International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:10, 2015

Performance Analysis of OQSMS and MDDR Scheduling Algorithms for IQ Switches

K. Navaz, Kannan Balasubramanian

Abstract**—**Due to the increasing growth of internet users, the emerging applications of multicast are growing day by day and there is a requisite for the design of high-speed switches/routers. Huge amounts of effort have been done into the research area of multicast switch fabric design and algorithms. Different traffic scenarios are the influencing factor which affect the throughput and delay of the switch. The pointer based multicast scheduling algorithms are not performed well under non-uniform traffic conditions. In this work, performance of the switch has been analyzed by applying the advanced multicast scheduling algorithm OQSMS (Optimal Queue Selection Based Multicast Scheduling Algorithm), MDDR (Multicast Due Date Round-Robin Scheduling Algorithm) and MDRR (Multicast Dual Round-Robin Scheduling Algorithm). The results show that OQSMS achieves better switching performance than other algorithms under the uniform, non-uniform and bursty traffic conditions and it estimates optimal queue in each time slot so that it achieves maximum possible throughput.

Keywords**—**Multicast, Switch, Delay, Scheduling.

I. Introduction

MULTICASTING is the ability to provide point-to-multipoint connections. Driven by the Internet and its applications, such as video on demand (VOD), music on demand (MOD), teleconferencing, videoconferencing and distributed data processing, more and more communication services and applications will require that information from a source that delivers to multiple destinations.

In ancient switches, the input output ports communicated using a single shared bus. Consequently, this bus was a limitation, as not more than one pair of ports can communicate at a time. The classical crossbar switch overcame the bottleneck imposed by this shared bus architecture that restricted the use of N input-output port pairs in parallel. The crossbar switch is an NxN matrix of 2N buses, connecting input output ports. The switches and routers basically store, route and forward these packets before they reach the destination. One core functionality of such switches (a layer 2 switch, or a layer 3 IP router) is to transfer the packets from the input port to one of the output ports. This functionality, called switching, though appears simple, is such a challenging problem to solve at line rates that, there is a wealth of literature on this topic.

K. Navaz is Research Scholar with the Manonmaniam Sundaranar University, Tirunelveli, 627 012, Tamilnadu, India (e-mail: navazit@gmail.com)

Dr. Kannan Balasubramanian is Professor with the Mepco Schlenk Engineering College, Department of CSE, Sivakasi, 626 005, Tamilnadu, India (e-mail: kannanbala@mepcoeng.ac.in).

Multicasting will become an important feature for any switching network designed to support broadband integrated service digital networks (B-ISDN). Generally speaking, packet switch architectures can be divided into three major categories [10]: the shared memory packet switch, the shared medium packet switch and the space division packet switch. Theoretically, each of these three architecture types can be modified to support multicast. However, in shared memory and shared medium architectures, there is a scalability problem as the need for a high-speed memory or a bus greatly limits their use when the switch size grows larger.

A crossbar switch is a switch connecting multiple inputs to multiple outputs in a matrix manner. The crossbar constraints of an IQ switch require it to schedule packets to be transferred between inputs and outputs. The throughput and delay in IQ switch are heavily dependent on this scheduling decision. In past there has been a lot of research done to design multicast scheduling algorithms for IQ switches. In addition, people are more interested in sharing knowledge and information for various purposes. Innovations in information sharing are continuously accelerated to cater user needs in such environments. These motives have encouraged the construction of sophisticated environments for effective communication to deliver information. It is important to note that the applications for IP Multicast are not solely limited to the Internet. Multicast IP can also play an important role in large distributed commercial networks. The demand for network bandwidth is very essential and many of the networking applications require high speed switching for multicast traffic at the switch/router level to preserve network bandwidth. It causes an incrementing interest in the input-queued switches. A switch consists of three components: 1) input queues for cells arriving at the input links 2) output queues for cells exit on output links 3) A switch fabric for transferring cells from the inputs to the desired outputs. LAN and Asynchronous Transfer Mode (ATM) switches are considered as a high performing internetworking protocol and uses a crossbar switch based on switched backplane. Further, these systems use the input queues for holding packets which are waiting to traverse through the switch fabric. Thus, it is known that the first in first out (FIFO) input queues can be used to maintain packets. A scheduling algorithm is utilized to configure the crossbar switch to decide the order in which packets will be accommodated.

Many integrated scheduling algorithms have been proposed earlier. They have been mainly proposed for input queued crossbar switching architecture but multicast scheduling, mainly concerns how to transmit as many cells as possible from input to output. In the unicast traffic, Head-of-Line (HOL) blocking quandary occurs that is induced by first-in-first-out (FIFO) queue which gets avoided by utilizing virtual output queuing (VOQ) technique. Here, in this type of technique every single input maintains a separate queue for each output [7].

Numerous unicast scheduling algorithms have been proposed so far. iSLIP is the fast and efficient algorithm which has achieved 100% throughput in a single iteration for uniform traffic. In [10], MRR (Modified Round Robin Algorithm) proposes that it can show a performance equivalent to iSLIP, yet require less number of processing steps. Using multicast traffic [2], [5], we can avoid HOL blocking by utilizing 2N -1 queues for each input port in N×N switch. This type of queue architecture is called Multicast Virtual Output Queuing (MC-VOQ). However, in the medium/sizably voluminous switches, because of its low scalability, it is virtually not tackled. One such practical queuing scheme utilized for multicast switches is to assign a single FIFO queue at each input for all multicast traffic, however, the HOL blocking quandary limits the throughput. Whereas the other algorithms [1], [3] considered a circumscribed number of FIFO queues is maintained at each input to reduce the HOL blocking problem. Thus queuing architecture is denominated as k-MC-VOQ and performance of these multicast switches are analyzed theoretically [11], [14]. As the link speed grows dramatically, high speed switches will have less time to perform scheduling process. As a result, iterative schemes and high matching overhead would cause delay, matching overhead scales up very expeditiously, the link speed and the switch size increases, the requirement for simple and high performance switches becomes very essential.

In this paper, we analyze the average cell delay and throughput of the OQSMS, MDDR and MDRR algorithms for IQ switch under Bernoulli uniform and non-uniform traffic patterns. The rest of the paper is organized as follows. In Section II, related works on designing multicast scheduling algorithms are reviewed. In Section III, MDDR and OQSMS algorithms have been reviewed. In Section IV, Performance evaluation and result analysis have been presented. Finally, we conclude the paper in Section V.

II. Related Works

Most of the existing multicast switches [8], [9] require in-switch packet replication, and a sophisticated central scheduler to maximize performance of the switch. TATRA [9] is a multicast algorithm on single FIFO queue, where each input port has a single common queue for both unicast and multicast traffic. The central scheduler maintains the N virtual queues and each is destined for one output port. In each time slot, the head-of-line (HOL) packet of each input queue is scheduled to join different virtual queues according to its destination output ports. Fan-out splitting [4], which sanctions a multicast packet to be sent to a subset of its outputs, is adopted to increment the throughput of the switch. However, TATRA suffers from serious HOL blocking because of its single queue nature. TATRA avoids starvation but is additionally perplexed to implement a hardware due to heavy computations.

To minimize the HOL blocking, multiple dedicated multicast queues have been utilized in [3] and [13]. In [13], each input port has a set of multicast queues. When a multicast packet arrives, it selects one of the multicast queues to join according to its load balancing policy. In each time slot, the scheduling priority is given to either a unicast packet or a multicast packet. According to the accommodation ratio of the two types of traffic. An iterative scheduling algorithm is adopted to maximize the switch throughput.

ESLIP [6] adopts the VOQ structure to buffer unicast packets and puts all the multicast packets in a special single queue at each input port. It utilizes a variant of the iSLIP algorithm to schedule mixed unicast and multicast traffic. As can be expected, ESLIP eliminates the HOL blocking for unicast traffic, but not for multicast traffic. In an extreme situation, where all the incoming packets are multicast packets, ESLIP cannot benefit from the VOQ structure and it authentically works on the single input queued switch.

Multicast packet split scheme is proposed in [3] for further reducing the HOL blocking problem. In [3], the set of output ports is divided into m non–overlapped subsets, and each input port maintains m unicast / multicast shared queues and each is dedicated to a subset of outputs. When a multicast packet arrives, if its fan-out set wholly fit in a queue, it will join the queue, otherwise, the multicast packet is divided into smaller ones (each with a modified fan-out set) to join multiple queues. Again, an iterative scheduler is adopted to maximize throughput.

FIFOMS [8] is an efficient multicast scheduling algorithm which is proposed to avoid HOL blocking. The basic idea is to discretely store unicast/multicast packets and memory addresses. FIFOMS uses common unicast VOQ's as pointer queues. More concretely, when a multicast packet with a fan-out of f (f = 1 for unicast packet) arrives, it is time stamped and stored in shared memory, and its memory address / pointer joins f different VOQ queues according to the fan-out set. In each time slot, the scheduling priority is given to pointers which are the unicast copies of a multicast packet with the most diminutive timestamp. Indeed, In Scheduling all multicast packets are "converted" into a unicast. At this end, the HOL blocking is completely eliminated. But in order to maximize switch throughput, in-switch packet replication is still utilized for sending multiple replicas of a multicast packet in the same slot. This is achieved by an iterative scheduling algorithm, which incurs considerable amount of communication overhead.

In [12], Multicast Dual Round Robin Scheduling Algorithm called MDRR, it is proposed to achieve maximum throughput with low-matching overhead. Here input schedulers are distributed to each input, and a global pointer 'g' is collectively maintained by all output schedulers. Each input scheduler has two priority pointers that guarantee high throughput: a primary pointer and a secondary pointer. MDRR needs more message transfer between the input and output ports in the request phase. It does not ensure a minimum delay compared with MaxService [3]. When the number of queues and the fan-out size (ef) increase MDRR could not obtain a maximum throughput than MaxService scheme done. Dual pointer utilization in the input ports are overhead here which takes longer execution time.

We compare the scheduling scheme called OQSMS with MDDR and MDRR. OQSMS achieves better switching performance than other algorithms under the admissible traffic conditions because OQSMS will estimate optimal queue selection based on more queue combinations so that it achieves maximum possible throughput. MDDR primarily minimizes the request overhead at the output ports and eliminates the dual pointer utilization in input ports. It shows that MDDR more preponderant than the MDRR algorithm.

III. Switch Architecture and Multicast Scheduling Algorithms

In this section, we give the packet switch architecture, MDDR and OQSMS multicast scheduling algorithms for input queued crossbar switch in detail.

A. Packet Switch Architecture

The scheduling algorithms are made for synchronous input-queued (IQ) switches. The fixed-size packet which is transmitted by the switch fabric is called cell. But only the fan-out splitting discipline is considered because the cells may deliver output over several cell times. Any multicast cell is characterized by its fan-out set, i.e., by using the set of outputs to which the cell is directed. We define the fan-out size ‘f’ as the number of destinations of a multicast cell. The NxN switch architecture is shown in Fig. 1. Let us assume NxN switch having N input ports and N output ports, and the fabric is connecting input ports and output ports for any time slot. A small number k of FIFO queues dedicated to multicast traffic is maintained at each input port. Q_{ij} is the jth queue in the ith input port. Arriving multicast cells are partitioned into the k queues according to the fan-out size. Each queue contains the multicast cells with fan-out sets. A scheduling algorithm does the arbitration between the N input ports and N output ports, obtained by solving the bipartite graph-matching problem. This matching is a collection of edges, from the set of non-empty input queues to the set of output ports. Such that each input is connected to at most N outputs and each output is connected to at most one input. In each time slot, Input ‘i’ is connected with set of output destinations. If the fan-out of the cell is completely served, a cell is removed from the corresponding queue to output destinations by properly configuring the non-blocking multicast switch fabric otherwise a cell is retained until all its destinations are served.

B. Multicast Due Date Round-Robin (MDDR)

In [16], MDDR multicast scheduling algorithm has been proposed. Input schedulers are distributed at each input and a global pointer g is collectively maintained by all the output schedulers. Each input maintains a Due Date to be sent. This due date is generated based on the priority of cells contained in the fan-out. The highest fan-out size port gets the first priority and next fan-out size has the second priority and so on. By keeping this order, the throughput will be increased. This algorithm works in the following phases.

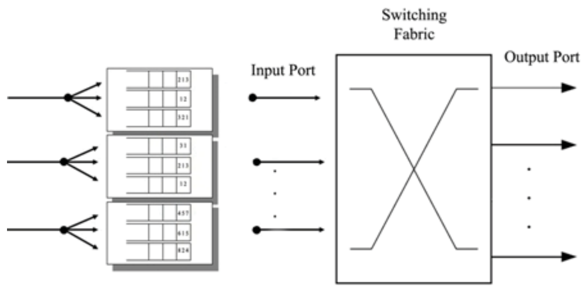

Fig. 1 NxN Input Queued Switch Architecture

Request: The input sends request to all the destined output ports corresponding to the first nonempty queue. At request phase, fan-out size of the current non-empty queue is measured in each input port and prioritize the input ports based on fan-out size. Next step is to assign the due dates to the cells within the fan- out. This Due dates are assigned in a priority input port which will assign the first Due Date (Due Date = 1) to the cells. On the second priority port, elements already presented in first priority are assigned to second Due Date (Due Date = 2) and remaining cells are assigned to the first Due Date (Due Date = 1) and so on. On the completion of these Due Dates, the requests will be made to output ports.

Grant: In the Grant phase, if more than one request is made for the same port, the global pointer pointing one is granted and the others are rejected then the global pointer is incremented to the next position.

When considering alternative multicast switch schedules, we can evaluate how they affect cost structure and whether they make operation easier to manage. With this aspect, MDDR got the following characteristics.

- Supports multicast and multi queues at input ports.
- Utilization of output ports in well since the throughput is increased.
- MDDR is easy to implement in the hardware.
- MDDR will manage maximum load offers.
- Have the ability to flex up to meet queue demand when multicasting.
- Execution of Due Date assignment is fast.

C. Optimal Queue Selection Based Multicast Scheduling Algorithm (OQSMS)

OQSMS [15] is an iterative based algorithm which works in two stages: Queue Selection and Reservation Set. RV is the pre-processing task which stores the queue combination. Based on the queue combination optimal queue is selected. OQSMS achieves the maximum possible throughput and it outperforms the existing MDDR and MDRR scheduling algorithms. In current art of work, queues in each input port are selected by round robin method. In OQSMS an optimal queue is selected in each time slot and fan-out splitting is also made after the queue selection. By using this algorithm maximum possible throughput is achieved without HOL blocking. In this section we detail the OQSMS algorithm design and its components.

Temporary Reservation Set at Each Time Slot

It is an iterative approach, in which each time the RV_T is estimated. It declares the queue selection and achievable throughput temporarily.

Queue Selection at Each Input Port

For each input port an optimal queue is selected such a way that the overall queue selection should result maximum throughput.

Fan-Out Split in Reservation Set

When a queue is selected temporarily or permanently, the fan-out sets of the cells in the queue should be splitted in order to avoid HOL.

Searching Maximum Throughput Reservation Set as Final Reservation Set (Optimized Set)

At each temporary reservation set, the temporary throughput will be estimated and compared with the previous estimation. This iteration ends when a maximum throughput range is possible.

Grant Input Ports with Selected Queue and Fan-Out Split

The end of temporary reservation set is the final reservation set to be granted in order to send traffic.

Reservation Set

The reservation set RV is the array set of queue names selected to each input port. For all the input ports we can have multiple permutations with various queue. But we have to select the optimized queues in each input port. Here reservation set is the pre-process task to store queue combination and its expected throughput range (R_n). Here throughput range R_n refers the number of traffic cells granted at particular timeslot. Here two types of reservation sets are maintained one is Temporary Reservation Set (RV_T) and Final Reservation Set (RV). From the fan-out list of the RV_T, we can estimate the Throughput Range (R_n).

Queue Selection

In order to achieve the maximum throughput, we have to select a queue in each input ports such that the final queue set loads to a maximum throughput. There are so many combinations can be obtained by permutation logic. Therefore, RV_T and R_n estimation will be done iterating for all the combinations until getting a maximum R_n. Maximum R_n will be equal to number of output ports. Thus in each time slot we can select an optimal queue in order to achieve maximum R_n. This increases the maximum throughput in the scheduling. RV_T will be updated or changed during each iteration and finally maximum R_n is reached. RV_T will be assigned to RV. For each time slot RV will be calculated according to the final RV input port cells are granted to send traffic.

D. Scheduling Overhead Analysis

The existing iterative scheduling algorithms have main problem that the scheduling overhead scales up very quickly as the link speed and switch size increase, which limits the scalability in high-speed switches which have very short time to perform scheduling. Scheduling overhead is defined as the information exchanged at an input port in one matching cycle. In existing three-phase (request-grant-accept) integrated scheduling algorithms with log(*N*) iteration times. MDDR and MDRR perform with only one matching cycle, moreover, it has one less operational step (request-grant), and less information exchanged between inputs and outputs. MDDR and MDRR reduce the multicast scheduling overhead from O(k*N*) to O(*N*) by selecting one of the k multicast HoL cells for requesting, where k ($1<k<<2^N-1$) represents the number of multicast queues maintained at each input. But comparing with MDRR, MDDR minimizes the dual pointer overhead. In iterative based scheduling algorithm OQSMS there is no (request-grant-accept) steps. Reservation set (RV) is the pre-processing task to find the throughput range R_n. Based on the throughput range calculation the cells are transferred. OQSMS achieves the maximum possible throughput at each time slot. But the pre-processing task is the overhead at each time before the cells are transferred to output ports.

TABLE I
COMPARISON OF MDRR, MDDR AND OQSMS

Algorithm	Scheduling overhead	Pointer Usage	Queue Selection
MDRR	Request-grant	Dual pointer usage in input side	Based on primary and secondary pointer
MDDR	Request-grant	Global pointer in output side	High fan-out size
OQSMS	No information exchanged between inputs and outputs	No pointer usage in both sides	Optimal queue is selected based on R_n calculation

IV. PERFORMANCE EVALUATION AND RESULT ANALYSIS

To study the performance of OSQMS, MDDR and MDRR the simulation has been conducted in NS2 that models the input queued crossbar switch of size N×N. In general, most of the experiments, we used an 8 × 8 and 16 × 16 VOQ switch. In some experiments the value of N is specific to the experiments. The VOQ's are supplied with Bernoulli uncorrelated and Bursty correlated multicast traffic.

In a Bernoulli process the probability of an arrival in any time slot is p, independent of any other time slot. We consider both uniform and non-uniform traffic scenario. In uniform traffic scenario all input (and output) ports are equally loaded and the fan-out set of new cell is generated randomly, according to a uniform distribution

$$(N_c=2^N-1,\ P_f=\binom{N}{f}/N_c). \qquad (1)$$

A non-uniform traffic scenario, where the fan-out set is chosen according to a non-uniform binomial distribution, with mean fan-out size e_f. That is the probability P_f of choosing a fan-out set of size f is

$$P_f = \binom{N}{f} (e_f / N)^f (1 - e_f / N)^{N-f} \qquad (2)$$

Bursty traffic is a sequence of packets from the same source, travelling to the same destination (with replies in the opposite direction). These two traffic as two different arrival processes. This performance analysis concentrates on two Performance metrics which are Average Cell Delay and Throughput. The graphs drawn in Figs. 2-11 show that the overall performance of Delay and Throughput comparison of MDRR, MDDR and OQSMS algorithms.

Delay: 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. Delay increases when they become unstable as the offered load increases.

Throughput: Throughput is the another performance measurement used in this investigation which is defined as the ratio between the total number of cells forwarded to output interfaces, and the total number of cells arrived at input interfaces. It is essentially a measure of the cell loss probability at input queues.

Fig. 2 shows the average cell delay against the offered load for MDRR, MDDR and OQSMS under Bernoulli uniform traffic. We can observe that as output load increases, OQSMS is very effective in reducing and constantly maintaining the average delay, and performs reasonably well. It is shown that the average delay of OQSMS is always smaller than that of MDRR and MDDR, especially under heavy load. The reason is that MDRR performs an inefficient matching where some of the grants can be wasted because of dual pointer usage in the input side.

Fig. 3 shows the simulation result that the delay occurred according to the load offered by Bernoulli non-uniform traffic. MDDR and OQSMS have no higher level significance of difference in delay. MDRR could not achieve the minimum delay comparing with MDDR and OQSMS because the VOQ' s in each input port is selected based on round-robin pointer. And no priority is provided to the input ports to control the non-uniform distribution.

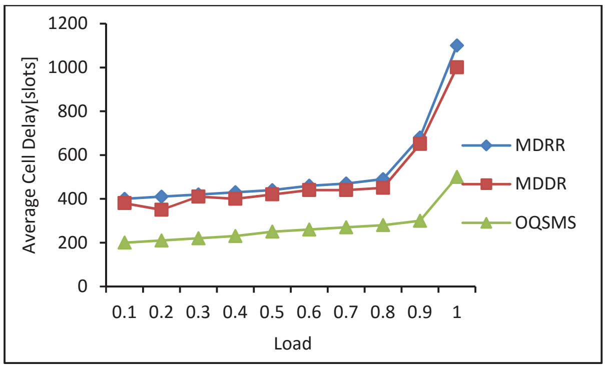

Fig. 2 Average Cell Delay as a function with respect to Load for Bernoulli uniform traffic

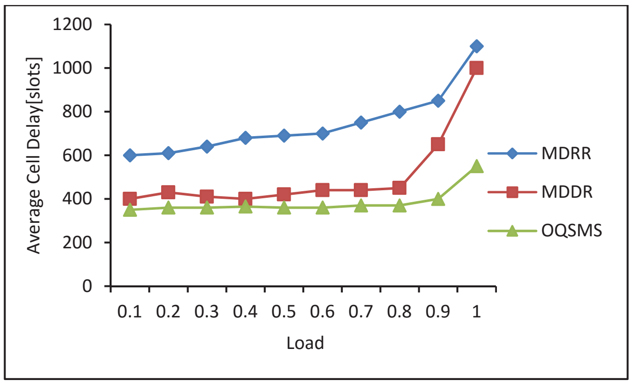

Fig. 3 Average Cell Delay as a function with respect to Load for Bernoulli non-uniform traffic

Fig. 4 illustrates the improvement of maximum achievable throughput performance introduced by increasing the number of queues under Bernoulli uniform traffic. It shows that OQSMS achieves nearly above 95% throughput than MDRR and MDDR for the number of queues above 5. We can observe that when number of queues are increased OQSMS obtains a high switching performance while pointer based algorithms obtain low throughput when number of queues are high. We can conclude that small number of queues are enough to maintain the constant level throughput for pointer based algorithms.

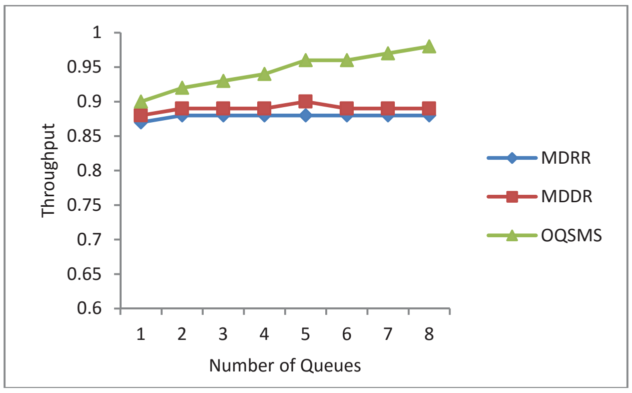

Fig. 4 Number of Queues Vs Throughput under Bernoulli uniform traffic

Fig. 5 shows the throughput analysis of OQSMS, MDDR and MDRR under uniform Bernoulli traffic pattern with respect to load. Above 95% throughput has been achieved by OQSMS. It's shown that when the load is above 0.6 the throughput of MDDR and MDRR are decreased.

Fig. 6 illustrates the throughput as a function of multicast fraction and compare the maximum achievable throughput of OQSMS with MDDR and MDRR under uniform burst traffic pattern. It's shown that irrespective of the arrival traffic pattern OQSMS always achieves higher throughput performance than MDDR and MDRR since when the load increases OQSMS has more queue combinations so that it achieves more throughput.

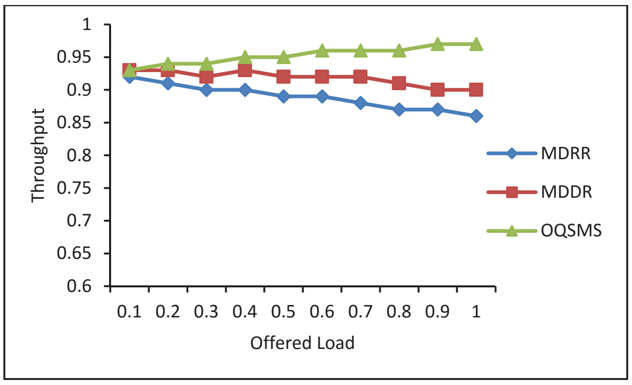

Fig. 5 Throughput as function with offered load for uniform Bernoulli traffic

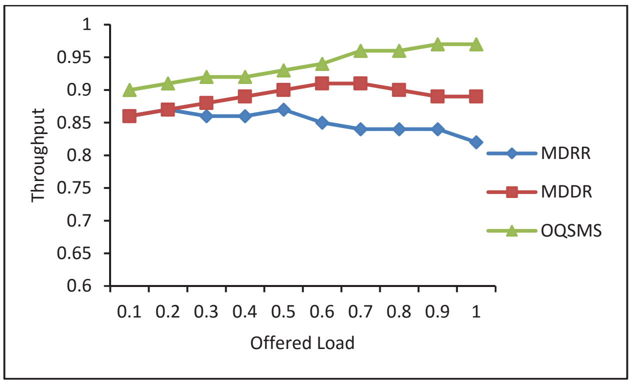

Fig. 6 Throughput as function with offered load for uniform Bursty traffic

Fanout size points to the number of outputs that are destined. This fan-out size is also a performance attribute at output port load. Fig. 7 shows the maximum achievable throughput as a function of mean fan-out size under Bernoulli non-uniform traffic for a 20x20 switch. We can observe the throughput improvement of OQSMS when fan-out size is above 8. Since more transmission possibilities could be provided to the output ports and also when fan-out increases iteration steps to calculate throughput range in OQSMS are effectively reduced which yields good switching performance.

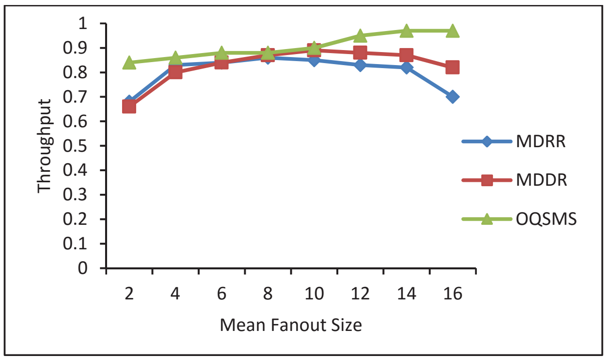

Fig. 7 Throughput as function of mean fanout size for non-uniform traffic

It is shown that the MDRR and MDDR obtain minimum throughput when fan-out increase. In MDDR when fan-out increases, most of the due date assignment value 1 is postponed to the next time slot which is slightly reduces the throughput performance.

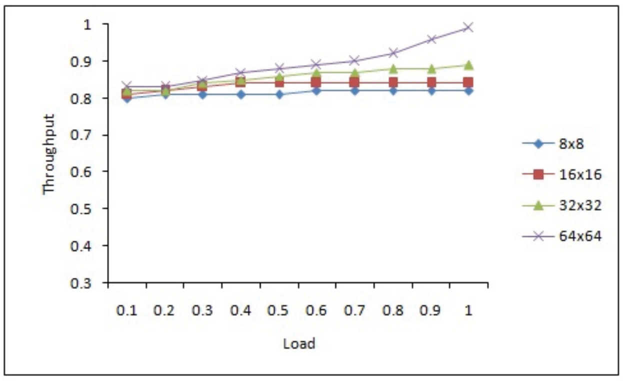

Fig. 8 Throughput performance of OQSMS for different switch sizes under Bernoulli uniform traffic

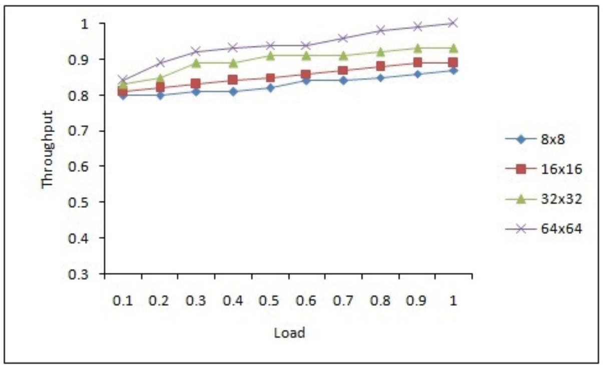

Fig. 9 Throughput performance of OQSMS for different switch sizes under Bernoulli non-uniform traffic

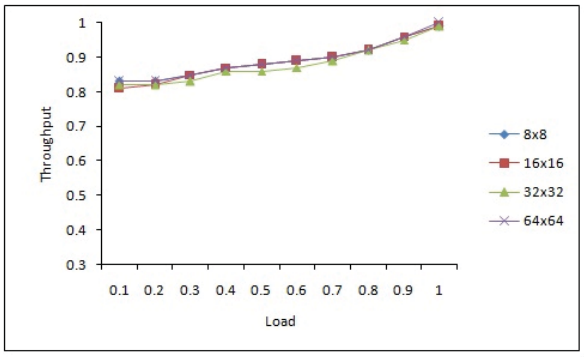

Fig. 10 Throughput performance of OQSMS for different switch sizes under non-uniform bursty traffic

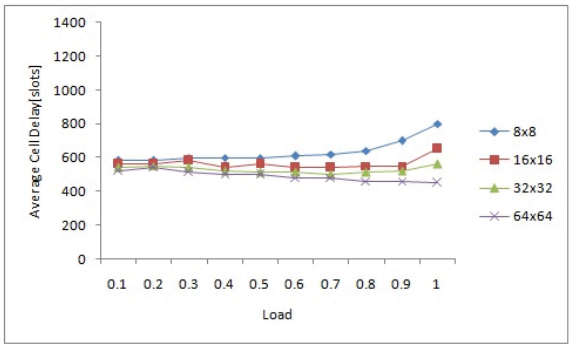

Fig. 11 Average delay of OQSMS for different switch sizes under Bernoulli non-uniform traffic

Finally, the investigation is how a switch size impact the performance of our scheduler OQSMS and how it performs for Bernoulli uniform, Bernoulli non uniform and bursty traffic. Figs. 8-10 show the throughput of the OQSMS for the switch sizes 8x8, 16x16, 32x32 and 64x64. From the simulation results we conclude that when the switch size is 64x64 OQSMS will have more queue combination this will lead to achieve the 100% throughput for both uniform and non-uniform traffic patterns. Fig. 10 shows the when the traffic is bursty there is no slight difference in throughput when the switch is increased.

Fig. 11 shows that the scheduler OQSMS achieves minimum delay with increasing switch sizes under non-uniform traffic. In a 64x64 switch, when the input load reaches 0.5, there is a significant decrease in delay. This is due to multicast cell's large fan-out. When the fan-out is large OQSMS iteration steps is reduced this leads to the minimum delay.

Our future work includes, implementing the MDDR and OQSMS algorithms with feedback based two-stage switch architecture and analyse the performances also extend the OQSMS algorithm to the integrated scheduler which supports both unicast and multicast scheduling simultaneously.

V. Conclusion

In this paper we have implemented scheduling algorithms OQSMS, MDDR and MDRR. From the simulation results that OQSMS achieves more that 95% of the throughput under both uniform and non-uniform traffic pattern. OQSMS estimates optimal queue selection based on more queue combinations so that it achieves minimum delay and maximum in throughput. MDDR performance is better than MDRR since MDDR queue selection is based on the highest fan-out size. MDDR primarily minimizes the request overhead at the output ports and eliminates the dual pointer utilization in input ports. The single pointer algorithms achieve better performance than dual pointer algorithms.

References

[1] 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," *In Proceedings of Workshop on High Performance Switching and Routing*, pp. 111–116, 2003.

[2] Bianco A, Scicchitano A., "Multicast support in multi-chip centralized schedulers in input queued switches," *Computer Networks*, vol. 53, no. 7, pp. 1040–1049, 2009

[3] Gupta S, and Aziz A., "Multicast scheduling for switches with multiple input-queues," *In Proceedings of High Performance Interconnects Symposium*, pp. 28–33, 2002.

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

[5] McKeown N, and Prabhakar B., "Scheduling multicast cells in an input queued switch," *In Proceedings of IEEE INFOCOM*, vol. 1, pp. 271–278, 1996.

[6] McKeown N., "A Fast Switched Backplane for a Gigabit Switched Router," *Business Communication Review*, vol. 27, no. 12, 1997.

[7] McKeown N, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[8] Pan D. and Yang Y., "FIFO-based multicast scheduling algorithm forvirtual output queued packet switches," *IEEE Transactions on Computers*, vol. 54, no. 10, pp. 1283–1297, 2005.

[9] Prabhakar B, McKeown N, and Ahuja R., "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 855–866, 1997.

[10] Shanmugam Arumugam, Shanthi Govindaswamy., "Performance of the Modified Round Robin Scheduling Algorithm for Input-Queued Switches Under Self-Similar Traffic," *In Proceedings of the International Arab Journal of Information Technology*, vol.3, no.2, 1996.

[11] Song M, and Zhu W., "Throughput analysis for multicast switches with multiple input queues," *IEEE Communications Letters*, vol. 8, no. 7, pp. 479–481, 2004.

[12] Yongbo Jiang, Zhiliang Qiu, Ya Gao, and Jun Li, "Multicast Support in Input Queued Switches with Low Matching Overhead", *IEEE Communications Letters*, vol. 16, no. 12, 2012.

[13] Zhu W, and Song M., "Integration of unicast and multicast scheduling in input-queued packet switches," *Computer Networks*, vol. 50, pp. 667–687, 2006.

[14] Zhu W, and Song M., "Performance analysis of large multicast packet switches with multiple input queues and gathered traffic," *Computer Communications*, vol. 33, no. 7, pp. 803–815, 2010.

[15] Navaz K, and Kannan Balasubramanian., "OQSMS: Optimal Queue Selection Based Multicast Scheduling Algorithm for Input-Queued Switches," Australian Journal of Basic and Applied Sciences, 9(27) August 2015, Pages: 373-378

[16] Navaz K, Dr. Kannan Balasubramanian., "Multicast Due Date Round-Robin Scheduling Algorithm for Input-Queued Switches" International Journal of Computer Network and Information Security, 2016, 2, 56-63

Navaz K received the B.Tech degree in Information Technology in 2006, and the M.E degree in Computer Science and Engineering in 2009, all from Anna University, Tamilnadu, India. He is working towards his Ph.D in the department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli, Tamilnadu, India. His areas of interest in research are Computer Networks, Network Design and Simulations, Switch Architecture and Scheduling Algorithms for High Performance Switches.

Dr. Kannan Balasubramanian received the Ph.D degree in Computer Science from UCLA, and the M.Tech degree in Computer Sceince and Engineering from IIT Bombay, India and his Msc (Tech) degree in Computer Science from BITS., Pilani, India. He is a Professor Mepco Schlenk Engineering College, Sivakasi, India. His research interest includes Network Architecture, Protocols, Security and Performance