Transactions D:
Computer Science & Engineering and
Electrical Engineering
Vol. 16, No. 2, pp. 119–127

Research Note

A Multiple Slot Cell Scheduling Algorithm for Multicast Switching Systems

F.Ch. Lee[1], W.F. Wang[1,*] and J.B. Shih[1]

Abstract. *In this study, we propose a multicast switching system called the Blocking Reduction Multiple Slot Cell Scheduler (BRMSCS) switch. The BRMSCS switch consists of shared memory banks, a crossbar fabric and the BRMSCS scheduler. Our goals are to relieve the blocking situation in the scheduler and to guarantee freedom from a memory access conflict, that is, no more than two output ports should access different cells that come from the same input port. To meet the goals, the BRMSCS scheduler can quickly insert address cells into a scheduling table and fill the scheduling table as full as possible. The simulation results show that the BRMSCS scheduler can efficiently insert the address cells into the conflict free locations of the scheduling table and has the advantage of reducing blocking.*

Keywords: *Multicast; Packet switch; Multicast scheduling algorithm; Switch.*

INTRODUCTION

Switches can basically be classified into Input Queuing (IQ) switches and Output Queuing (OQ) switches. The ideal OQ switches can achieve ultimate throughput, and their packet latency is similar to an M/D/1 queuing system. However, the OQ switches are impractical when the input line rate or port number increases. For an input queuing switch, Karol et al. showed that the system throughput is limited to 58.6% [1]. This is caused by Head Of Line (HOL) blocking. Adopting Virtual Output Queues (VOQs) can completely eliminate HOL blocking [2].

For multicast traffic, a traditional unicast switch replicates a multicast packet into multiple unicast packets before entering the switch system. This causes a serious input and output blocking problem. As a result, switching multicast traffic by a traditional unicast switch causes two problems. First, the size of memory requirement increases. Second, the utilization of the bandwidth decreases. Recently, high performance multicast switching architectures and scheduling algorithms have been proposed to tackle the problems.

In [3], McKeown et al. proposed a switching architecture with multicast queues and an ESLIP scheduler, which is an enhanced version of the iSLIP algorithm [3], to improve the performance of switching multicast traffic. Chao et al. proposed a serial Dual Round Robin Matching (DRRM) scheduler and token tunneling method to reduce the information changing inside the switch [4-6]. Lee et al. proposed a PSLIP scheduler which stands for a parallel iSLIP scheduler, combining a dedicated path in a switch fabric to improve the multicast switching performance in [7]. In [8], the TATRA scheduler was proposed to schedule multicast traffic. The TATRA scheduler is easy to implement; however, more than two output ports access different cells which come from the same input port. We call this situation the problem of Memory Access Conflict (MAC). Chen et al. proposed a switching architecture which combines the copy network, the non-blocking routing network and the MSCS scheduler for multicast traffic [9]. The MSCS scheduler selects as many cells as possible for the copy network and the copied cells go through the non-blocking routing network to output ports. However, there is a blocking situation caused by the MSCS scheduler.

In this study, we propose a multicast switching architecture, called a BRMSCS (Blocking Reduction Multiple Slot Cell Scheduler) switch, to reduce the blocking situation in the switch and guarantee no memory access conflict. By simulation, it is shown that the BRMSCS scheduler has the advantage of reducing blocking and efficiently scheduling multicast traffic.

1. *Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Douliu, Yunlin 64002, Taiwan.*
*. *Corresponding author. Email: wwf@yuntech.edu.tw*

Received 11 February 2009; received in revised form 21 September 2009; accepted 7 November 2009

The rest of this paper is organized as follows. First, the model of the proposed BRMSCS switch is described. Next, we make a detailed description and illustration of the BRMSCS scheduler. Finally, the simulation results of the BRMSCS switch and conclusions are presented.

SWITCH MODEL

As shown in Figure 1a, the BRMSCS switch is organized by input queues, a crossbar fabric and a BRMSCS scheduler. It separates the control and data sections in its internal architecture [10], as shown in Figure 1b. The data section is responsible for storing the payload of incoming cells. The control section is responsible for scheduling the output order for incoming cell headers. Before entering the crossbar fabric, the cell header and its corresponding payload is rebound.

Buffering Strategy

To eliminate the HOL blocking problem, N queues must be adopted in each input port where N is the switch size. However, for switching multicast traffic, there are $(2^N - 1)$ queues to completely eliminate the multicast HOL blocking problem. This will make the multicast switches impractical. For saving cost and for memory efficiency, the BRMSCS switch adopts shared memory as input queues, as shown in Figure 1. When a cell arrives, the Data/Address Separator will assign an available address for storing the cell's payload in the input shared memory. At the same time, the cell's header will be copied as an address cell and sent to the BRMSCS scheduler. An address cell involves the following information: the source port, the destination ports, the payload address and the arrival time. The payload address is used to indicate the physical address of the shared memory of the cell payload. The source port and destination ports are used to schedule the cell's departure order. If an incoming cell belongs to multicast traffic, the arrival time is used to recognize the cell replication. There is an additional table to record the number of replicated arrival cells. When a cell departs from the switch, the corresponding record will be decreased by one. When the record becomes zero, the corresponding cell's payload will be removed from the shared memory [11].

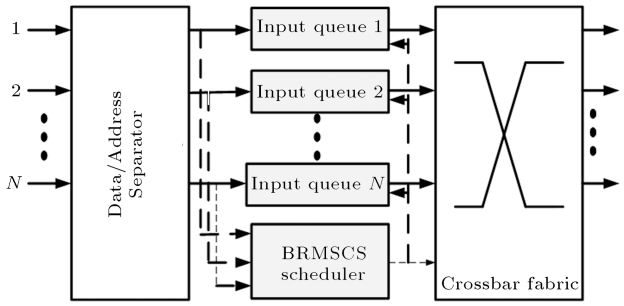

Figure 1a. An abstract view of the BRMSCS switch.

Switching Strategy

The BRMSCS switch adopts a crossbar fabric. The major reason for this is that the crossbar fabric has a native property for multicasting. By controlling the N^2 cross-points of the crossbar fabric, it can provide N non-blocking paths from inputs to outputs. The crossbar fabric has two constraints:

a) Each input port can only send, at most, one cell into the fabric in a time slot;

b) Each output port can receive, at most, one cell from the fabric in a time slot.

To maximize the switching performance under multicast traffic, a new integrated scheduler for unicast and multicast traffic is needed.

Scheduling Strategy

The BRMSCS scheduler is organized as in Figure 2. When an address cell arrives, it will be queued in

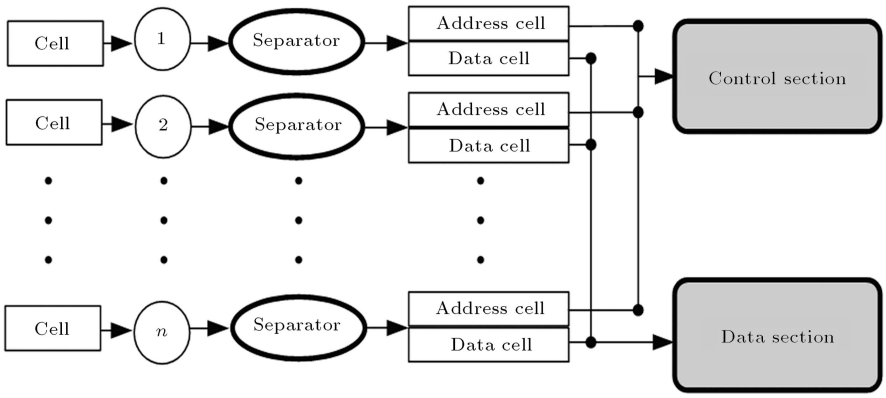

Figure 1b. A detailed view for the data flow of an incoming cell.

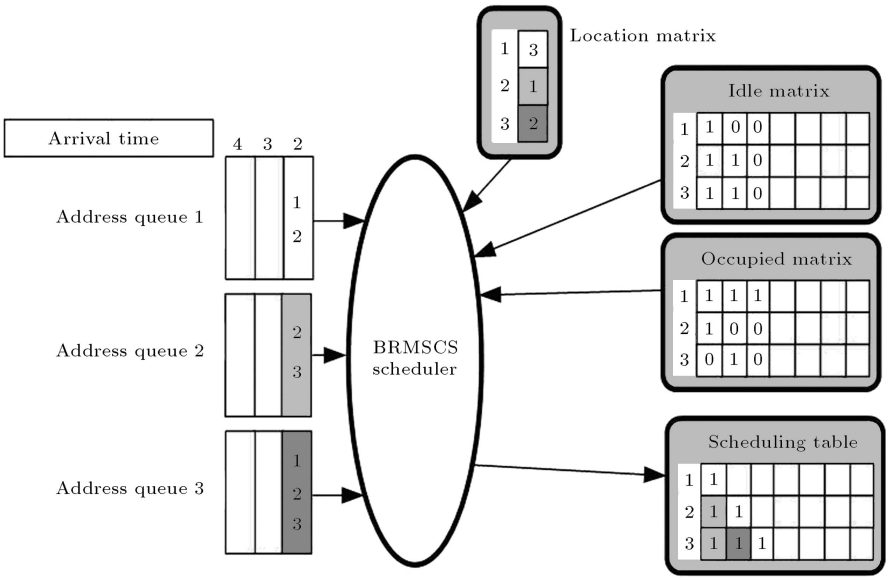

Figure 2. The BRMSCS scheduler.

the corresponding First In First Out (FIFO) address queues. Only the HOL address cell in each address queue can be scheduled by the BRMSCS algorithm. The scheduling results will be stored at the scheduling table. To meet the crossbar constraints, the scheduling results must guarantee that there is no MAC problem. The BRMSCS algorithm guarantees to find the MAC free location in the scheduling table. At the end of each time slot, each address cell at the first column of the scheduling table will be removed and rebound with its cell's payload. The rebound cell is transferred from the input queue through the crossbar fabric to its output port.

Since HOL blocking might occur in an address queue, the BRMSCS algorithm will not only insert the address cell quickly to a MAC free location in the scheduling table by using the location matrix, but also find the optimal MAC free location by using an occupied matrix and an idle matrix.

BRMSCS ALGORITHM

In this section, we mainly describe what the BRMSCS algorithm is and how it works. The definitions used in the algorithm are given, as well as a description of its details. We also give a demonstration for the algorithm using examples.

Definitions

1. $C_{i,j}$: an address cell coming from input port i and destined to output port j. If it is a multicast cell, it contains multiple output ports.
2. ID($C_{i,j}$): the identifier of an address cell, $C_{i,j}$.
3. ST(x, y): a scheduling table, where x and y represent row x and column y of the scheduling table. ST(x, y)=ID($C_{i,j}$) means that column y and row x of the scheduling table are occupied by address cell $C_{i,j}$, whose identifier is ID($C_{i,j}$).
4. LM(n), Location Matrix: LM(n) is the last location of the address cell in the scheduling table, which comes from input port n.
5. OM(x, y): an occupied matrix, where x and y represent row x and column y of the occupied matrix. When OM(x, y) is not null and OM(x, y) = k, there is at least one location of column y of the scheduling table that is occupied by an address cell, which comes from input x and destined to output k.
6. T(ST(x, y)): the arrival time of address cell, $C_{i,j}$, which the identifier equals to ST(x, y).
7. IM(x, y): an idle matrix where x and y represent row x and column y of the idle matrix. IM(x, y) = True means that row x and column y of the scheduling table is occupied, otherwise IM(x, y) = False.

BRMSCS Algorithm

The BRMSCS algorithm involves three procedures: fetch, insert, and search and exchange, as shown in Figure 3. The fetch procedure fetches the source port, destination port and arrival time of the HOL address cell, $C_{i,j}$. After fetching, the insert procedure finds a MAC free location of the scheduling table quickly by using the Location Matrix, LM(j), and inserts the HOL address cell into the scheduling table. The search and exchange procedure is used to find an empty location and exchange the first eligible address

```
[The BRMSCS Algorithm]
/* Procedure Insert */
For i = 1 to N /* Switch size */
/* referring to the LM and finding a MAC free location */
   If HOL cell of Address queue (i) ≠ NULL
     Let C_{m,n} = HOL cell of Address queue (i)
    For each fanout of C_{m,n}
     For k = LM(i) + 1 to Len
      If IM(n, k) is false
        Set ST(n, k) = ID (C_{m,n})
        set IM(n, k) = True
        set OM(m, k) = n
      EndIf
     EndFor
    EndFor
   EndIf
EndFor
/* Procedure Search and Exchange */
For i = 1 to N
   For j = 1 to L     /* L is the search depth */
/* If scheduling table is null, finding the eligible cell */
    If IM(i, j) is False
     For k = j + 1 to L
      If IM(i, k) is True
        Set C_{m,y} is the corresponding Cell of IM(i, k)
        Check eligible:
        (1) OM(i, j) = null
        (2) (OM(i, j) ≠ null) AND (T(ST(i, j)) =
          T(ST(OM(i, k), y)))
        If eligible
          Swap (ST(i, k), ST(i, j))
          Update OM and IM
        EndIf
      EndIf
     EndFor
    EndIf
   EndFor
EndFor
```

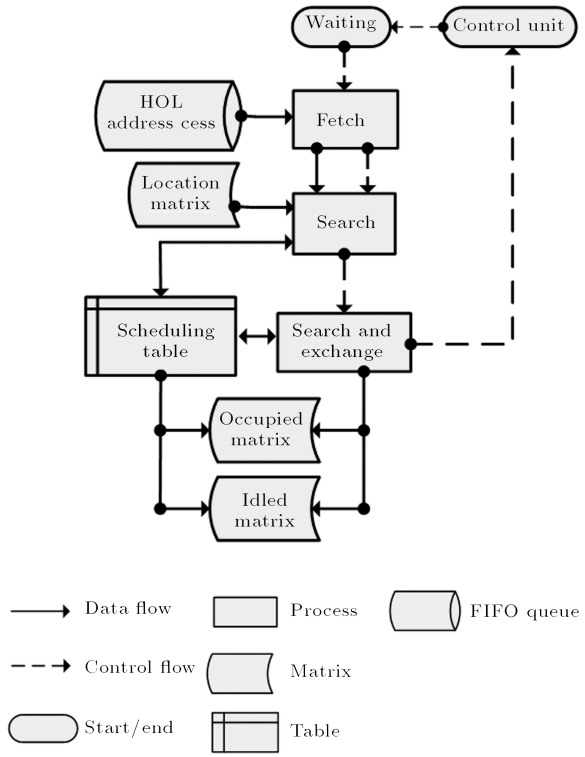

Figure 3. The BRMSCS algorithm.

cell with the empty location by using the Occupied Matrix, OM(i, j), and the Idle Matrix, IM(i, j). An eligible address cell is the address cell behind the empty location that guarantees to be MAC free, after exchanging the eligible address cell with the empty location.

Fetch Procedure

In the fetch procedure, the source port, destination port and arrival time of each HOL address cell of FIFO address queues are fetched, using a local round-robin pointer.

Insert Procedure

In the insert procedure, the address cells, $C_{i,j}$, are selected and inserted into the scheduling table in the fetching order. The insert procedure first refers to LM(i) as the starting location, i.e. LM$(i) = p$, $0 \leq p \leq Len$, where Len is the length of the scheduling table, and then searches for the first empty location from ST$(j, p+1)$. This procedure executes for, at most, $(L-p)$ times. To keep fairness, at the end of the insert procedure, the round-robin pointer is updated by an increase of one. The LM will be updated at the end of the search and exchange procedure.

Search and Exchange Procedure

To maximize the switch performance, the HOL column of the scheduling table must be filled as full as possible. The search and exchange procedure tries to fill the empty location of the scheduling table by exchanging the address cell from its original location with a front and empty location without MAC. The search and exchange procedure will search the first empty location for each column. If the empty location, say ST(x, y), is found, the procedure starts to search the first eligible address cell, say $C_{m,n}$, from column $y+1$ in row n. To guarantee being MAC free, the eligible address cell, $C_{m,n}$, located in ST(n, p) must satisfy the following two conditions:

Condition 1 : [OM$(x, y) =$ null],

Condition 2 : [(OM$(x, y) \neq$ null) and (T(ST(n, p))

$$= \text{T(ST(OM}(n, p), y))\text{)}].$$

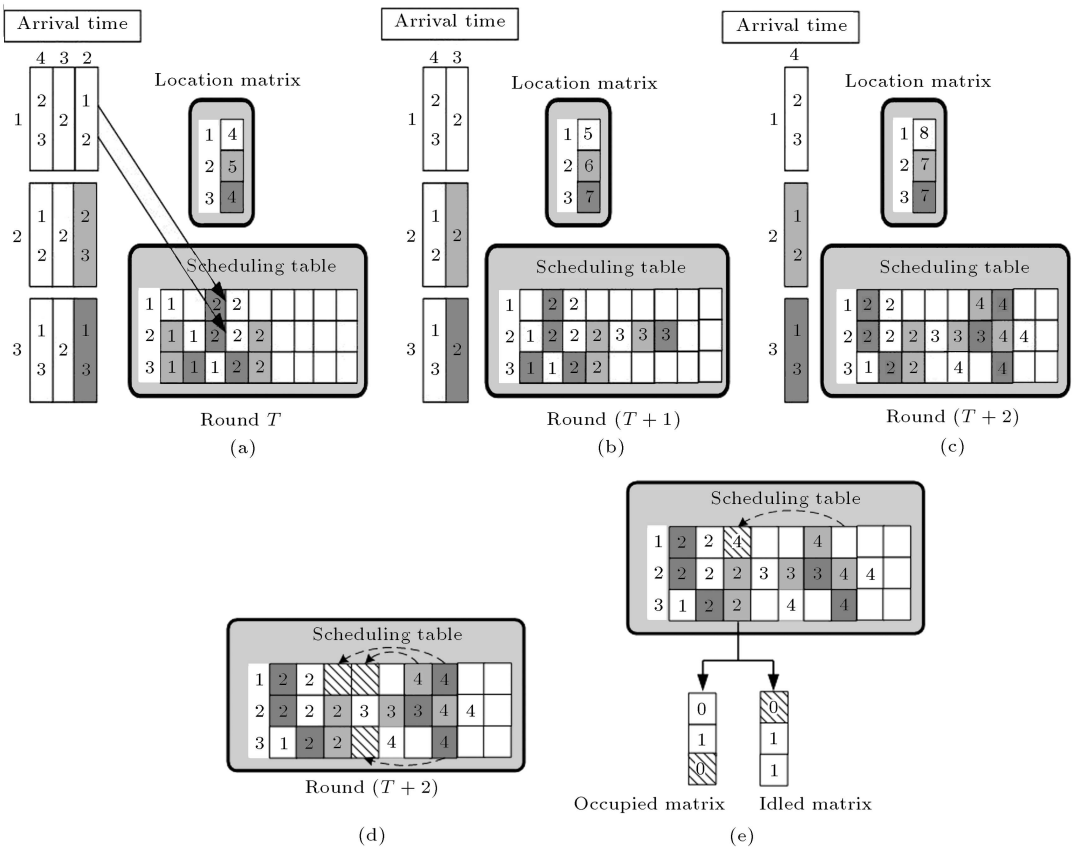

Figure 4. The procedure insert and procedure search and exchange.

If the address cell located after column $y+1$ in row x is unicast traffic, the search and exchange procedure only verifies Condition 1. $\mathrm{OM}(x,y) = \mathrm{null}$ means that the input port of $C_{m,n}$ has not appeared in column y of the scheduling table. As a result, the $C_{m,n}$ is eligible to move to an empty location, $\mathrm{ST}(x,y)$, from $\mathrm{ST}(n,p)$ without MAC. If the verified address cell is multicast traffic, the search and exchange procedure verifies both Condition 1 and Condition 2. For example, if cell D occupies $\mathrm{ST}(x,z)$, $z \neq y$, and the arrival time of the eligible cell, $C_{m,n}$, are the same, it means that the eligible cell, $C_{m,n}$, and cell D are the same fan-out of a multicast cell. Moving the eligible cell, $C_{m,n}$, to an empty location can still guarantee a MAC free condition.

Illustration

Figures 4 and 5 give examples of the BRMSCS algorithm. Figures 4a, 4b and 4c demonstrate the BRMSCS algorithm only executing the insert procedure. As shown in Figure 4a, address queue 1 is selected by the insert procedure. It will refer to LM(1) and find the first empty location from column 4 to the end of the scheduler table. After executing the insert procedure three times around, the inserting result is shown in Figure 4c. Obviously, as shown in Figure 4d, we can exchange the eligible address with the empty position of the scheduling table to maximize the switch performance. When the first empty location is found (e.g. ST(1,3)), the search and exchange procedure starts to find the first eligible address cell, $C_{m,n}$. The cell located in ST(1,6) will be verified first. It is, however, against Condition 1. The cell located in ST(1,7) will be verified next. Since address $C_{3,1}$ follows Condition 1, it will be exchanged with ST(1,3), as shown in Figure 4e.

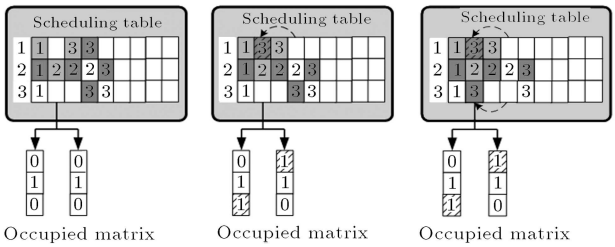

Figure 5. Searching & exchanging multicast traffic.

Figure 5 demonstrates the search and exchange procedure exchanging a multicast cell. Assume that the first eligible address cell, D, is exchanged from ST(1,4) to ST(1,2) successfully, as shown in Figure 5b. The search and exchange procedure will try to find

an eligible address cell to exchange with ST(3,2). As shown in Figure 5c, address cell $C_{3,3}$ is against Condition 1 and the search and exchange procedure will verify whether $C_{3,3}$ follows Condition 2 or not. Because of address cell D and $C_{3,3}$ having the same arrival time, address cell D and $C_{3,3}$ are the same fan-out of one multicast cell. Exchanging $C_{3,3}$ from ST(3,4) to ST(3,2) will not cause the MAC problem.

SIMULATION RESULTS

In the simulation, search depth L can be set to 4, 16 and 64. For example, BRMSCS(4) means that the search depth is 4. The switch size is set to 32×32 and the simulation time set to one million time slots. Each time slot includes two phases: arrival of incoming cells and running of the BRMSCS algorithm. Also, we assume there is no cell lost in the switch. Three kinds of traffic model in [9] are used to evaluate the performance of the proposed BRMSCS switch: unicast random traffic, multicast random traffic and multicast bursty traffic. The average fan-out (NC) is set to 2, 4 or 8. We compared the BRMSCS scheduler to the MSCS scheduler, proposed in [9], and a base scheduler. The base scheduler is an unsophisticated scheduler of BRMSCS that only executes fetch and search procedures. All simulation programs are done by C++, and performance metrics include:

- Throughput [12];
- Average HOL delay: The time starting from an address cell becoming the HOL address cell until its last fan-out leaves from the scheduling table;
- Average overbooking rate: When the fan-out of the HOL address cell cannot be scheduled into the scheduling table in one time slot, overbooking has occurred. The overbooking rate is the ratio of the number of successfully scheduled fan-outs to unsuccessfully scheduled fan-outs. It can be used to evaluate the blocking frequency of BRMSCS and MSCS schedulers.

Uniform Random Traffic

Figures 6 to 8 illustrate the simulation results of BRMSCS and MSCS schedulers under uniform random traffic. Obviously, we can see that when the search depth increases, the throughput can be improved significantly. Even for the base scheduler, when the search depth increases, the throughput can achieve 89%. In Figures 7 and 8, the average overbooking rate of BRMSCS is lower than MSCS. Consequently, the throughput and average HOL delay of BRMSCS are better than MSCS.

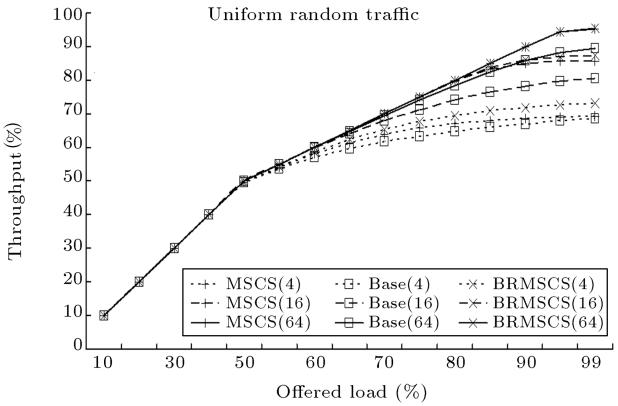

Figure 6. The throughput under uniform random traffic.

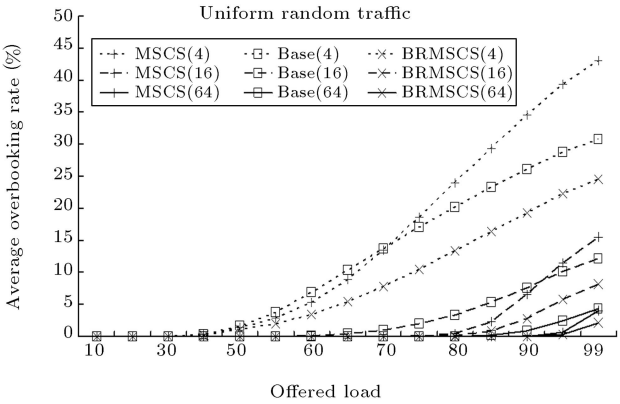

Figure 7. The average overbooking rate under uniform random traffic.

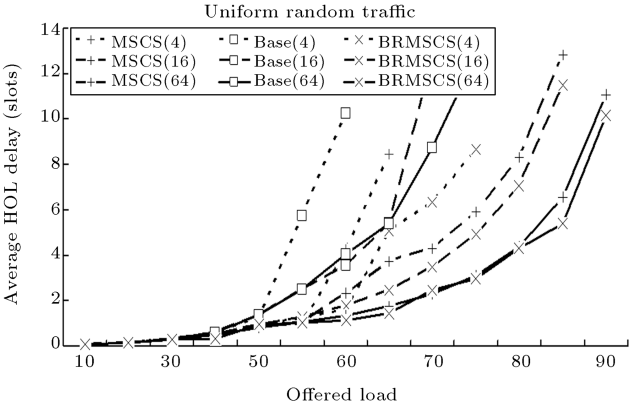

Figure 8. The average HOL delay under uniform random traffic.

Multicast Random Traffic

Figures 9 to 14 illustrate the simulation results of BRMSCS, MSCS and base schedulers under multicast random traffic with $NC = 2$, 4 or 8. Each incoming multicast cell generates copies, according to independent Bernoulli trails [13]. Obviously, the larger the search depth, the better the delay performance and the lower the overbooking rate. As the NC increases,

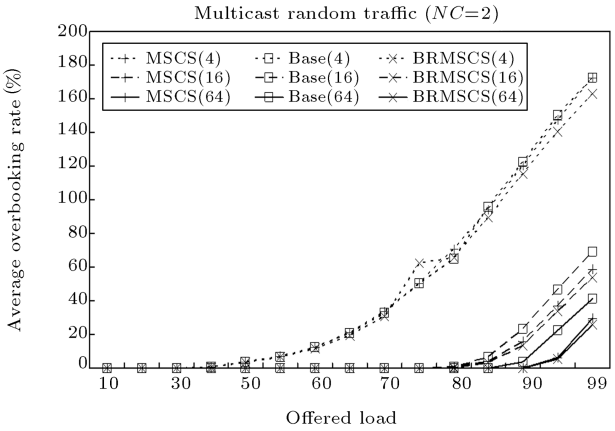

Figure 9. The average overbooking rate under multicast random traffic ($NC = 2$).

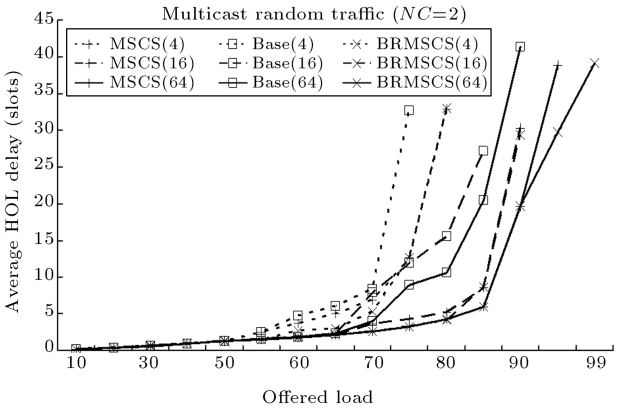

Figure 10. The average HOL delay under multicast random traffic ($NC = 2$).

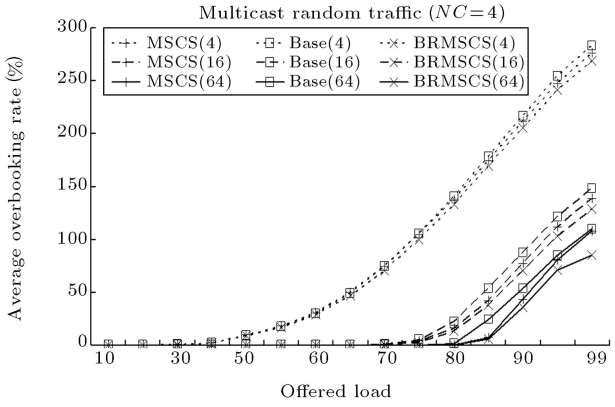

Figure 11. The average overbooking rate under multicast random traffic ($NC = 4$).

the BRMSCS and MSCS perform similarly under a low offered load; however, BRMSCS performs better than MSCS under a high offered load.

Both of these schedulers also perform better than the base scheduler. These results also show that if we want to get a higher performance, both schedulers must search more deeply.

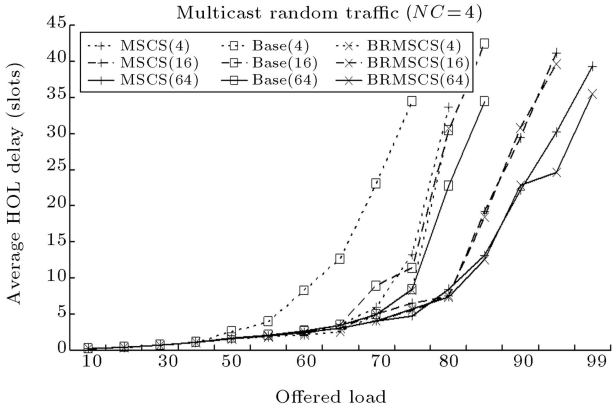

Figure 12. The average HOL delay under multicast random traffic ($NC = 4$).

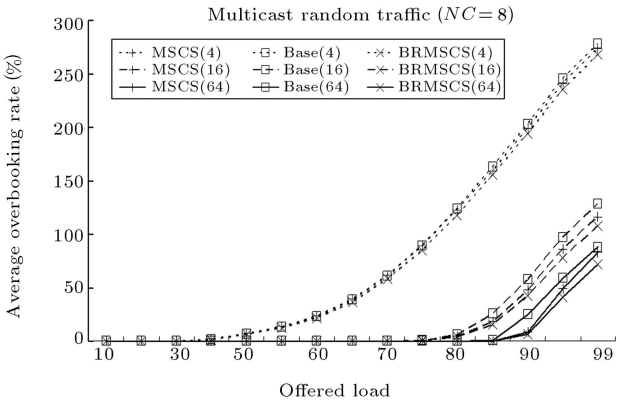

Figure 13. The average overbooking rate under multicast random traffic ($NC = 8$).

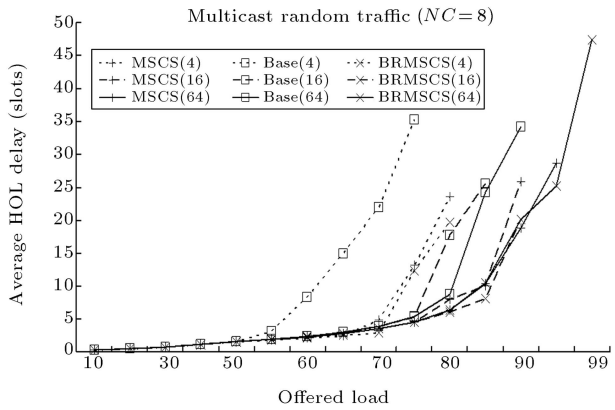

Figure 14. The average HOL delay under multicast random traffic ($NC = 8$).

Multicast Bursty Traffic

Figures 15 to 18 illustrate the simulation results of BRMSCS, MSCS and base schedulers under multicast bursty traffic with $NC = 4$ and 8. Obviously, when the NC increases, all three schedulers cannot perform well under bursty traffic and the overbooking rate grows quickly. Even by increasing the search depth to 64, the

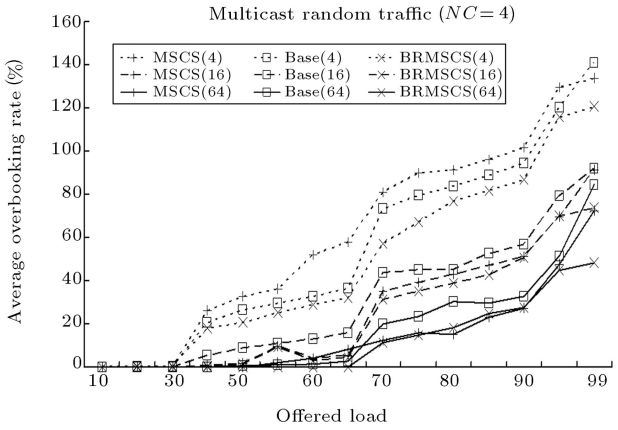

Figure 15. The average overbooking rate under multicast bursty traffic ($NC = 4$).

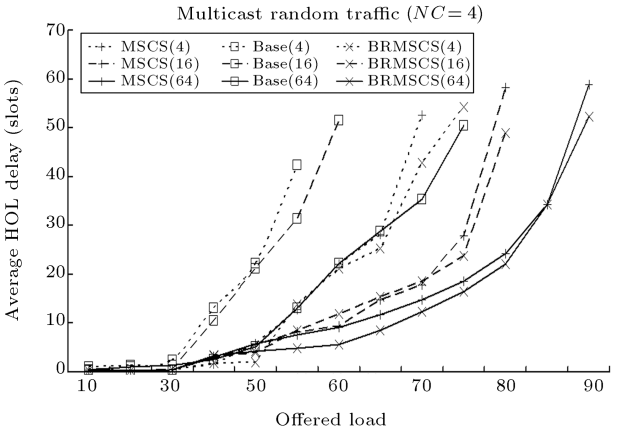

Figure 16. The average HOL delay under multicast bursty traffic ($NC = 4$).

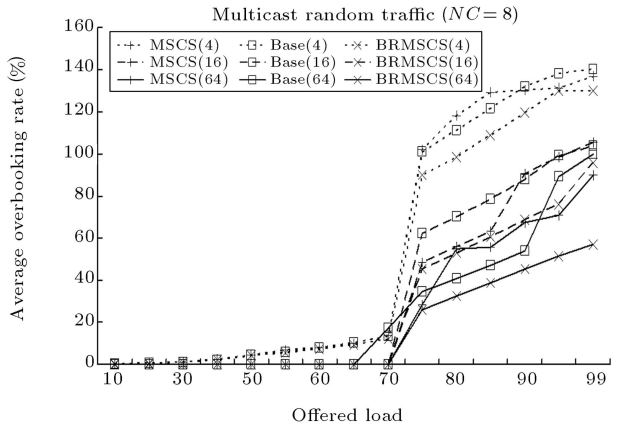

Figure 17. The average overbooking rate under multicast bursty traffic ($NC = 8$).

performance is still bad. The main reason is that the incoming bursty traffic is segmented into multiple cells. Each cell will occupy one location of the scheduling table. When the scheduling table is full, it causes serious HOL blocking occurred in the FIFO address queue. However, the BRMSCS scheduler still has a

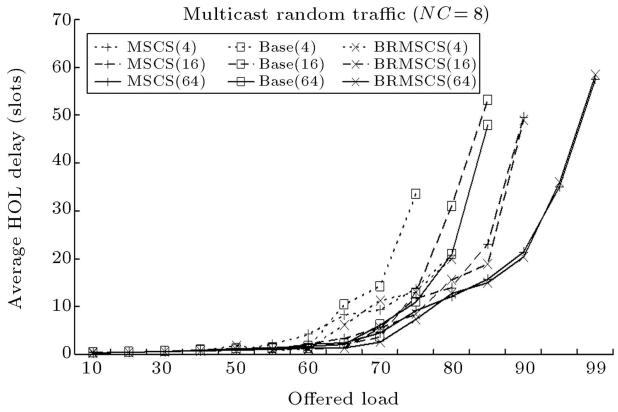

Figure 18. The average HOL delay under multicast bursty traffic ($NC = 8$).

better capacity for handling multicast bursty traffic. A number of interesting points can be found from the simulations:

- Both the BRMSCS and MSCS schedulers can perform well as the search depth increases;
- The base scheduler cannot perform as well as the BRMSCS and MSCS schedulers because it uses the scheduling table inefficiently;
- The lower the overbooking rates, the less HOL cell delay;
- Under multicast random traffic, the BRMSCS scheduler has a better performance than the MSCS scheduler. This is because it relieves the blocking situation in the scheduler;
- Under multicast bursty traffic, the scheduling table is occupied quickly. To overcome this problem, the scheduler must use the scheduling table more efficiently.

CONCLUSION

In this study, we proposed an efficient multicast switch called the BRMSCS switch. It is organized by a shared memory, a crossbar fabric and the BRMSCS algorithm. We simulated BRMSCS under three different traffic models and also compared BRMSCS to the MSCS and base schedulers. The simulation results show that the BRMSCS scheduler can relax any HOL blocking happening inside the scheduler and guarantee the output order is memory access conflict free. BRMSCS also performs better than MSCS under uniform multicast traffic. Both BRMSCS and MSCS perform poorly under multicast bursty traffic. However, BRMSCS still performs better than the MSCS scheduler. This is because it relieves the blocking situation in the scheduler.

ACKNOWLEDGMENT

This project was financially supported by the National Science Council under Grant No. NSC 96-2221-E224-006.

REFERENCES

1. Karol, M., Hluchyj, M. and Morgan, S. "Input versus output queuing on a space-division packet switch", *IEEE Transactions on Communications*, **35**(12), pp. 1347-1356 (1987).
2. McKeown, N. and Anderson, T.E. "A quantitative comparison of scheduling algorithms for input-queued switches", *Computer Networks and ISDN Systems*, **30**(24), pp. 2309-2326 (1998).
3. McKeown, N. "A fast switched backplane for a gigabit switched router", *Business Communications Review*, **27**(12) (1997).
4. Chao, J. "Saturn: A terabit packet switch using dual round-robin", *IEEE Comm. Mag.*, **38**(12), pp. 78-84 (2000).
5. Li, Y., Panwar, S. and Chao, J. "The dual round-robin matching switch with exhaustive service", *IEEE Workshop on High Performance Switching and Routing*, pp. 58-62 (2002).
6. Li, Y., Panwar, S. and Chao, J. "Performance analysis of a dual round Robin matching switch with exhaustive service", *IEEE Globecom*, **3**, pp. 2292-2295 (2002).
7. Lee, H.S. "A multicast switching algorithm based on iSLIP", *International Technical Conference on Circuits/Systems Computers and Communications*, pp. 1011-1014 (2002).
8. Prabhakar, B. and McKeown, N. "Designing a multicast switch scheduler", *Proceedings of the 33rd Annual Allerton Conference on Comm., Control, and Computing*, pp. 984-993 (1995).
9. Chen, W.T., Huang, C.F, Chang, Y.L. and Hwang, W.Y. "An efficient cell- scheduling algorithm for multicast ATM switching system", *IEEE/ACM Tran. on Networking*, **8**(8), pp. 517-525 (2000).
10. Minkenberg, C. "Integrating unicast and multicast traffic scheduling in a combined input- and output-queued packet-switching system", *Computer Communications and Networks*, pp. 127-134 (2000).
11. Bianco, A., Giaccone, P., Leonardi, E., Neri, F. and Piglione, C. "On the number of input queues to efficiently support multicast traffic in input queued switches", *IEEE Workshop on High Performance Switching and Routing*, pp. 111-116 (2003).
12. Gupta, P. "Scheduling in input queued switches: A survey", [Online] Available: http://klamath.stanford.edu/~pankaj/research.html (1996).
13. Song, M., Zhan, X. and Song, J. "An efficient queuing scheme for multicast packet switching routers", *IEEE Int'l Conference on Computer Supported Cooperative Work*, **1**, pp. 572-577 (2004)