A novel round-robin based multicast scheduling algorithm for 100 Gigabit Ethernet switches

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

Published in:
proceedings INFOCOM

Link to article, DOI:
10.1109/INFCOMW.2010.5466651

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

A Novel Round-Robin Based Multicast Scheduling Algorithm for 100 Gigabit Ethernet Switches

Hao Yu
Supervisors: Sarah Ruepp, Michael S. Berger
Technical University of Denmark
2800 Kgs. Lyngby, Denmark
haoyu@fotonik.dtu.dk

***Abstract*—This paper proposes a round-robin based multicast scheduling algorithm for high-speed input-queued switches. Fan-out information of each head-of-line cell is examined by the packet scheduler to form a matrix called *Traffic Matrix*. A subscheduler for each column executes the round-robin scheduling algorithm and scheduling decisions are collected into the *Decision Matrix*. To avoid unnecessary multiple transmissions of a multicast cell, the *sync* mechanism is introduced after the *Decision Matrix* is formed. By simulation, the results demonstrate that the number of transmissions is effectively decreased by *sync* while maintaining the same output utilization.**

Keywords: multicast scheduling; round-robin; high-speed switch

I. INTRODUCTION

Multicast capability of a switch is of crucial importance to the communication networks in the foreseeable future. As broadband services such as video conferencing, online gaming and IPTV becoming popular, it is no longer a resource-efficient way to use unicast for these services. Multicast, on the other hand, is able to reduce the network traffic load and multicast latency.

Intensive research has been carried out in the area of scheduling for unicast multistage switches. However, for multicast switch scheduling, research in this area is still at the primary stage. Based on the position of the buffer which is used to store blocked packets, switches are mainly categorized as input-queued (IQ), output-queued (OQ), and virtual output-queued (VOQ). OQ is least favored because of the poor scalability that output memories are required to run with a speedup of N, where N denotes the number of input ports. On the contrary, IQ is advantageous because no speedup is required for the memory access. However, IQ suffers from head-of-line (HOL) blocking problem that reduces the throughput to 58.6% [1]. By creating a queue for each output within each input queue, VOQ is able to solve the HOL problem. Using VOQ for multicast traffic, however, requires 2^N-1 queues in each input for all the possible combinations of destinations, which greatly reduces the scalability. Pan et al [2], [3] propose a way to reduce the number of required queues to N by introducing the concept of address cells and data cells. However, the risk of multiple transmissions of a multicast packet is not eliminated. In this paper, we propose a multicast scheduling algorithm for IQ $M\times N$ switches based on the round-robin discipline.

II. ROUND-ROBIN BASED MULTICAST SCHEDULING AGLORITHM

Normally, a packet is fragmented into fixed-size cells at the input before traversing the switch fabric and then defragmented at the output. The proposed multicast scheduling algorithm is applied after the fragmentation, and is based on the assumption that multicast cells in an input queue are stored in a first-in-first-out (FIFO) fashion. A multicast cell carries a fan-out vector which records the output ports the cell should be sent to. We assume that the fan-out vector has N bits as $b_{N-1}b_{N-2}...b_1b_0$, each of which represents an output port. With a bit set to 1, the multicast cell should be sent to the corresponding output port. A broadcast cell will then have a fan-out vector of all 1s. Before a time slot of cell transmission begins, the packet scheduler examines the fan-out information of the HOL cell in each input in parallel. Information is then used to form a matrix, called *Traffic Matrix* (*TM*) with M rows and N columns, and totally $M{\cdot}N$ elements, $E_{i,j}$ ($0 \le i \le N-1$, $0 \le j \le M-1$). Take a 4×4 switch for instance, if the fan-out vectors of Input 0 to 3 are *1110*, *1100*, *0101*, and *0011* respectively, then the *TM* will be constructed as shown in Fig. 1.

Each column has a round-robin subscheduler to make independent scheduling decisions. After the decisions are made, a *Decision Matrix* (*DM*) is then formed, containing the scheduling results as shown in Fig. 1. Based on this matrix, the scheduler reads each row and sends the corresponding cell from each input. If the row equals to the fan-out of the cell, the cell will be removed from the HOL position and sent to its destination ports. If the row is a subset of the fan-out vector, the cell will only be sent to the granted destinations and remains in its position with an updated fan-out vector. An all-zero row indicates no cell should be released from the input.

Since the subschedulers are independent and execute algorithms in parallel, unnecessary multiple transmissions of a multicast cell may occur. To eliminate this, the scheduler rearranges the *DM* before releasing any cell. This process is defined as *sync*. The role of *arbiter* is passed to each subscheduler in a round-robin fashion. Once a subscheduler becomes the arbiter, the scheduler will use the row that the arbiter's decision points to as a *base* and compare others'

This work was supported in part by the Danish National Advanced Technology Foundation in the project the Road to 100 Gigabit Ethernet

978-1-4244-6739-6/10/$26.00 ©2010 IEEE

decision with it. If the scheduling decision of a nonarbiter is contained in the base, the decision will be ignored and integrated to the base. As shown in Fig. 2, Column 3 is assumed to be the arbiter and its decision points to Row 0 in the *TM*. The scheduler then compares decisions of each column with the Row 0. Since the results of Column 1 and 2 are included in the base, they are then set to 0 and integrated into the base. Thus, an updated *DM*, *DM**, is formed after the *sync* process. The scheduler will then release the cell with fan-out of *1110* from Input 0 and a copy of the cell in Input 2 with fan-out of *0001* to maximize the output port utilization. These two cells are transmitted to the output through the switch fabric simultaneously. After this time slot of cell transmission, the scheduler will update the *TM* as described previously. Since there is no iteration in the proposed scheduling algorithm, a complexity of *O(1)* is achieved, which is suitable for high-speed switches as 100 Gigabit Ethernet switches.

III. Simulation Results Analysis

In this section, we show some simulation results derived from OPNET Modeler [4]. An 8×8 switch is used to evaluate the performance of the proposed algorithm. Bursty traffic is provided to each input with a mean burst size of 16 cells [5]. Fan-out vectors are uniformly distributed between 2 to 8 with an average fan-out cardinality of 4, since we do not consider the mixture of unicast and multicast traffic in this study.

Fig. 2 shows the average number of transmissions per cell as a function of traffic arrival rate. Using *sync* mechanism, the average number of transmissions per cell is significantly reduced and the total number of cells sent to the switch fabric is correspondingly lowered as shown in Fig. 3.

Fig. 4 demonstrates the output port utilization as a function of arrival rate. Nearly the same output port utilization is achieved by both schemes, which shows that the *sync* does not degrade the utilization to reduce the number of transmissions.

IV. Conclusion

In this paper, we propose a multicast scheduling algorithm for high-speed input-queued switches. Fan-out information of HOL cells is examined by the scheduler and is used to form the *Schedule Matrix*. Each subscheduler independently executes the round-robin algorithm on each column and scheduling decisions result in the *Decision Matrix*. A *sync* mechanism is then carried out to reduce the number of transmissions of each multicast cell. Simulation results show that the algorithm effectively performs as designed.

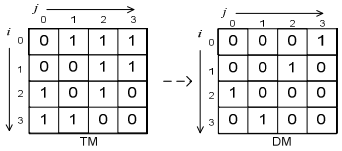

Figure 5. An example of a 4×4 Traffic Matrix and the Decision Matrix

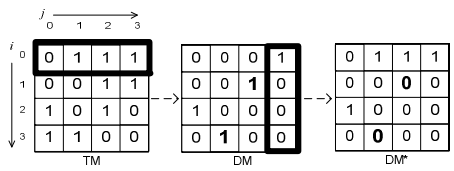

Figure 1. Demonstration of *sync* process

References

[1] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan, "Input versus output queueing on a space-division packet switch", IEEE Transaction on Communications, vol. com-35, no. 12, December 1987

[2] Deng Pan, and Yuanyuan Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches", IEEE Transactions on Computers, vol. 54, no. 10, October 2005

[3] Deng Pan, and Yuanyuan Yang, "Bandwidth guaranteed multicast scheduling for virtual output queued packet switches", Journal of Parallel and Distributed Computing, vol. 69, issue. 12, pp 939-949, August 2009

[4] OPNET Modeler, Available at: http://www.opnet.com/solutions/networr_rd/modeler.html

[5] Balaji Prabhakar, Nick McKeown, and Ritesh Ahuja, "Multicast Scheduling for Input-Queued Switches", IEEE Journal on Selected Area In Communciations, vol. 15, no. 5, June 1997

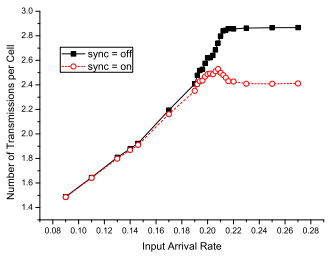

Figure 2. Average number of transmmisions per cell as a function of traffic arrival rate

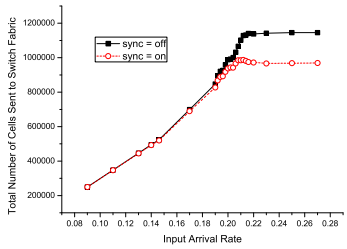

Figure 3. Total number of cells sent to switch fabric as a function of traffic arrival rate

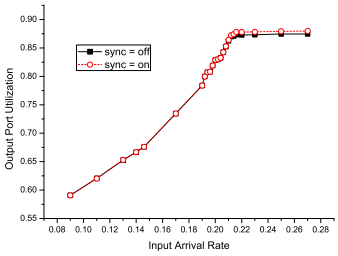

Figure 4. Output port utilization as a function of traffic arrival rate