrates the capability of PDRRM of providing lowest delay. In fact, this is not surprising since the Exhaustive service feature of PDRRM takes care of the non-uniform traffic while the pseudo-grant feature takes care of the uniform traffic. We next look at the performance of the PDRRM algorithm under bursty traffic. The traffic model we use is an on-off Markov-modulated

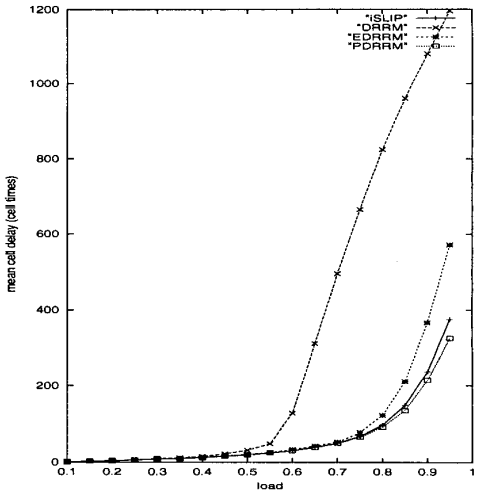

Fig. 5. Total latency of cells in PDRRM and other schemes under bursty traffic

process. We considered a geometrically distributed burst lengths with mean of 16 cells. Figure 5 shows our results. One can see from the figure that the PDRRM performs better than EDRRM, DRRM and iSLIP.

V. Conclusions

The PDRRM algorithm is a variation of the DRRM scheduling algorithm. Two features of PDRRM switch differ from DRRM switch: (i) when an input is matched with an output all the cells in the corresponding VoQ are served continuously before any other VoQ of the same input can be served and (ii) the *pseudo-grant* by an output which did not get any request in the request phase in an iteration. The performance of an PDRRM switch is better than a EDRRM switch but comparable to DRRM switch for uniform traffic. For non-uniform traffic the PDRRM switch performs undoubtedly better than DRRM and EDRRM switches.

References

[1] T. Andersen, S. Owicki, J. Saxe and C. Thacker, ""High Speed Switch Scheduling for Local Area Networks"," *ACM Trans. Compt. Sys.*, pp. 319–352, Nov 1993.

[2] N. McKeown, *"Scheduling Cells in an Input-Queued Switch"*, Ph.D. thesis, University of California at Berkeley, May, 1995.

[3] S. Panwar Y. Li and H. J. Chao, "'On the Performance of a Dual Round-Robin Switch'," *Proc. IEEE INFOCOM, 2001*, April 22-26 2001.

[4] D. N. Serpanos and P. I. Antoniadis, "' FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Multiple Input Queues'," *Web*, vol. 3, 2000.

[5] S. Panwar Y. Li and H. J. Chao, "'The Dual Round-Robin Matching Switch with Exhaustive Service'," *preprint*, Aug. 2001.