Multicast Scheduling in Buffered Crossbar Switches with Multiple Input Queues

Shutao Sun[1], Simin He[2], Yanfeng Zheng[2], Wen Gao[1,2]
[1]Graduate School, Chinese Academy of Sciences, Beijing, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{ stsun, smhe, yfzheng, wgao }@jdl.ac.cn

***Abstract*—We consider the problem of scheduling multicast traffic in a buffered crossbar switch with multiple input queues at each input port. In this paper, we design and investigate a series of combinations of queuing policies and scheduling algorithms and report the simulation result. It is shown that a small number of input queues at each input port can dramatically improve the performance under burst multicast traffic in buffered crossbar switches. Under this architecture, it is feasible to design simple queuing policies and scheduling algorithms for high speed switches while keeping high performance and small size of buffer within crossbar.**

Index terms—scheduling; multicast; buffered crossbar switch

I. INTRODUCTION

Input queuing (IQ), together with a crossbar switch fabric, virtual output queues (VOQs) [1], [2] and fixed-size cells, is now extensively used in high-speed routers. In each time slot, the scheduling of unicast traffic in an $N \times N$ input-queued switch is a bipartite matching problem, and naturally many bipartite matching algorithms, maximum or maximal, weighted or not, are utilized in IQ scheduling, such as LQF[2], LPF [3], PIM [1], iSLIP [2], and DRRM [4].

Recently, more and more applications are involved in the use of multicast, such as multimedia streaming, group interactions and cluster-based communication. This leads to a tremendous increasing of multicast traffic over the Internet. Many schemes have been proposed to address the multicast scheduling for input queued crossbar switches, but most of them are based on the simplest queuing discipline – first in first out (FIFO). The multicast scheduling scheme can be based on no fanout-spliting or fanout-splitting service disciplines [5], [6]. In the no fanout-splitting service discipline, a multicast cell must be transferred to all of its destinations in the same time slot. In the fanout-splitting service discipline, a multicast cell may be transferred to the output ports over any number of time slots. It has been shown that the fanout-splitting service discipline significantly improves the throughput of the multicast switch system.

Algorithm TATRA was proposed in [7]. The idea behind this algorithm is to schedule the head of line cells in such a way that it leaves the residue, i.e., copies of head of line cells that cannot be scheduled in the current time slot, on the smallest number of input ports so as to make more new cells in the input queues attend the scheduling process in the next time slot. Although TATRA achieves good performance, it is difficult to implement since the process cannot be parallelized. WBA was also proposed in [7], which achieves similar performance to TATRA but is easy to be parallelized.

This work was supported in part by NSFC-69983008, KGCXZ-103 and the basic research fund of ICT.

Switches with a single FIFO queue for multicast at each input port usually experience HOL blocking, which degrades the throughput. HOL blocking for multicast traffic can be eliminated by the multicast VOQ architecture [8], in which each input has to maintain up to $(2^N - 1)$ queues. However, this architecture is impractical in large switches because of its poor scalability.

In order to alleviate the HOL blocking, a window-based resolution scheme [9] was proposed, which allows a window of cells at the head of an input queue to contend for accessing idle output ports in the current time slot. However, it needs complicated hardware logic to implement. Other approaches were proposed in [10], [11], which maintain multiple queues in one input port. Simulation results show this scheme achieves better performance.

In [12], the authors considered the output contention resolution for multiple time slots instead of the current time slot only. It outperforms most of those algorithms which only consider the output contention resolution for the current time slot, yet at the cost of the higher scheduling complexity.

However, for high-capacity switch, none of these algorithms has been considered as an efficient solution because of performance and/or implementation problems. Buffered crossbar switches have recently attracted more attention owing to their potential to yield both faster and less expensive switches. Researchers have found that the buffered crossbar switches achieve good performance with simple scheduling algorithms [13-16]. But existing works mainly focus on the unicast scheduling.

Mhamdi etc. first proposed to handle multicast traffic by buffered crossbar switch with a single FIFO queue at each input port [17]. The authors designed a MXRR algorithm and evaluated its performance. It was shown that the new scheme is more efficiently and far better than the previous schemes. However, under burst traffic, the gain of buffered crossbar in handling multicast traffic is marginal due to serious HOL blocking existing in FIFO queue.

In this paper we proposed buffered crossbar with multiple queues at each input port to handling multicast traffic. We design and investigate different algorithms under the new architecture. The simulation results show that adding a few queues at each input port can greatly improve the performance while keeping the simplicity of the algorithms.

The rest of the paper is organized as follows. In Section II we describe the proposed architecture and algorithms. In Section III we evaluate the proposed schemes by simulation. We end this paper with concluding remarks in Section IV.

II. ARCHITECTURE AND ALGORITHMS

A. Architecture

Traditional discussions on multicast usually focused on crossbar switches without any buffer at the crosspoints. For this architecture, the scheduling algorithms are usually quite complex, hence unsuitable to high capacity switches. However, by adding a small amount of buffer at the crosspoints, which is feasible in current technology, the scheduling problem radically changes and is dramatically simplified: For a $M \times N$ switch, $M + N$ schedulers, M at inputs and N at outputs, operate in an independent and parallel manner, and each of them deals with only a single resource.

By extending the work in [17], we consider a switch model as Fig. 1. There are M input and N output ports in our model. Each input has k queues to contain fixed-size multicast cells. There is no inner speedup and hence no output queues. The crossbar contains $M \times N$ small queues, one per crosspoint. The scheduler of buffered crossbar switch with multiple input queues operates in three stages. The first is the cell assignment (CA) stage. Each input with a cell arrival assigns the cell to one of the queues. Because the number of queues is much smaller than the number of possible multicast addresses, this operation is not as simple as that for unicast scheduling, in which the scheduler puts the cell directly in the queue for the appropriate output port. The second is input scheduling (IS) stage. Each input picks a cell and sends its copies to the crosspoint buffers corresponding to its fanout set. The third is output scheduling (OS) stage. Each output picks a crosspoint buffer and takes a cell from it. All the processing can be run on each input and output independently and in parallel. Therefore the scheduling in buffered crossbar can be pipelined to run at high speed, and this makes buffered crossbar switches appeal for high performance switches and routers.

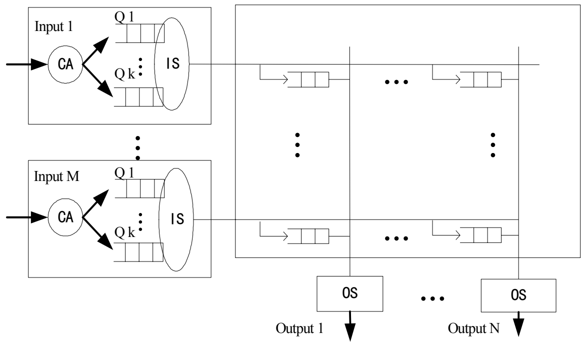

Fig. 1. The architecture of buffered crossbar with multiple input queues for multicast scheduling

B. Algorithm

In this subsection, we will discuss the details of scheduling schemes of different stages.

1) Cell assignment (CA)

The scheme to assign incoming cell to queues has an effect on the performance of the system. There are three intuitive design principles pointed out in [11], [12].

1. The head of queues should be "diverse", i.e. they should span a large part of the set of all outputs for which the input holds cells.
2. Cell with multicast addresses that are the same or very similar should go to the same queue, so as to provide more scheduling options.
3. Load balancing among queues may be beneficial to improve performance.

A scheme named Majority was proposed in [10]. It associates multiple sets of masks with each input queue. When a cell arrives, it is put in the queue for which the number of matching ports between the mask and the cell fanout is the highest. Ties are resolved by multiple levels of comparisons. However, there is no detail for the algorithm and it's not clear how many levels of comparisons are enough. Bianco etc. proposed minimum distance queuing (MDQ) and load-balanced queuing (LBQ) for cell assignment [11]. In order to partition the load among the queues, they keep a list with all the multicast flows. The size of the list is up to $(2^N - 1)$ in the worst case. Furthermore, both of them need to be aware of the traffic patterns of the multicast flows to get a good partition of the load among the queues. This is too complex for a high speed switches.

In this paper we propose some cell assignment schemes for high speed switch. We choose the schemes with reasonable complexity as the candidates.

1. Cell oriented round-robin assignment (CRRA)
   Each cell is assigned to one of the k queues according to the round-robin priority. This scheme distributes the load to the queues uniformly.
2. Burst oriented round-robin assignment (BRRA)
   Each burst is assigned to one of the k queues according to the round-robin priority. In this paper, the burst is defined as a continuous arrival of cells with the same fanout set. This scheme satisfies the second design principle to some extent; however, it cannot assure that cells in different bursts but with the same multicast address were assigned to the same queue.
3. Cell oriented shortest queue first (CSQF)
   This scheme assigns a cell to the queue with the smallest occupancy. It considers the load balance among queues at an input, but the decision is made based on the instant status of the switch.
4. Burst oriented shortest queue first (BSQF)
   This scheme assigns a burst to the queue with the smallest occupancy. It considers the load balance among queues at an input while satisfying the second design principle to some extent

2) Input scheduling (IS)

After the cell assignment, the input should pick a cell to send its copies to the crosspoint buffers corresponding to its fanout set in each time slot. We call it input scheduling. In this paper, we propose and investigate 4 schemes.

1. Round robin (RR)
   In this scheme, the scheduler selects the cell to transfer to the crosspoint buffers according to the round-robin priority.
2. Minimum residue first (MRF)
   Among the cells at the heads of the queues, this scheme picks the cell with minimum residue after transmission. If there are more than one cell having the minimum residue, ties are broken by selecting one of them randomly. The residue of a cell is defined as the number of outputs to which that the multicast cell has not transferred. This scheme tries to complete the transmission of a cell as soon as possible, so as to let the new cell behind it attend the contention.
3. Maximum service first (MSF)
   In this scheme, for each cell at head of queue, the number of its copies that can be transferred to crosspoint buffers in current time slot is calculated. And the scheduler picks a cell with the largest number. Ties are broken randomly.
4. Maximum ratio of service first (MRSF)
   In this scheme, for each cell at head of queue, the ratio of the number of its copies which can be scheduled in current time slot, to the number of all its copies waiting for being scheduled is calculated. And the scheduler picks a cell with the maximum ratio. Ties are broken randomly. This scheme tries to combine the advantage of both MRF and MSF.

3) *Output scheduling (OS)*

1. Round robin (RR)
   The scheduler selects the cells in crosspoint buffers to transfer to output according to the round-robin policy.
2. Longest queue first (LQF)
   This scheme can be used when the size of the crosspoint buffer is greater than one cell. It chooses the cell at the head of the longest queue at crosspoint, and tries to free the heavily occupied crosspoint buffer so that more blocked cell at input can be handled.

All scheduling stages, including cell assignment, input scheduling and output scheduling, can use random scheme. However, we find that this scheme usually shows a little poorer performance than the round robin scheme. In addition, its complexity is also a little more than round robin. So we don't discuss and evaluate random scheduling in this paper.

III. PERFORMANCE EVALUATION

The simulation results are gathered from a 2×8 and an 8×8 buffered crossbar switch with different number of queues in each input. Traffic patterns used in our simulation include Bernoulli i.i.d arrival and burst arrival, and all inputs and outputs are equally loaded. For burst traffic, the traffic is generated as a burst arrival of cells (ON burst), followed by bursts of no cells (OFF burst). All cells within an ON burst have the same destination. Following [17], the average burst size we use is 16 cells. For both traffic models, the multicast vectors are uniformly distributed over all possible multicast vectors. Thus for a $M \times N$ switch, the average fanout is $N/(2(1-2^{-N}))$. The traffic load over an output is $\rho M/(2(1-2^{-N}))$, where ρ is the cell arrival rate at an input port. In order to evaluate the maximum throughput, we keep traffic load over an output 100% in our simulation.

A. *Evaluation of the cell assignment schemes*

In this subsection, we evaluate performances of different cell assignment schemes. We mainly focus on the throughput behavior. When conducting this evaluation, we adopt the round robin scheduler for both input and output scheduling. Fig. 2 shows the saturated throughput performance for an 8×8 buffered crossbar switch with 4 queues at each input, and Fig. 3 shows the throughput for a 2×8 buffered crossbar switch with 4 queues at each input. From the simulation result, we find that there is no significant difference between round-robin and shortest queue first scheme, even though they have different complexity.

It is very interesting that the saturated throughput of cell oriented algorithms is higher than that of burst oriented algorithms. We speculate that for highly offered load, there are usually many cells in queues. So even if the cells belonging to the same burst are assigned to different queues, they usually don't arrive at the head of the queue at the same time. Therefore they don't alleviate the diversity of the head of queues. For cell based schemes, after completely transferring a cell, a new cell with different destination will appear sooner than the burst oriented algorithms, so the crosspoint buffer gets more opportunities to be filled. This will improve the throughput performance.

B. *Evaluation of the input scheduling schemes*

In this subsection, we evaluate the performance of different input scheduling schemes. We use CRRA for cell assignment and RR for output scheduling. The number of queues at each input is equal to 4.

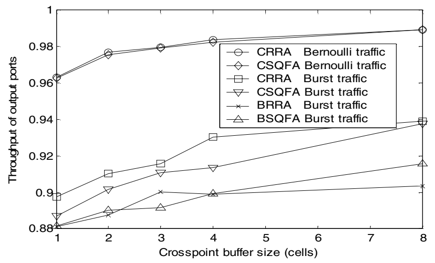

Fig. 2. Throughput of output ports for an 8×8 switch with different cell assignment schemes

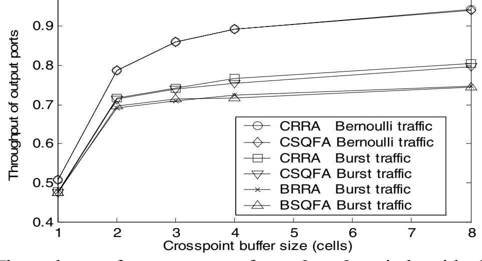

Fig. 3. Throughput of output ports for a 2×8 switch with different cell assignment schemes

Figs. 4 and 5 show the simulation results of different input scheduling schemes for a 2 × 8 switch under different traffic patterns. The RR scheme usually achieves the poorest performance, whereas the MRSF achieves the best. When the size of crosspoint buffer is equal to 1, the throughputs of different schemes show quite large difference. The difference between RR and MRSF is more than 0.1. But when the size of crosspoint buffer is greater than 3, the difference becomes quite small.

For an 8 × 8 switch, all schemes achieve similar throughput performance with a difference less than 0.015. We omit the figures due to space limitation.

C. Evaluation of the output scheduling schemes

When we conduct the evaluation of the performance of different output scheduling schemes, we use CRRA for cell assignment schemes and RR for input scheduling scheme. All the evaluations are carried out under buffered crossbar switches with 4 queues at each input. Fig. 6 shows the simulation result. We find that different output scheduling schemes achieve almost the same throughput.

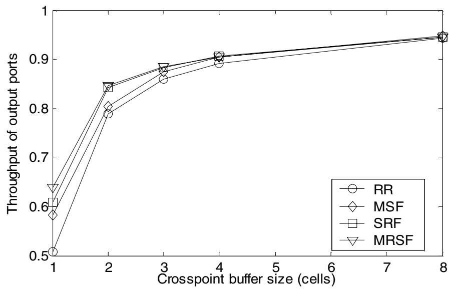

Fig. 4. Throughput of output ports for a 2 × 8 switch with different input scheduling schemes under Bernoulli i.i.d traffic

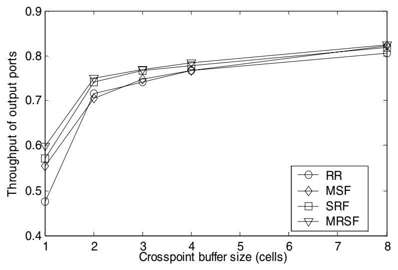

Fig. 5. Throughput of output ports for a 2 × 8 switch with different input scheduling schemes under burst traffic

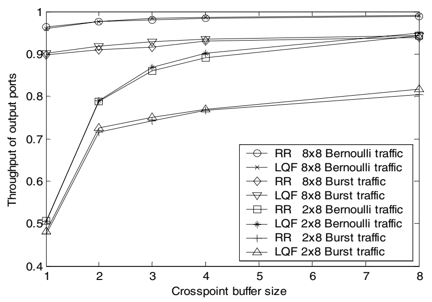

Fig. 6. Throughput of output ports for different output scheduling schemes

D. Evaluations of the effects of the number of queues at each input and buffer sizes at crosspoint

We choose two sets of scheduling algorithms, CRR-RR-RR and CRR-MRSF-LQF. CRR-RR-RR is a combination with lowest complexity. MXRR proposed in [17] appears in our simulation as CRR-RR-RR with single queue at each input. CRR-MRSF-LQF tries to achieve the best performance by the algorithms with higher complexity.

Fig. 7 shows the throughput for an 8 × 8 switch with CRR-RR-RR scheme under Bernoulli i.i.d traffic. We find that both increasing the number of queues at each input and increasing the size of crosspoint buffer can improve the throughput performance. But the size of crosspoint buffer counts more.

Fig. 8 shows the throughput for a 8 × 8 switch with CRR-RR-RR scheme under burst traffic. Unlike the result under Bernoulli i.i.d traffic, increasing the size of the crosspoint buffer is of less help to the improvement of throughput performance than increasing the number of queues at each input.

Fig. 9 shows the throughput for a 2 × 8 switch with CRR-RR-RR scheme under Bernoulli i.i.d traffic. Increasing the number of queues at each input cannot improve the throughput. This is because for a 2 × 8 switch with a single queue, the head of line blocking is usually not very serious under Bernoulli i.i.d traffic. The scheduler always completely transfers a cell and then serves a new one behind it at once. This probably is more efficient than serving different queues at an input port in round robin manner over multiple time slots, which sometimes continuously accesses the cells with little residue.

However, we can see from Fig. 10 that the gain still mainly comes from the multiple input queues at each input for burst traffic. Especially, when the size of crosspoint buffer is greater than 1 cell, the throughput performance is improved much more with the increasing of the number of input queues.

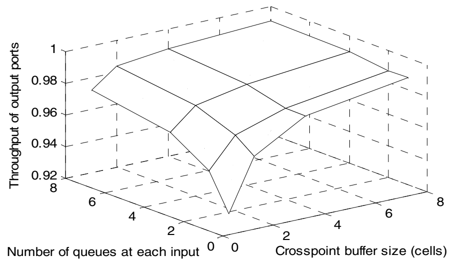

Fig.7. Throughput of output ports for an 8 × 8 switch with CRR-RR-RR scheme under Bernoulli i.i.d traffic

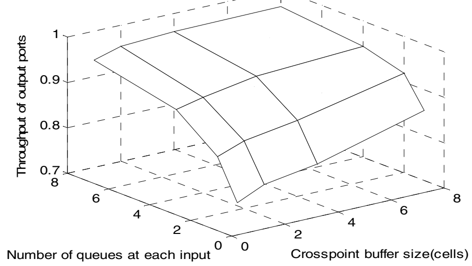

Fig. 8. Throughput of output ports for an 8 × 8 switch with CRR-RR-RR scheme under burst traffic

Figs. 11 and 12 show the simulation result of CRR-MRSF-LQF scheme for a 2×8 switch. It achieves better performance than CRR-RR-RR at the cost of higher complexity. It achieves throughput more than 0.9 with 8 queues at each input when the size of crosspoint buffer is equal to 8. But for 1 queue at each input, the throughput declines to 0.65.

For an 8 × 8 switch, the performance of CRR-MRSF-LQF is similar to that of CRR-RR-RR. We omit the simulation result due to space limitation.

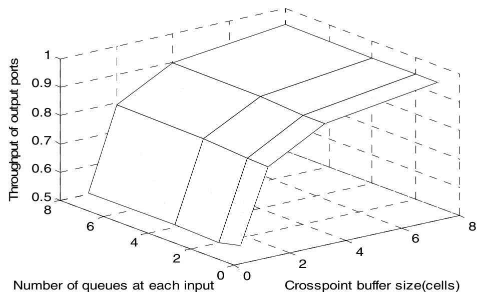

Fig. 9 Throughput of output ports for a 2 × 8 switch with CRR-RR-RR scheme under Bernoulli i.i.d traffic

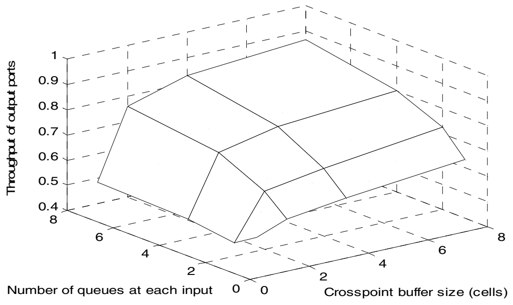

Fig. 10. Throughput of output ports for a 2 × 8 switch with CRR-RR-RR scheme under burst traffic

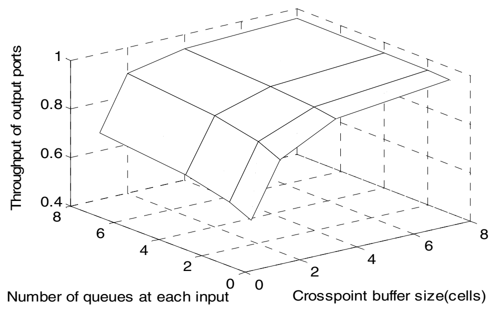

Fig. 11. Throughput of output ports for a 2 × 8 switch with CRR-MRSF-LQF scheme under Bernoulli i.i.d traffic

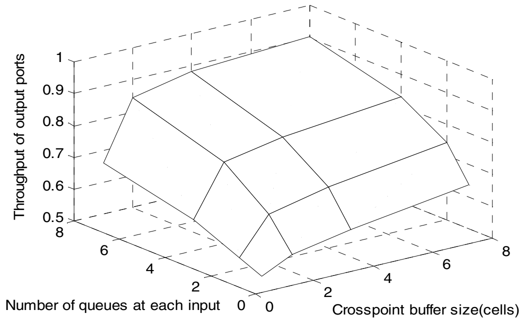

Fig. 12. Throughput of output ports for a 2 × 8 switch with CRR-MRSF-LQF scheme under burst traffic

IV. CONCLUSION

In this paper, we propose to handle multicast traffic by buffered crossbar switch with multiple input queues at each input port. We design a series of algorithms for proposed architecture and make an intensive investigation on them. The simulation results show that increasing the size of crosspoint buffers cannot improve much the performance under burst multicast traffic when the input only uses a single FIFO queue. However, slightly increasing the number of queues at each input can significantly improve the throughput performance under burst multicast traffic, even by very simple algorithms. So it is feasible to construct high speed multicast switches by the buffered crossbar with multiple queues at each input while achieving high performance. Among three scheduling stages, the input scheduling is the key factor to affect the system performance.

REFERENCES

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

[2] N. McKeown, "Scheduling algorithms for input-queued switches," Ph.D. Thesis, University of California at Berkeley, 1995.

[3] A. Mekkittikui and N. MecKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. INFOCOM*, San Francisco, 1998, pp. 792-799.

[4] H.J. Chao and J.S. Pack, "Centralized contention resolution schemes for a large-capacity optical ATM switch," in *Proc. IEEE ATM Wksp.*, Fairfax, VA, May 1998, pp. 11-16.

[5] X. Chen and J.F. Hayes, "Call scheduling in multicasting packet switching," *Proc. IEEE ICC'92*, pp. 895-899.

[6] J.Y. Hui and T. Renner, "Queuing strategies for multicast packet switching," *Proc. GLOBECOM'90* , 1990, pp. 1347-1356.

[7] B. Prabhakar, R. Ahuja, and N. McKeowm, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 15, 1997, pp. 855-866.

[8] M. Ajmone Marsan, A. Bianco, P.Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Trans. Networking*, vol. 11, no.3, June 2003, pp. 465-477.

[9] K. Schultz and P.Gulak, "Distributed multicast contention using content addressable FIFOs," *Proc. IEEE ICC'94*, New Orleans, LA, 1994, pp. 1495-1500.

[10] S. Gupta and A. Aziz, "Multicast scheduling for switches with multiple queues," *IEEE Hot Interconnects'02*, Stanford, CA, Aug. 2002.

[11] A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and C. Piglione, "On the number of input queues to efficiently support multicast traffic in input queued switches," *Proc. HPSR 2003*, Torino, Italy, June 2003, pp. 111-116.

[12] W.T. Chen,C.F. Huang, Y.L. Chang, and W.Y. Huang, "An efficient cell-scheduling algorithms for multicast ATM switching systems," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, Aug. 2000, pp. 517-525.

[13] R. Rojas-Cessa, E. Oki, and H.J. Chao, "CIXOB-k: combined input crosspoint-output buffered packet switch," *Proc. IEEE GLOBECOM,*San Antonio, Texas, 2001, pp. 2654-26-2660.

[14] L. Mhamdi and M. Hamdi, "MCBF: A High-Performance Scheduling Algorithm for a Buffered Crossbar Switch Fabric," *IEEE Communications Letters*, vol.7, no. 9, Sep. 2003, pp. 451-453.

[15] T. Javidi, R.Magill, and T.Hrabik, "A high-throughput scheduling algorithm for a buffered crossbar switch fabric," *Proc. ICC'01*, Helsinki, Finland, June 2001, pp. 1586-1591.

[16] N. Chrysos and M. Katevenis, "Weighted fairness in buffered crossbar scheduling," Proc. *IEEE HPSR'03*, Torino, Italy, June 2003, pp. 17-22

[17] L. Mhamdi and M. Hamdi, "Scheduling multicast traffic in internally buffered crossbar switches," *Proc. IEEE ICC'04*, Paris, France, June 2004, pp. 1103-1107.