On the Performance of a Dual Round-Robin Switch

Yihan Li, Shivendra Panwar, H. Jonathan Chao

Abstract— **The Dual Round-Robin Matching (*DRRM*) switch [2] [3] has a scalable, low complexity architecture which allows for an aggregate bandwidth exceeding 1 *Tb/s* using current CMOS technology. In this paper we prove that the *DRRM* switch can achieve 100% throughput under i.i.d. and uniform traffic. The *DRRM* is the first practical matching scheme for which this property has been proved. The performance of the *DRRM* switch is then studied and compared with the *iSLIP* switch. The delay performance under uniform traffic and the hot-spot throughput of DRRM is better than that of *iSLIP*, while the throughput of *iSLIP* under some non-uniform traffic scenarios is slightly higher than that of DRRM. Since throughput drops below 100% under nonuniform traffic, we also examine some variations of the *DRRM* matching scheme for nonuniform traffic.**

Keywords— **switching,scheduling,Virtual Output Queueing,Dual Round Robin.**

I. Introduction

HIGH speed packet switches are necessary to meet the exponential growth of multimedia and other Internet traffic. Fixed-length switching technology is widely accepted as an approach to achieve high switching efficiency. Variable-length IP packets are segmented into fixed-length "cells" at inputs and are reassembled at the outputs.

In addition to using cell switching, it is important that a scalable architecture be employed to accommodate increased switching needs. Output Queuing (OQ) switches have the optimal delay-throughput performance for all traffic distributions, but the N-times speed-up in the fabric limits the scalability of this architecture. An Input Queuing (IQ) switch is desirable for high speed switching, since the internal operation speed is only slightly higher than the input line. However, an Input Queuing switch has a critical drawback [7], [8]: the throughput is limited to 58.6% due to the head-of-line (HOL) blocking phenomena.

One scheme to overcome the drawbacks and combine the advantages of an Input Queuing switch and an Output Queuing switch is a Virtual Output Queuing (VOQ) switch, in which each input maintains N queues, one for each output. By using VOQ, no addition speedup is required and HOL blocking can be eliminated. On the other hand, it is more complex to implement a VOQ switch than an IQ switch. It is important to adapt an efficient scheduling algorithm to avoid contention at inputs and thus guarantee high performance.

Considerable work has been done on improving performance using the matching algorithms. It has been proved that by using some *maximum weight matching algorithm* 100% throughput can be reached for independent arrivals [11], [12], [13]. But maximum match is not practical to implement in hardware due to its complexity, and may not guarantee fairness and quality of service. A number of practical *maximal matching algorithms* have been proposed [4], [9], [14], but maximal matching algorithm cannot achieve as high a throughput as maximum matching algorithms. Iterative algorithms as *PIM* [1] and *iSLIP* [10], [13], use multiple iterations to converge on a maximal matching.

One way to improve the performance of a VOQ switch is to increase the speedup of the switch fabric. In [5], [9], [17] a Combined Input and Output Queuing (CIOQ) switch with some matching algorithms have been proved to reach up to 100% throughput with a speedup of 4 or 2. Charny et al [4] showed that a speedup of 4 is sufficient to ensure 100% asymptotic throughput with any maximal matching algorithm. By using a fluid model, Dai and Prabhakar [6] proved that any CIOQ switch, by using any maximal matching algorithm, can deliver 100% throughput at a speedup of 2, when the input traffic satisfies the strong law of large numbers and does not oversubsribe any input or output.

Then what is the maximum throughput a VOQ switch can achieve without speedup? In this paper we will show that with a matching algorithm proposed by Chao et al [2] [3] called the Dual Round-Robin Matching (*DRRM*) achieves 100% throughput under i.i.d. and uniform traffic. Furthermore, the *DRRM* scheme provides fairness and prevents starvation. It has lower implemention complexity compared to algorithms with similar performance. In section II we will describe the *DRRM* algorithm in detail. In section III, the proof that in a DRRM switch 100% throughput can be achieved is presented. A similar proof

Yihan Li is a Ph.D. candidate in the Electrical Engineering Department, Polytechnic University, Brooklyn, NY 11201, email: yli@photon.poly.edu.

Shivendra Panwar and H. Jonathan Chao are on the faculty of the Electrical Engineering Department, Polytechnic University, Brooklyn, NY 11201, email: panwar@catt.poly.edu, chao@antioch.poly.edu

This work is supported in part by the New York State Center for Advanced Technology in Telecommunications (CATT), and also in part by the National Science Foundation under grants ITR0081527 and ITR0081357 .

for iSLIP will also be presented. The *DRRM* switch performance under uniform and nonuniform traffic will be studied in sections IV and V, respectively. Under hot-spot conditions, an overloaded hot-spot output link maintains 100% throughput. Since the *DRRM* switch does not achieve 100% throughput under nonuniform traffic, therefore in section VI we will discuss some variations of the *DRRM* scheme to accommodate nonuniform traffic.

II. THE *DRRM* ALGORITHM

In the *DRRM* scheme [2] [3], each input port maintains N VOQs. An input arbiter at each input selects a nonempty VOQ according to the round-robin service discipline. After the selection, each input port sends one request, if any, to an output arbiter. An output arbiter at each output receives up to N requests and chooses one of them based on the round-robin service discipline and sends a grant to the chosen input port. A detailed description of the two step algorithm follows:

Step 1: Request.[1] 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 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.

Figure 1 shows an example of the *DRRM* arbitration algorithm. In a request phase, each input chooses a VOQ and sends a request to an output arbiter. Assume input 1 has cells destined for both outputs 1 and 2. Since its round-robin pointer, r_1, is pointing to 1, input arbiter 1 sends a request to output 1 and updates its pointer to 2 after the request is granted by output 1. To consider output 3 in the grant phase, since its round-robin pointer, g_3, is pointing to 3, output arbiter 3 grants input 3 and updates its pointer to 4.

In *iSLIP* [10], [13], each input also maintains N VOQs for N outputs. The three steps of each iteration operate in parallel on each input and output:

[1] Note that this step is slightly but significantly different from the first step of *DRRM* as presented in earlier papers [2][3].

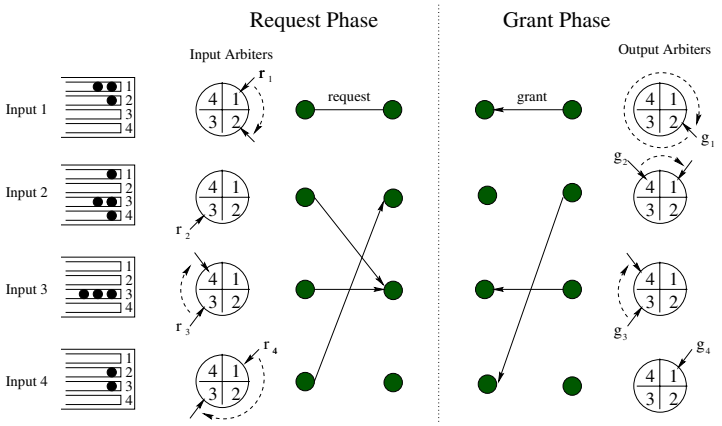

Fig. 1. An example of the dual round-robin scheduling algorithm.

Step 1: Request. Each input sends a request to every output for which it has a queued cell.

Step 2: Grant. If an output receives any 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 input whether or not its request was granted. The pointer is incremented to one location beyond the granted input if, and only if, the grant is accepted in *Step 3*.

Step 3: Accept. If an input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule stating from the highest priority element. The pointer to the highest priority element of the round-robin schedule is incremented to one location beyond the accepted output.

In *DRRM*, one arbitration has two data exchanges: (1) inputs send a total of up to N requests to outputs, and (2) outputs send grants to inputs. In *iSLIP*, three data exchanges are needed: (1) inputs send a total of up to N^2 requests to outputs, (2) outputs send grants to inputs, and (3) inputs send accept signals to outputs. In one matching cycle, *DRRM* has one less operational step than *iSLIP*, and less data exchange is needed between inputs and outputs. Thus allows the *DRRM* arbitration mechanism to be implemented in a distributed manner to make the switch simpler and more scalable. As described in [3], a terabit switch using the *DRRM* algorithm is achievable with existing electronic technology. With 0.25 μm CMOS technology, the arbitration time can be as small as $10ns$. This allows for a 256x256 switch with an incoming line bandwidth of 5 Gb/s and an internal speedup of 2.

The *DRRM* algorithm is fair and does not suffer from starvation. If input i successfully sends a cell to output j in one time slot, their pointers will all move one location and separately point to the output and the input which waited the longest. Thus output j and input i will have the lowest

priority to be chosen by input i and output j in the next time slot. If input i requests for output j but is not granted by output j in one time slot, it will keep requesting for output j until it receives a grant. An input request will wait for at most $N-1$ time slots before it is granted by the output. These advantages are also present in the *iSLIP* architecture.

With more than one iteration, the *iSLIP* algorithm can achieve better performance [10][13]. In this paper, for a fair comparison with *DRRM* without speedup, we will only consider the *iSLIP* algorithm with one iteration. By means of a simulation, the *iSLIP* algorithm with a single iteration has been demonstrated to achieve 100% throughput when the traffic is independent and uniform [10][13]. In the next section we will prove that *DRRM* can achieve 100% throughput under the same conditions. It turns out that a similar proof allows us to prove the same result for *iSLIP*.

III. PROOF OF 100% THROUGHPUT

Theorem 1: The maximum throughput of the *DRRM* switch is 100% under uniformly distributed i.i.d. traffic.

Proof:

Consider the case of heavy traffic such that none of the N^2 input queues is empty at any time. We will show that the maximum throughput is 100% by showing that under heavy traffic, after a finite time, the pointer of each input arbiter will point to an output different from all other input pointers, so that every output will be fed by a cell in every time slot.

Recall that the *DRRM* arbitration scheme has the following rules:

- At any input, if the pointer points to k in a time slot, and it is selected by the k*th* output arbiter, in the next time slot the pointer points to $(k+1) mod N$; if it is not selected, in the next time slot the pointer is still at k.
- At any output k, if only one input arbiter pointer points to k, this input will be selected by output k; if there are m $(m>1)$ input arbiter pointers pointing to output k, one of them will be selected.

We define a vector $X_i = (x_{1,i}, \ldots, x_{k,i}, \ldots, x_{N,i})$ to express the state of output arbiters; in time slot i there are $x_{k,i}$ input arbiter pointers pointing to output k, $k = 1, \ldots, N$, $0 \le x_{k,i} \le N$, $\sum_{k=1}^{N} x_{k,i} = N$. If at time slot i, $x_{k,i} = 1$, $k = 1, \ldots, N$, which indicates that each output is pointed by an input arbiter pointer, then the throughput is 100% in this time slot. We will now proceed to show $X_i = (1, 1, \ldots, 1)$ for all $i \ge N-1$. Thus a throughput of 100% can be sustained indefinitely after $N-1$ time slots.

To simplify the notation, we will drop the *mod N*, *i.e.*, *(k+l) mod N* will be represented by $k+l$. Using the *DRRM* arbitration rules to the vector X_i, we get:

$$x_{k,i+1} = \begin{cases} 0 & x_{k,i} \le 1, x_{k-1,i} = 0 \\ x_{k,i} - 1 & x_{k,i} > 1, x_{k-1,i} = 0 \\ 1 & x_{k,i} \le 1, x_{k-1,i} > 0 \\ x_{k,i} & x_{k,i} > 1, x_{k-1,i} > 0 \end{cases} \quad (1)$$

By cyclically shifting X_i to the left by one slot every slot time, we get another vector $Y_i = (y_{1,i}, \ldots, y_{k,i}, \ldots, y_{N,i})$. This Y_i is defined as follows: in time slot 0,

$$Y_0 = X_0 \quad (2)$$

i.e., $y_{k,0} = x_{k,0}$, $k = 1, \ldots, N$, and in time slot $m \ge 0$,

$$y_{k,m} = x_{k+m,m}, \quad k = 1, \ldots, N \quad (3)$$

At any time slot, $y_{k,i}$ represents the state of one output arbiter. If, and only if, in time slot i, $y_{k,i} = 1$ for all $k = 1, \ldots, N$, then $x_{k,i}$ also equals to 1 for all k. Therefore it is sufficient to show that $Y_i = (1, 1, \ldots, 1)$ for all $i \ge N-1$ to prove the theorem.

According to (1), (2), and (3), we get

$$y_{k,i+1} = \begin{cases} 0 & y_{k+1,i} \le 1, y_{k,i} = 0 \quad (\textit{condition1}) \\ y_{k+1,i} - 1 & y_{k+1,i} > 1, y_{k,i} = 0 \quad (\textit{condition2}) \\ 1 & y_{k+1,i} \le 1, y_{k,i} > 0 \quad (\textit{condition3}) \\ y_{k+1,i} & y_{k+1,i} > 1, y_{k,i} > 0 \quad (\textit{condition4}) \end{cases} \quad (4)$$

From (4), by considering the third and fourth conditions, we can conclude that whenever y_k is larger than 0, it will always be larger than 0 after that. According to this conclusion, if in time slot i, $y_{k,i} = 1$ for all $k = 1, \ldots, N$, then for any time slot $j > i$, all $y_{k,j}$ will always be larger than 0. Since there are N inputs and N outputs, and $\sum_{k=1}^{N} y_k = N$, after time slot i, $y_k = 1$, $k = 1, \ldots, N$, which indicates that $x_k = 1$ for all k and the throughput will always be 100%. We will next prove that with any initial state Y_0, in a finite number of time slots M, where M is no more than $N-1$, we will always have $y_{k,M} = 1$, $k = 1, \ldots, N$.

The state vector Y and its state transitions can be expressed as a game as shown in figures 2 and 3. In the game, we have N balls placed in N boxes. In time slot i, there are $y_{k,i}$ balls in the k*th* box, $k = 1, \ldots, N$. We will show that no matter how many balls there are in each box at the beginning, after at most $N-1$ time slots, every box will always contain exactly one ball.

The rule that determine the movement of the balls in the boxes is as follows:

In time slot i, if box k is occupied and has m balls, $m > 0$, then in time slot $i+1$, one of the m balls will stay in

box k and the others, if any, will move to box $k-1$. Since all the balls are identical, without losing generality we will require that the ball which arrived at box k earliest will stay in and occupy box k, and the others, if any, will move to box $k-1$. If more than one ball arrives at an empty box, one of them is picked arbitrarily to stay there. Thus if a ball is put into an empty box, it stays there indefinitely.

Figures 2 and 3 show two examples of the movement of balls. In the figures, each black ball occupies a box permanently and white balls keep moving until they find an empty box and occupy it. We will prove that each of the N balls will find a box to occupy permanently in no more than $N-1$ time slots, so that every box will always have one ball in it.

We will now show that the game corresponds to the state transitions of Y_i as defined in (4). By following the rules above, and by knowing how many balls there are in box k and box $k+1$ in time slot i ($y_{k,i}$ and $y_{k+1,i}$), we can get the number of balls in box k in time slot $i+1$ ($y_{k,i+1}$), which is identical with (4):

- *Condition 1*: If in time slot i, box k is empty and box $k+1$ has at most one ball, then in time slot $i+1$, box k is still empty.
- *Condition 2*: If in time slot i, box k is empty and box $k+1$ has j balls, $j>1$, then in time slot $i+1$, one of these j balls stays in box $k+1$ and box k will have the other $j-1$ balls.
- *Condition 3*: If in time slot i, box k has j balls, $j>1$, and box $k+1$ has at most one ball, then in time slot $i+1$, no ball will move from box $k+1$ to box k, and only one ball (which permanently occupies box k) will stay in box k.
- *Condition 4*: If in time slot i, there are m balls in box k and j balls in box $k+1$, $m>1$ and $j>1$, then in time slot $i+1$, one of the m balls (which permanently occupies box k) stays in box k and others move to box $k-1$, one of the j balls (which permanently occupies box $k+1$) stays in box $k+1$ and $j-1$ balls move to box k; box k will then have j balls.

In time slot 0, if a solitary ball occupies a box, that means it has already found its final box. What we need to show is that if a ball does not occupy a box in time slot 0, it will find its box within $N-1$ time slots.

Suppose in time slot 0, box k is occupied and there is a white ball (named ball B) in it, then in the next time slot ball B must move to box $k-1$. We will next use a proof by contradiction. Assume that until time slot $N-1$ ball B still cannot find its own box to occupy, which means it moved in every time slot and traveled $N-1$ boxes that are all occupied. Since box k is already occupied, all N boxes are occupied by N balls. With ball B, there will be totally

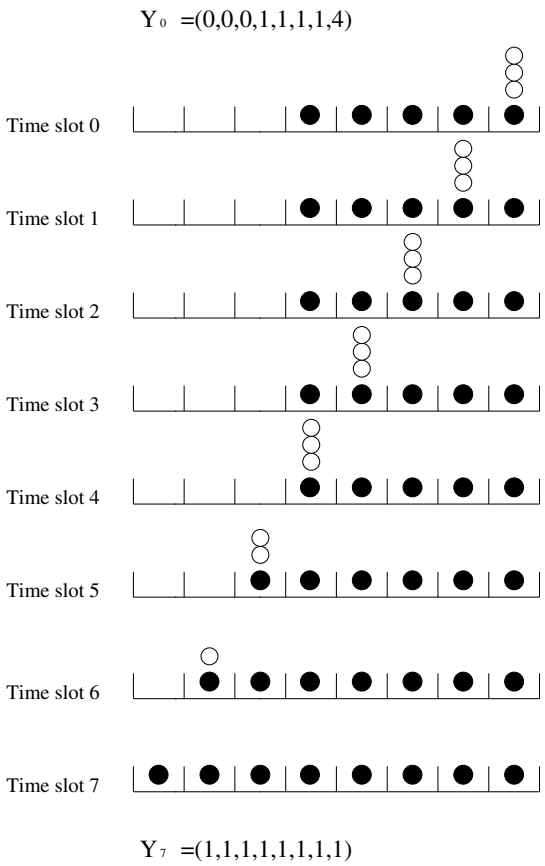

Fig. 2. The states of the system in 8 time slots when N=8 with the initial state Y_0=(0,0,0,1,1,1,1,4)

$N+1$ balls in the system, which is impossible. So the assumption is wrong and ball B will find a box to occupy within $N-1$ time slots.

Therefore we conclude that any ball can find a box to occupy within $N-1$ time slots, and from time slot $N-1$, each box has one ball in it. Thus Y_i, $y_{k,i}=1$, $k=1,\ldots,N$, $i \geq N-1$, for any Y_0, which indicates that after time slot $N-1$, each input arbiter pointer will point to a different output, and will continue to do so indefinitely. This guarantees a throughput of 100%. ■

Theorem 2: The maximum throughput of the *iSLIP* switch is 100% under uniformly distributed i.i.d. traffic.

Proof:

We will show that under heavy traffic such that none of the VOQs is empty, the maximum throughput is 100% by showing that after a finite time, each input will be pointed by an output arbiter different from all other inputs, so that every output will be fed by a packet in every time slot.

Under heavy traffic the *iSLIP* scheduling algorithm reduces to the following rules:

- In any time slot each input sends requests to every output.

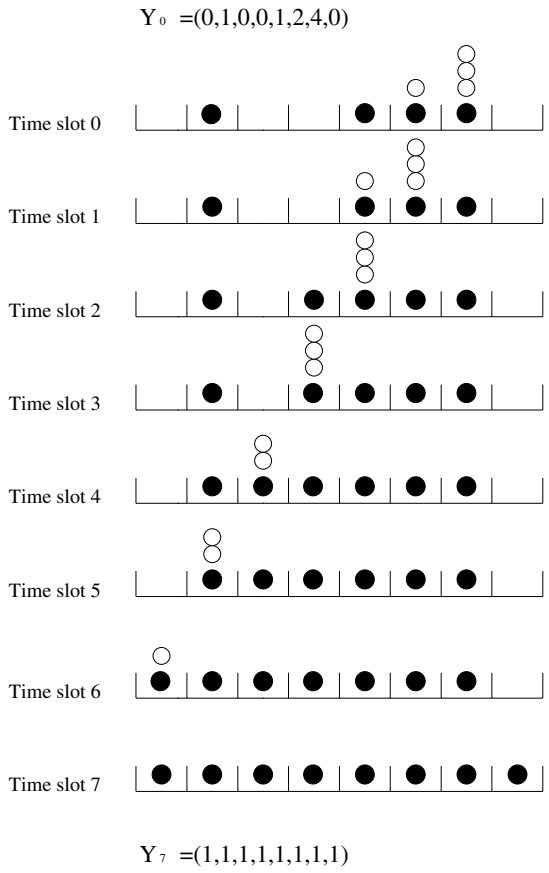

Fig. 3. The states of the system in 8 time slots when N=8 with the initial state Y_0=(0,1,0,0,1,2,4,0)

• At any output, if the pointer points to k in a time slot, this output grants the request from the k*th* input. If the grant is selected to be accepted by the k*th* input arbiter, in the next time slot the pointer points to $(k+1) mod N$; if it is not selected, in the next time slot the pointer is still at k.
• At any input k, if only one output arbiter pointer points to k, this output will be selected by input k; if there are m ($m > 1$) output arbiter pointers pointing to input k, one of them will be selected.

We define a vector $V_i = (v_{1,i}, \ldots, v_{k,i}, \ldots, v_{N,i})$ to express the state of input arbiters; in time slot i there are $v_{k,i}$ output arbiter pointers pointing to input k, $k = 1, \ldots, N$, $0 \leq v_{k,i} \leq N$, $\sum_{k=1}^{N} v_{k,i} = N$. If at time slot i, $v_{k,i} = 1$, $k = 1, \ldots, N$, which indicates that each input is pointed by an output arbiter pointer, then the throughput is 100% in this time slot. We will now proceed to show $V_i = (1, 1, \ldots, 1)$ for all $i \geq N - 1$. Thus a throughput of 100% can be sustained indefinitely after $N - 1$ time slots.

To simplify the notation, we will drop the *mod N*, *i.e.*, *(k+l) mod N* will be represented by $k + l$. Using the *iSLIP* arbitration rules to the vector V_i, we get:

$$v_{k,i+1} = \begin{cases} 0 & v_{k,i} \leq 1, v_{k-1,i} = 0 \\ v_{k,i} - 1 & v_{k,i} > 1, v_{k-1,i} = 0 \\ 1 & v_{k,i} \leq 1, v_{k-1,i} > 0 \\ v_{k,i} & v_{k,i} > 1, v_{k-1,i} > 0 \end{cases} \quad (5)$$

Note that the evolution of V_i is identical to that of X_i as defined in equation (1).

Thus by using the same method as in the proof of Theorem 1, we can prove that with any initial state V_0, in a finite number of time slots m, where m is no more than $N - 1$, we will always have $v_{k,m} = 1$, $k = 1, \ldots, N$, which guarantees a throughput of 100%.

■

IV. PERFORMANCE ANALYSIS AND SIMULATION UNDER UNIFORM TRAFFIC

In this section the performance of *DRRM* scheme without speedup under i.i.d. and uniform traffic load is presented. We will also compare its performance with the performance of the *iSLIP* switch with one iteration.

Figure 4 shows the average delay a cell may suffer in *DRRM* switch as a function of traffic load for different size of switches. Note that delay for a given load increases with switch size. As in [10], under heavy load, the *DRRM* algorithm serves each VOQ once every N cycles. Thus the queues approximate an M/D/1 queue with arrival rates $\frac{\lambda}{N}$ and deterministic service of N cell times length. For the *DRRM* switch under a heavy load of Bernoulli arrivals, the delay D for arrival rate λ per slot time can be approximated by:

$$D = \frac{\lambda N}{2(1-\lambda)} \quad (6)$$

The above expression explains the roughly linear increase in delay with N under heavy loads seen in figure 4. The average delay for an M/D/1 system is shown in figure 5, compared to a *DRRM* switch for the same value of N.

Figure 6 shows the performance of a 16x16 *DRRM* switch under bursty traffic. The traffic model we use is an on-off Markov-modulated process. Note that the delays increase approximately linearly with burst length.

Figure 7 compares the performance of a *DRRM* switch and an *iSLIP* switch with a size of 16x16 under i.i.d. and uniform traffic. The 95% confidence level intervals are also shown in the figure, but may be too small to be visible. The *DRRM* switch has somewhat higher delays for moderate loads, but has better performance for high loads.

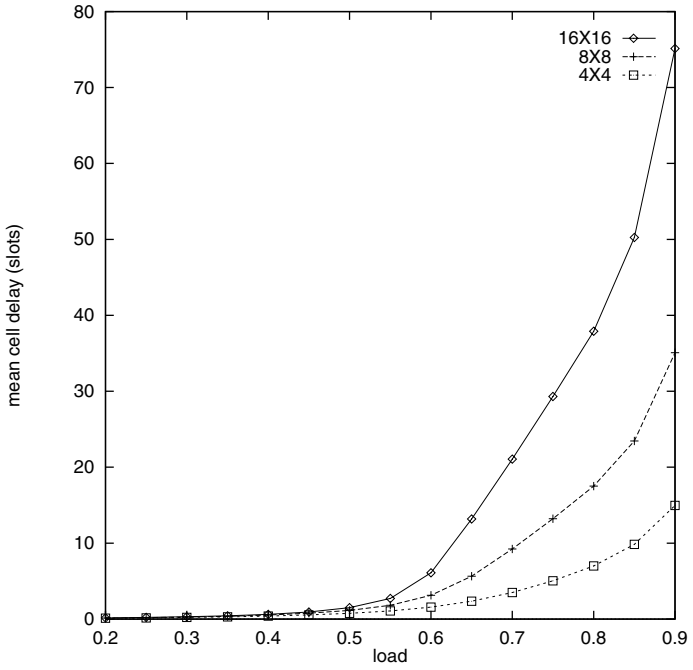

Fig. 4. The performance of *DRRM* switch with different size under uniform i.i.d. Bernoulli arrivals.

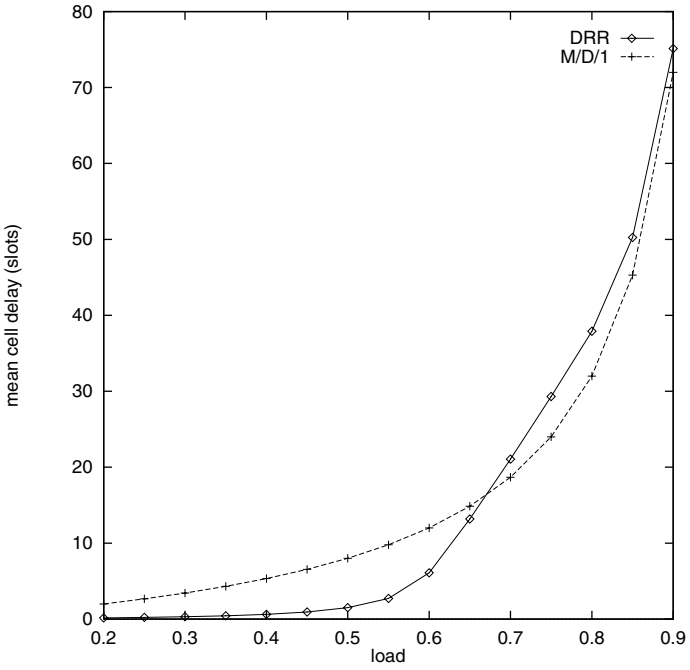

Fig. 5. Comparison of the average delay in a *DRRM* switch and an $M/D/1$ system with $N = 16$ and uniform i.i.d. Bernoulli arrivals.

V. PERFORMANCE UNDER ONE NONUNIFORM CASE - *HOT-SPOT*

We have shown the performance of *DRRM* when the arrival traffic is uniformly distributed destined to all outputs. In this section we will discuss the performance under one typical nonuniform traffic - the so-called *"hot-spot"* case.

"Hot-spot" traffic refers to a traffic pattern where many inputs send traffic one output line (the hot-spot) at the same time. This may lead to the output being loaded more than 100%. Such a condition may exist for a transient period of time. Traffic engineering, traffic control and routing should prevent such situations from occurring for extended periods of time. The hot-spot model we use here is the same as that in [16], [15]: a high rate of traffic is destined to one hot-spot and all other traffic is distributed to other outputs uniformly. Define h as the fraction of cells destined to the hot-spot. Then the cell arrival rate r on one input port can be expressed as:

$$r = rh + (1 - h)r \tag{7}$$

For one input, a cell rate of rh is destined to the hot-spot and $(1-h)r$ is destined uniformly to the other $N-1$ outputs. Thus from all the inputs, there is a total cell rate of rhN destined to the hot-spot output. When $h = \frac{1}{N}$, the traffic is uniform. Another special case is $h = 1$, when all arriving cells are destined to the hot-spot.

From simulation results, when r is 1.0, the hot-spot output line throughput in *DRRM* switch is found to be 100% for any h in the range ($\frac{1}{N}$,1). When more traffic is destined to the hot-spot, the VOQs related to the hot-spot have long queues, and only a few cells wait in the other VOQs. Some of the VOQs corresponding to other outputs may be empty. The hot-spot will always get at least one request from all inputs in each time slot, and no matter which request is granted, one cell will be sent from an input to the hot-spot.

The *iSLIP* scheme with one iteration was also examined under the same condition. Figure 8 shows the hot-spot throughput of a 16x16 *iSLIP* switch as a function of h. We can see that for a range of h the hot-spot throughput is well under 100%. This is due to the fact that with *iSLIP* an input can request for multiple outputs at one time. In some time slots, more than one input requests the hot-spot output, while at the same time each input can request for other outputs as well. The hot-spot output will grant one input among all the requests. If the chosen input accepts the grant, the hot-spot output will have a successful cell in this time slot. On the other hand, if the chosen input gets more than one grant and accepts another one, then in this time slot the hot-spot output will be unmatched even if it is the choice of several other inputs. This is different from the situation with a *DRRM* switch, in which an input will request for *at most* one output in each time slot, and if the request is granted, the input will always accept it.

For *iSLIP* when h is close to $\frac{1}{N}$, the traffic pattern is

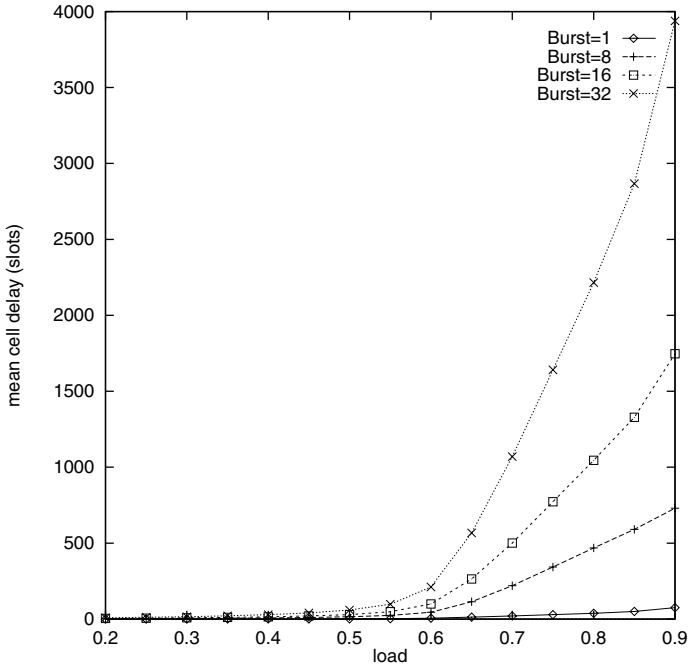

Fig. 6. The performance of a 16x16 *DRRM* switch under two-state Markov-modulated arrivals with different average burst length. Each burst of cells has one destination, which is uniformly distributed over all outputs.

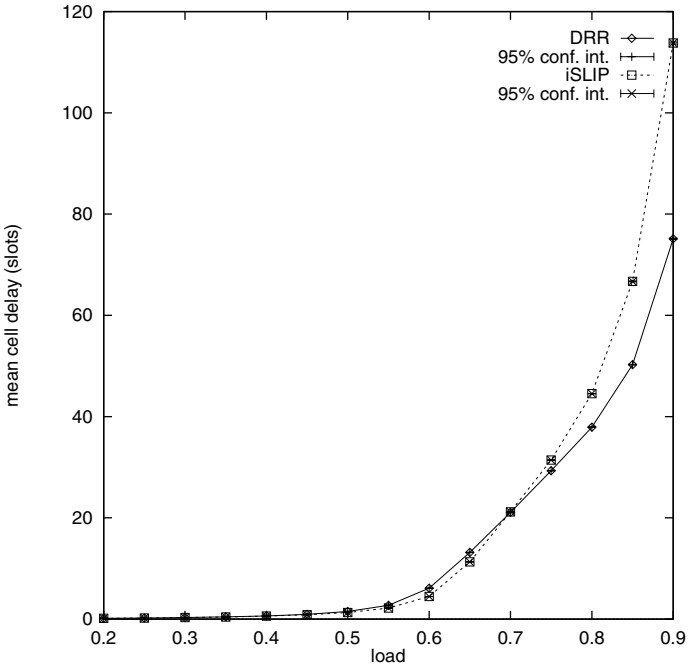

Fig. 7. Comparison on the performance of a *DRRM* switch and an *iSLIP* switch with size 16 under i.i.d. uniform Bernoulli arrivals.

nearly uniform and the throughput is close to 100%. Similarly, as h approaches 1, the hot-spot throughput will approach 100%. The reason is that when h is large, only a few incoming cells are destined to other outputs and therefore those VOQs will be empty at most of the time. It is more likely that some inputs will almost always only request for the hot-spot output and will then accept the grant if they are offered one by the output.

VI. VARIATIONS FOR NONUNIFORM TRAFFIC

In this section we study the *DRRM* switch performance for another i.i.d. nonuniform case and discuss a variation of the *DRRM* algorithm.

In the last section, we considered one nonuniform traffic pattern - hot-spot, in which the arrival rate for the hot -spot output can be more than 100% when the arrival rate to each input is 100%. We will consider a nonuniform traffic pattern, where the loading for each input and output is 100%, but the cells arriving at one input are not uniformly distributed to each output. Since the *DRRM* switch is fair uniformly across VOQs, some heavily load VOQs may only be served with a rate no higher than other lightly loaded VOQs. This is not desirable since some of the VOQs will have unacceptable loss due to buffer overflow. In this section the behavior of *DRRM* switch is studied and three variations of it are discussed for nonuniform traffic patterns.

The nonuniform traffic pattern we consider in this section is as follows. The arrival rate for each input is 100%, while the loading for each output is also 100%; each input has a "hot" output, with a fraction p of cells from an input are destined to its hot output, and other cells are uniformly destined to other outputs; each input has a hot output which is different from any other input's hot output. $p = \frac{1}{N}$ 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 only consider $\frac{1}{N} \leq p < 1$.

We define a ratio r of the cell rate from one input to its hot output, p, to the cell rate from the input to one of the other outputs, $\frac{1-p}{N-1}$, as follows:

$$r = \frac{p}{1-p}(N-1) \tag{8}$$

Another useful ratio is s, the ratio of the desired cells successfully transferred from one input to its hot output to the number of cells successfully transferred from the input to another output. If s is close to r, all VOQs are receiving fair service; if not, the scheduler is favoring one type of VOQ over the other.

Simulation results show that a *DRRM* switch with a speedup of 2 delivers 100% throughput. The throughputs

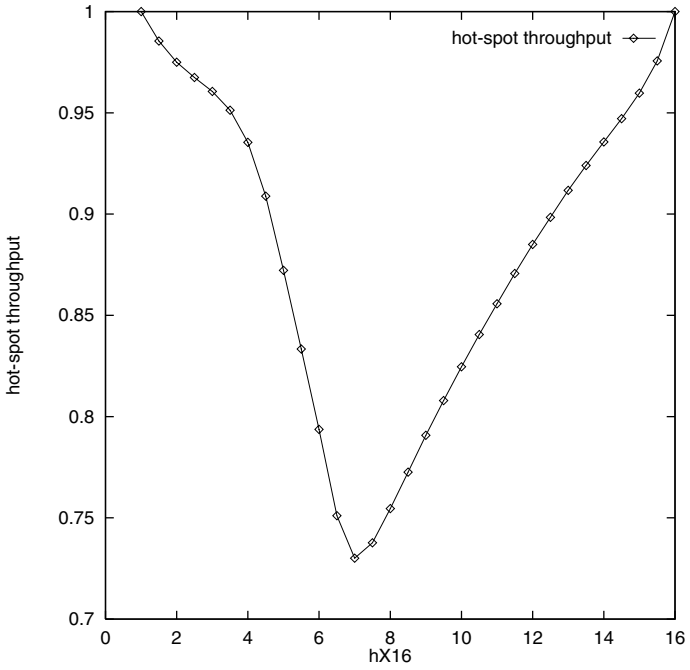

Fig. 8. The throughput of the hot-spot output line for a 16x16 *iSLIP* switch as a function of h. The DRR switch achieved a throughput of 1.0 for all h.

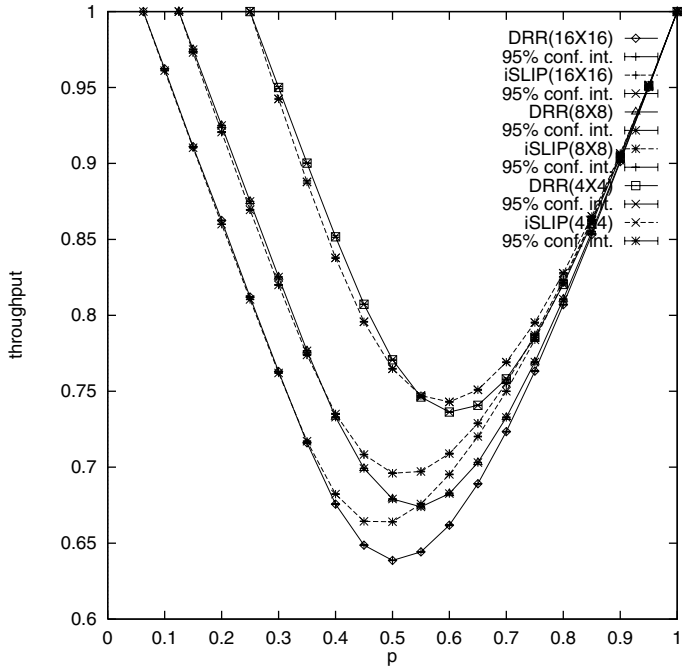

Fig. 9. Comparison of throughput for a DRR switch and an iSLIP switch under nonuniformly distributed arrivals.

of a *DRRM* switch without speedup and an *iSLIP* switch of size $N = 4$, 8 and 16 under the nonuniform traffic described above are shown in figure 9. The *DRRM* achieves somewhat lower throughput than *iSLIP* for values of p around 0.6, but the two switches have similar performance for other values of p.

In order to improve performance under nonuniform traffic and give the VOQs destined to hot outputs more chances than others, we consider three weighted *DRRM* schemes: for an input i and its hot output j,

- *Scheme (a)* modify input round-robin schedule so that m visits per cycle are given at input i for the hot output j;
- *Scheme (b)* modify output round-robin schedule so that m visits per cycle are given at output j for input i;
- *Scheme (c)* modify both input and output round-robin schedule scheme as in schemes (a) and (b).

We examined these three schemes under 100% traffic load by simulations. To simplify the notation, and without losing generality, we assume that the hot output of input i is output i. In scheme (c), output i uses the same round-robin sequence as input i. Figure 10 shows one example of the modified round-robin sequence in a 4x4 switch when $m = 2$, and we use these sequences to get the performance shown in Figure 11.

Figure 11 shows the system throughput for a 4x4 switch as a function of p with the original *DRRM* schemes and the weighted *DRRM* schemes with $m = 2$. From figure 11 we

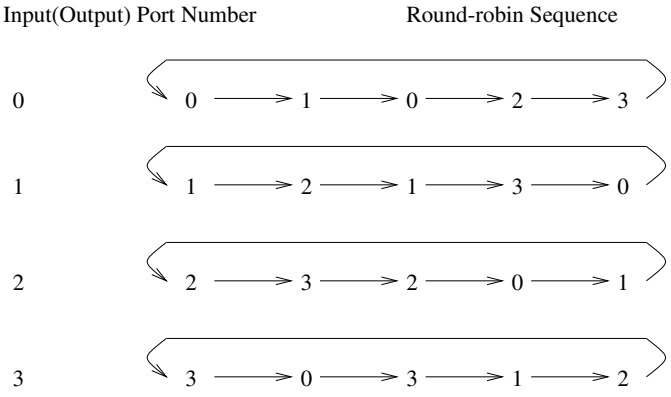

Fig. 10. An example of the modified round-robin sequences for a 4x4 switch

can see that when $m = 2$, schemes (a) and (b) have even lower throughput for some ranges of p than the original *DRRM* scheme. Scheme (c) improves the throughput when $0.4 < p < 0.7$ which corresponds to $2.0 < r < 7.0$ and has a similar or lower throughput for other ranges of p.

Figure 12 shows the ratio s as a function of p under the same condition as in figure 11. We find that when p is large enough, all the schemes including the original *DRRM* lead to similar values of s. As p increases, s is increasingly close to r. When $p = 0.7$ and $r = 7$, s is around 4.5; when $p = 0.9$ and $r = 27$, s is around 24. The inability of the weighted *DRRM* schemes to fully compensate for the nonuniform traffic is explained by the mismatch between s and r.

We also studied the weighted *DRRM* schemes with $m \geq 3$, and the simulation results show that they do not work well. Scheme (b) with $m \geq 3$ and $m = 2$ have similar performance. For scheme (a) and (c) with $m \geq 3$, s is similar to those with $m = 2$, while the throughput is almost always lower than the original *DRRM*. This is explained by how the *DRRM* arbiter updates pointers. With the variation, input arbiters give the hot output more chances in one round-robin cycle. When the input pointer is pointing to the hot output while the hot output is busy, the input pointer will not update until its request is granted after several time slots, which will block the cells in other VOQs that are destined to a free output. When m is too large, this influence becomes greater and decreases the overall system throughput.

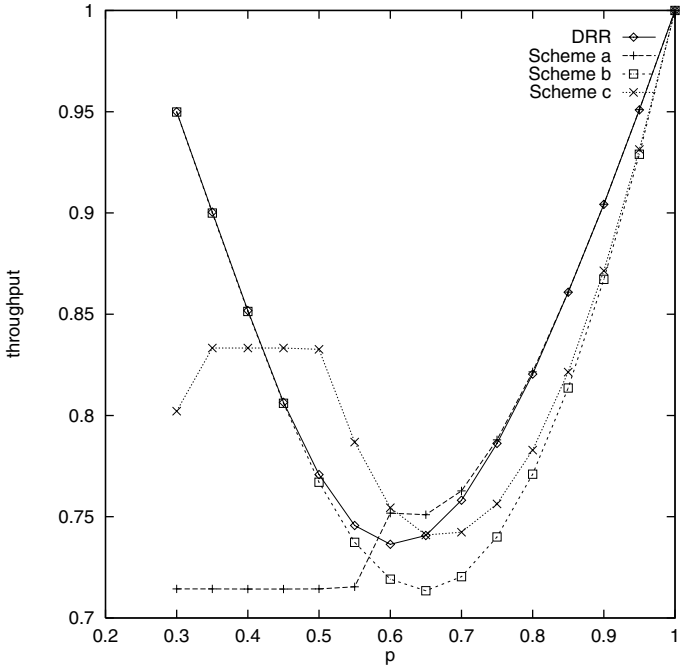

Fig. 11. The throughput as a function of p for *DRRM* and three weighted *DRRM* schemes.

According to the analysis and simulation results, we conclude that when p is within some intermediate range, Scheme (c) will lead to a better system performance. When p is close to $\frac{1}{N}$ or 1, the original *DRRM* scheme works well.

VII. Conclusions

The *DRRM* algorithm uses round-robin arbitration on both the input and output sides. It is simple to implement and is capable of switching at terabit speeds. In this paper, we prove that a switch with the *DRRM* matching scheme achieves 100% throughput under uniform traffic. Using a similar proof technique, we also show that the *iS-*

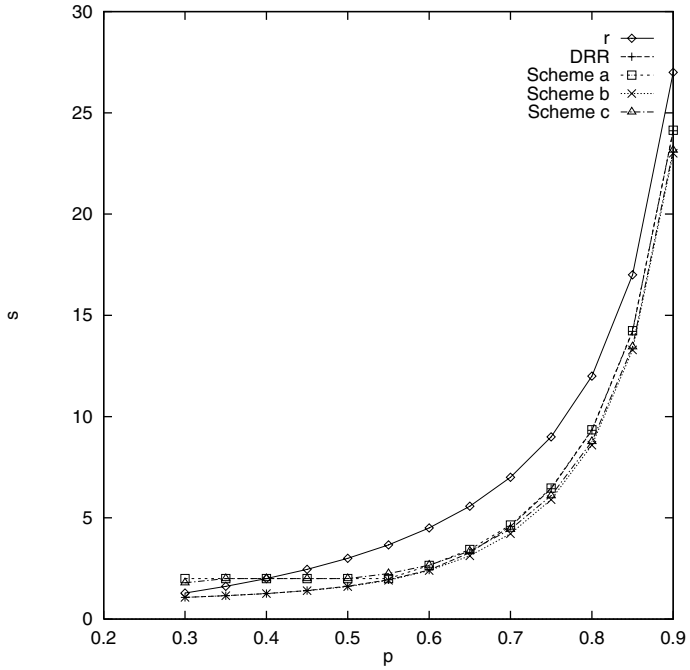

Fig. 12. s as a function of p for *DRRM* scheme and three weighted *DRRM* schemes compared to r

LIP matching scheme achieves 100% throughput uder uniform traffic. Under one typical nonuniform traffic - hotspot, the hot-spot output of a *DRRM* switch delivers 100% throughput. For other nonuniform traffic distributions, the throughput drops below 100%. A weighted variation of DRR is shown to improve performance for nonuniform traffic distributions. In future work, we will further examine the *DRRM* switch performance for nonuniform traffic and for speedup factors greater than 1.

Acknowledgments

The authors would like to acknowledge Jin-Soo Park, Roberto Rojas-Cessa and Eiji Oki for the illuminating discussion we had.

References

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe and C. P. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. on Computer Systems*, vol. 11, No. 4, pp. 319-352, Nov. 1993.

[2] H. J. Chao and J. S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch", *Proc. IEEE ATM Workshop*, Fairfax, VA, May 1998.

[3] H. J. Chao, "Saturn: a terabit packet switch using Dual Round-Robin", *IEEE Communication Magazine*, vol. 38 12, pp. 78-84, Dec. 2000.

[4] A. Charny, P. Krishna, N. Patel and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in Input-Buffered crossbars with speedup", in *IWQOS'98*, May 1998.

[5] S. Chuang and N. McKeown, "Matching output queuing with a combined input output queued switch", *INFOCOM'99*, pp. 1169-1178.

[6] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup", in *Proc. IEEE INFOCOM '2000*, pp. 556-564.

[7] M. J. Karol, M. Hluchyj, and S. Morgan, "Input vs. output queuing on a space-division packet switch", *Proc. GLOBECOM* 1986, pp. 659-665.

[8] M. J. Karol, M. Hluchyj, and S. Morgan, "Input versus output queuing on a space-division packet switch," *IEEE Trans. on Communications*, vol.35, pp. 1347-1356, 1987.

[9] P. Krishna, N. S. Patel, A.Charny and R. Simcoe, "On the speedup required for work-conserving crossbar switches", in *IWQOS'98*, May 1998.

[10] N. Mckeown, "The iSLIP scheduling algorithm for Input-Queued switches", *IEEE/ACM Trans. Networking*, vol. 7, pp. 188-201, April 1999.

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

[12] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an Input-Queued switch", *IEEE Trans. Communications*, vol. 47, No. 8, pp. 1260-1267, Aug. 1999.

[13] N. McKeown, "Scheduling algorithms for input-queued cell switches", *Ph.D. Thesis*, UC Berkeley, May 1995.

[14] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches", *IEEE INFOCOM 98*, Vol 2, pp. 792-799, April 1998.

[15] J. S. Park, "Design and Analysis of Large-Capacity Multicast Packet Switches", *Ph.D. dissertation*, Polytechnic University, Jan. 1998.

[16] G. F. Pfister, " 'Hot Spot' contention and combining in multistage interconnection networks", *IEEE Trans. on Computers*, vol. C-34, No. 10, pp. 943-948, Oct. 1985.

[17] B. Prabhakar and N. McKeown "On the Speedup Required for Combined Input and Output Queued Switching", *Computer Systems Technical Report CSL-TR-97-738*. November 1997.