Multi-Level Round-Robin Multicast Scheduling with Look-Ahead Mechanism

Yu, Hao; Ruepp, Sarah Renée; Berger, Michael Stübert

Published in:
2011 IEEE International Conference on Communications (ICC)

Link to article, DOI:
10.1109/icc.2011.5962481

Publication date:
2011

Multi-Level Round-Robin Multicast Scheduling with Look-Ahead Mechanism

Hao Yu, *Student Member, IEEE* Sarah Ruepp, and Michael S. Berger, *Member, IEEE*

***Abstract*—In this paper, we propose a multi-level round-robin multicast scheduling (*MLRRMS*) algorithm with look-ahead (*LA*) mechanism for *N*×*N* input-queued switches. Fan-out splitting is applied, where a multicast cell can be transferred to all its destinations over any number of cell times. The scheduler constructs the *Traffic Matrix* before each cell transmission based on the fan-out vectors of the cells in the queues. A scheduling pointer independently moves on each column of the *Traffic Matrix* in a round-robin fashion and returns the decision to the *Decision Matrix*. The *sync* procedure is carried out to reduce the unnecessary transmissions of a cell. The look-ahead mechanism is executed to reduce the head-of-line blocking problem resulting in increased the throughput and reduced cell delay.**

***Index Terms*—Multicast, switching, scheduling, input-queued switch, round robin**

I. Introduction

THE popularity of bandwidth-intensive services, e.g. IPTV, video conferencing, and telepresence, have placed a great demand on the multicast switching technology because multicast is able to deliver the traffic in a resource-efficient manner. However, resource contentions are more likely to occur if the packets are bound for multiple destinations. Based on various switch architectures, abundant literature has come up with multicast scheduling algorithms [1], [2], [4]-[11] to solve the resource contentions and increase the throughput. TATRA [1] is proposed based on the input-queued (IQ) architecture with first-in-first-out (FIFO) queuing discipline. The algorithm provides good fairness and efficiency in terms of high throughput and low latency but is too complex to implement. As a replacement, the weighted-based algorithm (WBA) [1] assigns weights to the new cell at the head-of-line (HOL) position based on the cell age and the fan-out in each input queue. Outputs grant permissions to the input with the highest weight. In [2], we proposed an efficient method to schedule multicast traffic based on the same FIFO IQ architecture. A process called *sync* is carried out to reduce the unnecessary transmissions of cells caused by the independent scheduling process. However, operating on the HOL cells suffers from the HOL blocking problem and fails to reach a high throughput [3].

To eliminate the HOL blocking problem and increase the throughput, output-queuing (OQ) can be used. But due to the poor scalability where the internal speed-up should be N times the line speed, where N denotes the number of inputs, OQ is not suitable for high-speed or large-scale switches. Using a non-FIFO IQ architecture, e.g. virtual output queuing (VOQ), the switch is able to reduce the HOL blocking problem without increasing the speed-up. However, $2^N - 1$ queues are required for multicast traffic in each input, which dramatically reduce the scalability. FIFOMS [4] and CMF [5] utilize the VOQ architecture for unicast and generate several address cells and one data cell for each arriving cell and store them in separate queues. For each destination in the fan-out set, an address cell is generated, which means the total number of queues for multicasting in an $N \times N$ switch is N^2. The bottleneck of such architecture is the unit which is responsible for generating data and address cells, which may require high complexity and speed-up.

In this paper, we focus on the FIFO IQ switch structure and propose a method with low implementation effort yet high performance to handle multicast traffic with a look-ahead (*LA*) mechanism to alleviate the HOL blocking problem described in [2].

II. Background

A. Assumed Switch Architecture

We consider an $N \times N$ input-queued switch due to the fact that an input and an output port usually reside in pair on the same line card. Within the switch, arriving packets are fragmented into fixed-size multicast cells and stored in the FIFO queues before traversing the switch fabric. Packets are reassembled at the output ports. Sufficient buffer capacities are assumed so that no cell loss occurs due to buffer overflow.

Any multicast cell is characterized by its fan-out set, which is the set of output ports for which the cell is bound. As a simple example shown in Fig. 1, input 0 has a cell at the head of the queue bounded for outputs {2, 3, 8}, and fan-out set can thus be expressed as {2, 3, 8}. We consider the case where *fan-out splitting* [6] is applied so that copies of multicast cells can be delivered to output ports over any number of cell times. Unless all the destinations in the fan-out set are reached, the cell is not removed but remains in the queue. A multicast scheduler makes scheduling decisions prior to each cell time and grants cell transmissions accordingly. We assume that the scheduler is able to examine the cells stored deeper in the queues and that it is capable of sending them to the corresponding outputs.

Manuscript received September 20, 2010. This work was supported in part by the Danish National Advanced Technology Foundation in the project *The Road to 100 Gigabit Ethernet*.

Hao Yu, Sarah Ruepp, and Michael S. Berger are with the Technical University of Denmark, Kgs. Lyngby, 2800, Denmark. (emails: {haoyu, srru, msbe}@fotonik.dtu.dk)

B. Definitions

We define several terms used in the scheduling algorithm throughout the paper.

Def. 1 (Maximum Look-Ahead Depth): The *maximum look-ahead depth*, *L*, is defined as the limit of the number of cells that the scheduler is able to examine further into the queue. $L = 0$ means that the switch only operates on the HOL cells, while $L = l$ indicates that the switch can look up to l cells after the HOL cell.

Def. 2 (Cell Position): The *cell position*, *p*, is defined as the position of a cell in the queue. The cell at the HOL of the queue has $p = 0$.

Def. 3 (Fan-out Vector): A *fan-out vector* is used to indicate the fan-out set carried by a multicast cell in input i at position p, and is denoted as $f^{(i,p)} \triangleq \langle f_k^{(i,p)} \rangle, k = 0,1,\dots,N-1, i = 0,1,\dots,N-1, p = 0,1,\dots,L, f_k^{i,p} \in \{0,1\}$. $f_k^{(i,p)} = 0$ indicates that output k is not in the fan-out set of the cell and $f_k^{(i,p)} = 1$ indicates the opposite. The cardinality of the fan-out set thus becomes $\left|f^{(i,p)}\right| \triangleq \sum_0^{N-1} f_k^{(i,p)}$.

Def. 4 (Traffic Matrix): The *Traffic Matrix* is an $N \times N$ matrix constructed by the scheduler based on the fan-out vectors of the cells in the position p of each input i before a cell transmission. It is denoted as $\boldsymbol{T}^{(p)} = \left(T_{i,j}^{(p)}\right), i = 0,1,\dots,N-1, j = 0,1,\dots,N-1$. Obviously, we have $T_{i,j}^{(p)} = f_j^{(i,p)}, \forall i,j,p$. And we define $T_{i,j}^{(p)} = 0, \forall j,p$ if input queue i is empty.

Def. 5 (Decision Matrix): The *Decision Matrix* is an $N \times N$ matrix denoted as $\boldsymbol{D}^{(p)} = \left(D_{i,j}^{(p)}\right), i = 0,1,\dots,N-1, j = 0,1,\dots,N-1, D_{i,j}^{(p)} \in \{0,1\}$. This matrix contains the scheduling decisions for each output j with $D_{i,j}^{(p)} = 1$ indicating that a copy of the cell in input i at position p will be transferred to output j and $D_{i,j}^{(p)} = 0$ meaning that no copy will be sent to output j. We can know that $0 \leq \sum_j D_{i,j}^{(p)} \leq 1, \forall j$

Def. 6 (Set of Decision Matrices): The *Set of Decision Matrices* is defined as $\boldsymbol{\Delta} = \left\{\boldsymbol{D}^{(0)}, \boldsymbol{D}^{(1)}, \dots, \boldsymbol{D}^{(L)}\right\}$. It contains up to *L* decision matrices. Multicast cells are released by the scheduler according to the decision matrices stored in $\boldsymbol{\Delta}$.

Def. 7 (Assistant Matrix): The *Assistant Matrix* is an $N \times N$ matrix denoted as $\boldsymbol{A}^{(p)} = \left(A_{i,j}^{(p)}\right), i = 0,1,\dots,N-1, j = 0,1,\dots,N-1, A_{i,j}^{(p)} \in \{0,1\}$. This matrix is used to help generate $\boldsymbol{D}^{(p)}, p > 0$.

Def. 8 (Cross Disable Mark $^\circ\boldsymbol{X}$): We define $^\circ\boldsymbol{X}$ as a matrix transform mark for the sake of convenience where $\boldsymbol{X} = \left(X_{i,j}\right), X_{i,j} \in \{0,1\}$ is the in-operation matrix. If we have $\boldsymbol{Y} = {}^\circ\boldsymbol{X}$, first let $\boldsymbol{Y} = \boldsymbol{O}, (Y_{i,j} = 0, \forall i,j)$ with the same dimensions as $\boldsymbol{X}$, and if $X_{k,l} = 1$, then $Y_{k,j} = 1, Y_{i,l} = 1, \forall i,j$.

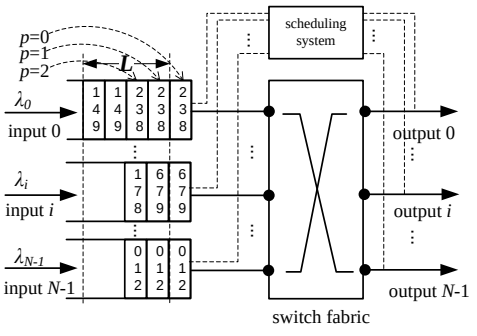


Fig. 1. *N*×*N* input-queued multicast switch with FIFO architecture.

III. FIFO-based Multi-Level Round Robin Multicast Scheduling

We here describe the proposed multicast scheduling algorithm in detail based on the previous definitions. Before each cell transmission time, the scheduler executes the following procedures and accordingly releases cells after completion.

Initial condition: $p = 0$, $\boldsymbol{\Delta} = \emptyset$, and $\boldsymbol{D}^{-1} = \boldsymbol{O}$ $(D_{i,j}^{-1} = 0, \forall i,j)$

1) The scheduler examines the fan-out vector $f^{(i,p)}$ of the cell in input *i* at position *p* for all inputs to construct $\boldsymbol{T}^{(p)}$.

2) $\boldsymbol{A}^{(p)} = \boldsymbol{T}^{(p)} - {}^\circ\left(\sum_{p=0}^{|\boldsymbol{\Delta}|} \boldsymbol{D}^{(p-1)}\right)$, and if $A_{i,j}^{(p)} < 0$, then set $A_{i,j}^{(p)}$ to 0, $\forall i,j$.

3) The round-robin scheduling algorithm is independently executed on each non-zero column of $\boldsymbol{A}^{(p)}$. Only one element in a column can be selected due to the constraint of one output port only being able to one transmission during a cell time. The results thus form $\boldsymbol{D}^{(p)}$.

4) The sync [2] procedure is carried out on $\boldsymbol{D}^{(p)}$ to reduce the unnecessary multiple transmissions of cells: if column y plays the role of dictator during this cell time and $D_{x,y}^{(p)} = 1$, and $\forall j \neq y$, $A_{x,j}^{(p)} = 1$ and $D_{x,j}^{(p)} \neq 1$, then let $D_{x,j}^{(p)} = 1$ and $D_{i,j}^{(p)} = 0, \forall i \neq x$. The scheduler stores the refined $\boldsymbol{D}^{(p)}$ to $\boldsymbol{\Delta}$, i.e. $\boldsymbol{D}^{(p)} \to \boldsymbol{\Delta}$.

5) If a zero column is found in $\sum_{p=0}^{|\boldsymbol{\Delta}|} \boldsymbol{D}^{(p-1)}$, check the queue size of each unreserved input, which is the corresponding row in $\sum_{p=0}^{|\boldsymbol{\Delta}|} \boldsymbol{D}^{(p-1)}$. If the queue size is larger than $p+1$, and $p+1 \leq L$, increase p with 1 and go to step 1. Otherwise, continue to step 6.

6) The scheduler should examine $\boldsymbol{\Delta}$ and release multicast cells at particular positions from input queues according to each $\boldsymbol{D}^p$. If the fan-out set of a cell becomes empty after the service, the cell will be removed from the queue. Otherwise, the cell remains with a new fan-out set.

IV. Complexity and Performance Analysis

A. Implementation of the MLRRMS

The MLRRMS algorithm is in essence designed to be implemented in a parallel and distributed fashion and require no linear scan. Since we allow the switch to look ahead into the queues, iterations will occur to increase the output utilization for each cell time as described in previous section. The position parameter p is incremented by 1 at the end of iteration.

At the beginning of each cell time, the scheduling process begins. All inputs and outputs are initially free and $p = 0$. Only those inputs and outputs not reserved at the end of one iteration are eligible for the next. The scheduling process below operates in parallel on each outputs and inputs.

i) Submission

Each free input submits to every free output for which it

has a multicast cell at p bounded. The outputs that have received requests from the inputs will appear in a round-robin schedule of the dictator assignment.

ii) Dictator Assignment

A dictator arbiter chooses the output that appears next in a round-robin schedule starting from the highest priority element to be the dictator over other outputs for p. The dictator pointer $a^{(p)}$ to the highest priority element of the round-robin schedule is incremented (modulo N) to one position beyond the current dictator after the assignment.

iii) Decision

If a free output receives any request, it chooses the one that appears next in a round-robin schedule starting from the highest priority element. The output notifies each input whether its request is selected as the decision. The decision pointer $d^{(p)}$ to the highest priority element of the round-robin schedule in incremented (modulo N) to one location beyond the selected input *if and only if the output receives a cell from its selected input*.

iv) Sync

If one input receives a decision from the dictator, it invalidates the decisions of other outputs which are contained in its request set, and keeps its own decision valid. The input that has an invalid decision loses permission to transmit cells. Only the input that has a valid decision is eligible for sending cells.

This process iterates until either all the outputs are reserved or the maximum *LA* depth is reached. The effect of the algorithm and the *sync* mechanism in this framework are the same as described in the previous section but the complexity is reduced.

B. Heuristic Analysis of the Look-Ahead (LA) Mechanism

The *LA* mechanism is only performed when the output ports are not fully reserved. There are potentially two reasons to cause the partial occupancy: 1) the HOL blocking, and 2) the traffic pattern. Obviously, there is nothing to improve if it is the traffic pattern that causes the partial occupancy. On the other hand, the HOL blocking phenomenon may be the cause and therefore *LA* mechanism will be able to reduce the degradation.

Assume that each multicast cell has the same probability of being bound to each output:

$$P\left(f_k^{(i,p)} = 1\right) = \delta, \forall i, k, p \tag{1}$$

The probability of a column j in $\boldsymbol{T}^{(p)}$ being a zero column or an idle output:

$$P\left(T_{i,j}^{(p)} = 0\right) = (1-\delta)^N = \theta, \forall i, j, p \tag{2}$$

Let the random variable X be the number of zero columns found in $\boldsymbol{T}^{(0)}$. Then we have the probability mass funcation (p.m.f). of X:

$$P(X = x) = \binom{N}{x}\theta^x(1-\theta)^{N-x}, x \in \{0,1,2,\dots,N-1\} \tag{3}$$

Let S be the random variable of the *LA* depth which the switch should examines to find a cell to send to an idle output. Since reserved inputs are not considered in the *LA* process, we can have the lower bound of the p.m.f. of S:

$$P(S = s) \geq \theta^{s-1}(1-\theta), s \in \{1,2,\dots\} \tag{4}$$

and the c.p.f:

$$P(S \leq s) \geq 1 - \theta^s, s \in \{1,2,\dots\} \tag{5}$$

In order to utilize all the idle outputs, the *LA* depth of the system, $\hat{\mathrm{S}}$, should be the maximum of each zero column's *LA* depth:

$$\hat{\mathrm{S}} = \max(\mathrm{S}_1, \mathrm{S}_2, \dots, \mathrm{S}_\mathrm{x}) \tag{6}$$

We can find the cumulative probability of all the x zero columns becoming non-zero (output utilization $U = 100\%$) after $\hat{S} = L$ under the assumption that each column is independent:

$$\begin{aligned} h(x,L) &\geq P\left(X = x, \hat{S} \leq L\right) \\ &= F_{S_1}(L)F_{S_2}(L)\dots F_{S_x}(L) \cdot P(X = x) \\ &= (1-\theta^L)^x \cdot P(X = x) \end{aligned} \tag{7}$$

and the p.g.f. of $h(x,L)$ with L being a constant is:

$$H(z,L) \geq [1 - \theta + \theta(1-\theta^L)z]^N \tag{8}$$

We can thus derive the full-utilization probability that the outputs are fully utilized after a *LA* depth of L:

$$A(L) = H(z,L)|_{z=1} \geq (1-\theta^{L+1})^N \tag{9}$$

In addition to the full-utilization probability, we are also interested in the relationship between the maximum *LA* depth L and the output utilization U. The purpose of the *LA* mechanism, as explained previously, is to reduce the number of idle outputs so that the output utilization can be increased and the HOL blocking problem can be reduced. We know that the probability that an output remains idle after a *LA* depth of L has a lower bound of δ^{L+1}. Then we can derive the lower bound of the probability that there exists Y idle outputs after searching L cells:

$$g(y,L) \geq P\left(Y = y, \hat{S} \leq L\right) = \binom{N}{y}(\theta^{L+1})^y(1-\theta^{L+1})^{N-y} \tag{10}$$

From above we know the probability that no idle output remains after a *LA* depth of L is $g(0,L) \geq (1-\theta^{L+1})^N$, which corresponds to (9). We can therefore derive the lowe bound of the probability of the output utilization L:

$$P\left(U = \frac{N-Y}{N}\right) \geq \binom{N}{y}(\theta^{L+1})^y(1-\theta^{L+1})^{N-y} \tag{11}$$

C. Simulation Result

We compare the proposed MLRRMS with WBA [1] and

FIFOMS [4] by simulations carried out in OPNET Modeler [11]. We assume that the multicast traffic to each input is independent and $N = 8$ for the simulated switch. To compare the performance of the algorithms in various traffic conditions, we consider Bernoulli traffic and burst traffic with different fan-out schemes.

We first apply the Bernoulli traffic to the switch. A cell arrives at an input with a probability of q and $\theta = 0.5$, which results in $E(|f|) = 4$ for the 8×8 switch. The offered load is calculated as $\lambda = q \cdot E(|f|)$.

Fig. 2 shows the simulation results when Bernoulli traffic is applied. Fig. 2(a) compares the average multicast delays under various traffic loads. A multicast cell is stored in the queue until all the destinations in its fan-out set are reached. The multicast delay of a cell is calculated as the cell times that the cell stays in the queue until it is removed. Since the WBA and MLRRMS (L=0) both operate only on the HOL cells, they become unstable as the offered load increases. With looking ahead maximum 1 cell further, the MLRRMS (L=1) has demonstrated a significant improvement of the multicast delay compared to MLRRMS (L=0) and WBA. As L increases, we can observe more improvement from MLRRMS (L=2) and (L=10) but we can also discover that the marginal improvement is decreasing. Among all, FIFOMS has the lowest delay because it uses the VOQ architecture to handle the multicast traffic with a total number of queues of N^2.

We examine the average queue size per input, including the cell in service, in Fig 2(b). Since MLRRMS (L=0) and WBA operate only on the HOL cells, they both suffer from the HOL blocking problem and have the highest average queue size compared to other schemes. We can again observe a significant improvement of MLRRMS (L=1).

In Fig. 2(c), we examine the average LA depth. For MLRRMS (L=0), the average LA depth is always 0. For MLRRMS (L=1), it allows the switch to search up to 1 cell deeper in the queues. When the traffic load is heavy, the average LA depth is almost the same as L. For MLRMMS (L=2) and (L=10), the average LA depth under heavy load is less than its L value revealing that the switch does not leverage its full potential. The average LA depth of MLRRMS (L=10) is approximately 7 when the switch is heavily loaded, indicating that the added implementation complexity of the switch is obsolete and the performance improvement is nonlinear. Both MLRRMS (L=1) and (L=2) begin to converge after $\lambda = 0.9$ because the queue size begin to become larger than the L values.

We further apply bursty traffic, or *Correlated Arrival Process*, which has two states, *busy* and *idle*. Cells are generated only in the *busy* state. The process stays in each state for a random number of cell times following the geometric distribution with mean values of $E[B]$ and $E[I]$, respectively. The arrival rate is calculated as $\lambda = E[B]/(E[B] + E[I])$. Since the traffic arrives at the switch in bursts, two modes of fan-out schemes can be applied, cell-based and burst-based. In cell-based fan-out mode, the fan-out vector is independently generated for each cell. And in burst-based mode, the fan-out vector is independently generated for each burst of cells, each burst of cells having the same fan-out vectors. $E[B] = 16$ [1] and each cell arrives at an input with $\theta = 0.5$, which results in $E(|f|) = 4$ for the 8×8 switch.

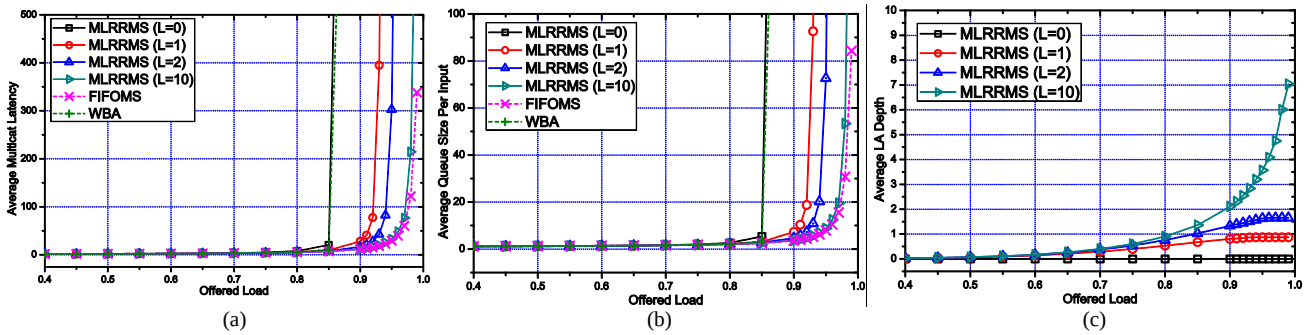
Fig. 3 compares the performances under bursty traffic with cell-based fan-out mode. In Fig. 3(a), the average multicast latency of all the scheduling schemes increases. WBA and MLRRMS (L=0) have the largest delay compared to others. With looking up to 1 cell, MLRRMS (L=1) has reduced the multicast latency dramatically. MLRRMS (L=2) does not provide the same level of improvement compared to the complexity it adds to the switch. The delay performances of MLRRMS (L=10) and FIFOMS are nearly the same under heavy traffic loads. In Fig. 3(b), the average queue size per input is examined. Due to the bursty characteristic of the traffic, the average queue size is larger for light traffic load than when the Bernoulli traffic is applied. Since the fan-out scheme is cell-based, the LA mechanism can still reduce the HOL blocking problem and increase the throughput of the switch. This can be observed in the improvement of MLRRMS (L=1) in both Fig. 3 (a) and (b). In Fig. 3(c), we can observe similar results as in Fig. 2(c) but the average LA depth of MLRRMS (L=2) converges earlier than in Fig. 2(c). This is because the bursty traffic leads to larger queue size under light load than the Bernoulli traffic.

Fig. 4 shows the results under bursty traffic with burst-based fan-out mode. Each burst of cells has the same fan-out vector which means looking ahead a limited number of cells in the queues is enough for the switch to alleviate the HOL blocking problem. A deeper searching should be carried out for this type of traffic. Among other schemes, WBA has the largest multicast latency as shown in Fig. 4(a). FIFOMS performs better than MLRRMS with different L's. Focusing on the MLRRMS group, we can see the improvement of MLRRMS (L=1) is not as significant as in Fig. 2(a) and Fig. 3(a), and MLRRMS (L=0), (L=1) and (L=2) have similar multicast delay. This phenomenon corresponds to the reason stated previously, i.e. the burst-based mode requires a larger L, since L=1 or 2 is not enough to include most possibilities of the burst distribution. MLRRMS (L=10) is able to generate a reduce the latency significantly because of the large L.

In Fig. 4(b), we can obtain similar results as in Fig. 4(a). All algorithms suffer from performance degradation due to the traffic pattern. The average LA depths of MLRRMS with different L's under burst-based mode are shown in Fig. 4(c). MLRRMS (L=1) and (L=2) begin to converge when the offered load λ reaches 0.6 due to the traffic pattern. The MLRMMS (L=10) also converges at $\lambda = 0.8$ which indicates that the traffic of burst-based fan-out mode has put a greater demand on the possible LA depth than the cell-based scheme and the queue size is always larger than 10.

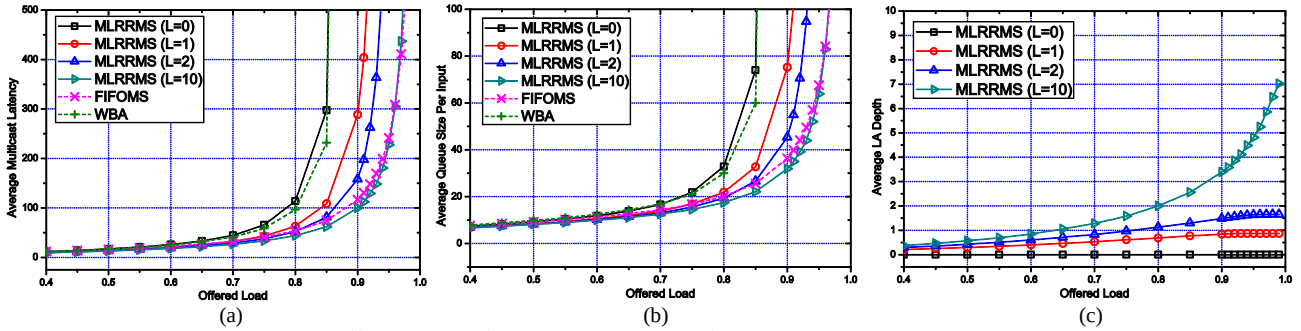
V. Conclusion

In this paper, we propose an efficient multi-level round-robin based multicast scheduling (*MLRRMS*) algorithm with a look-ahead (*LA*) mechanism for $N \times N$ switches with the FIFO-IQ architecture. As discussed, the *LA* and the *sync* mechanism consisting of matrix operations used in MLRRMS can be implemented in a parallel fashion with a low time complexity. Under varying traffic conditions, MLRRMS with $L = 1$ outperforms WBA and gains the



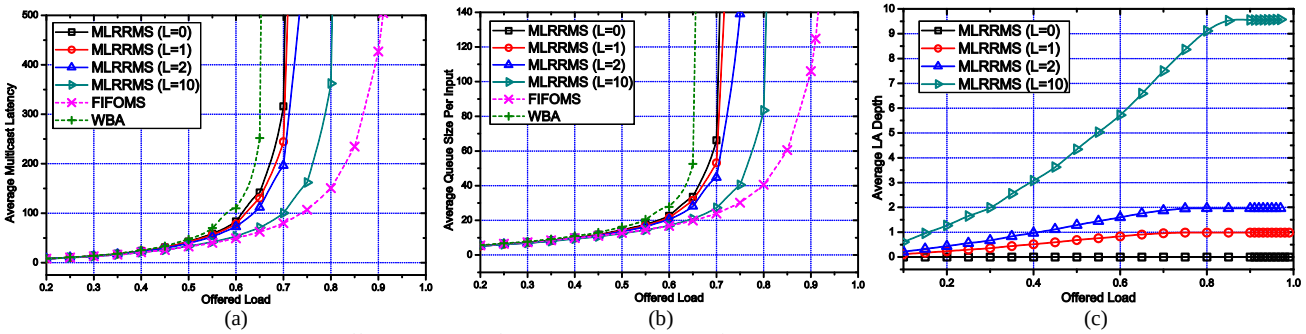
(a) (b) (c)

Fig. 2. Simulation results under Bernoulli traffic with N=8, θ=0.5. (a) Average multicast latency. (b) Average queue size per input (c) Average look-ahead depth.



(a) (b) (c)

Fig. 3. Simulation results under bursty traffic (cell-based fan-out mode) with N=8, θ=0.5. (a) Average multicast latency. (b) Average queue size per input (c) Average look-ahead depth.



(a) (b) (c)

Fig. 4. Simulation results under bursty traffic (burst-based fan-out mode) with N=8, θ=0.5. (a) Average multicast latency. (b) Average queue size per input (c) Average look-ahead depth.

largest performance improvement. The HOL blocking problem is alleviated by the *LA* mechanism. With a larger *L*, the algorithm performs close to FIFOMS which uses the VOQ structure, but the marginal performance improvement obtained does not justify the introduced implementation complexity. Being able to search up to 1 cell stored deeper in the queues for the switch, i.e. being capable of processing 2 cells at the head of the queues within one cell time, instead of creating multiple queues for each input is able to provide the largest performance improvement in terms of multicast delay and average queue size.

REFERENCES

[1] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches", *IEEE Journal on Selected Areas in Commun.*, vol. 15, no. 5, Jun. 1997.

[2] H. Yu, S. Ruepp, and M. S. Berger, "A novel round-robin based multicast scheduling algorithm for 100 gigabit Ethernet switches", in *Proc. IEEE INFOCOM SW*, 2010.

[3] M. J. Karol, M. G. Hluchyj, S. P. Morgan, "Input versus output queueing on a space-division packet switch", *IEEE Trans. Commun*, vol. com-35, no. 12, Dec. 1987.

[4] D. Pan, and Y. Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches", *IEEE Trans. Comput.*, vol. 54, no. 10, Oct. 2005.

[5] D. Pan, and Y. Yang, "Bandwidth guaranteed multicast scheduling for virtual output queued packet switches", *J. Parallel Distrib. Comput.*, vol. 69, issue. 12, pp. 939-949, 2009.

[6] J.F. Hayes, R. Breault, and M. Mehmet-Ali, "Performance analysis of a multicast switch", *IEEE Trans. Commun.*, vol. 39, pp. 581-587, Apr. 1991

[7] S. Sun, S. He, Y. Zheng, and W. Gao, "Multicast scheduling in buffered crossbar switches with multiple input queues", in *Proc. HPSR*, 2005.

[8] W. Chen, C. Huang, Y. Chang, and W. Hwang, "An efficient cell-scheduling algorithm for multicast ATM switching system", *IEEE/ACM Trans. Networking*, vol. 8, no. 4, Aug. 2000.

[9] M. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet scheduling in input-queued cell-based swiches", in *Proc. IEEE INFOCOM*, 2001.

[10] S. Gupta, and A. Aziz, "Multicast scheduling for switches with multiple input queues", in *Proc. Symp. Hot Interconnects*, 2002.

[11] M. A. 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, Jun. 2003

[12] OPNET Modeler, available at: http://www.opnet.com