This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Achieving 100% Throughput for Multicast Traffic in Input-queued Switches

Bing Hu[1], Chunzhi He[2], and Kwan L. Yeung[2]

[1]Depart. of Information Science and Electronic Engineering
Zhejiang University
Hangzhou, PRC
Email: binghu@zju.edu.cn

[2]Depart. of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC
Email: {czhe, kyeung}@eee.hku.hk

***Abstract*—A general approach of designing input-queued multicast switch is to employ multicast switch fabric, where packets can be replicated inside the switch fabric. As compared with unicast switch fabric, the achievable traffic rate region of a switch can be increased, but it is still less than the admissible traffic rate region. In other words, achieving 100% throughput for any admissible multicast traffic pattern is not possible. In this paper, we first revisit the fundamental problems faced by input-queued switch in supporting multicast traffic. We then argue that multicast switch fabric is not necessary if a load-balanced approach is followed. Accordingly, an existing load-balanced two-stage switch architecture [12], consisting of unicast switch fabrics, can be adopted to provide 100% throughput for any admissible multicast traffic pattern. Since the two-stage switch requires no speedup in both switch fabric and packet buffers, we consider it a two-stage input-queued switch. It can be seen that its implementation complexity is much lower than conventional (single-stage) input-queued multicast switches. As compared with the work in [12], our approach is more systematic and we propose a more effective load balancing mechanism.**

Keywords- Input-queued switches; 100% throughput; multicast traffic

I. INTRODUCTION

An input-queued switch only allows one packet to be sent/received by each input/output in each time slot. This makes input-queued architecture more suitable for high-speed switches/routers. An $N\times N$ unicast switch needs to support N^2 unicast flows, where flow(i,j) arrives at input i and goes to output j. An input/output port is overloaded if the input/output load is larger than 1 packet per slot. Let $\lambda_{i,j}$ denote the packet arrival rate of flow(i,j). A traffic pattern is admissible if there are no overloaded input/output ports, i.e. $\sum_j \lambda_{i,j} \le 1$ for all input i's and $\sum_i \lambda_{i,j} \le 1$ for all output j's. The set of all admissible traffic patterns forms the admissible traffic rate region. In designing an input-queued switch, the key objective is to achieve 100% throughput for any admissible traffic [1, 2]. In other words, the achievable traffic rate region of an input-queued switch is desired to be same as admissible rate region. Notably, the maximum weight matching (MWM [3]) algorithm guarantees 100% throughput for any admissible unicast traffic.

An increasing proportion of traffic on the Internet is multicast. The need for supporting multicast becomes as important as unicast. Supporting multicast traffic is inherently more challenging [5]. A multicast switch should also guarantee 100% throughput for any admissible *multicast* traffic pattern. But it is important knowing that multicast traffic before and after packet replication is different. Packet replication is a process of replicating a multicast packet for sending copies to different output ports, according to its multicast address, or fanout. When a multicast packet arrives, it contributes as a single packet to input load. After replication, the loading imposed to outputs is expanded/multiplied by its fanout size. With the above in mind, an admissible multicast pattern can be more precisely defined as a pattern that no inputs are overloaded by packets before replication, and no outputs are overloaded by packets after replication. Packet replication can happen any time between a packet arriving at an input and (a copy of it) leaving an output. Nevertheless, there are two major approaches, replication at input ports or inside the switch fabric. The latter calls for a more expensive switch fabric that is capable of packet replication, or multicast switch fabric.

In this paper, we follow a load-balanced approach to design efficient input-queued switch for supporting multicast traffic. We first challenge the necessity for multicast switch fabric. Based on the concept of load-balancing, we propose to use an existing two-stage switch architecture [12], which consists of unicast switch fabrics and requires no speedup. A new mechanism for load balancing is introduced and we show that it can provide 100% throughput for any admissible multicast traffic pattern. Since the two-stage switch requires no speedup in both switch fabric and packet buffers, we consider it a two-stage input-queued switch. It can be easily seen that its implementation complexity is much lower than conventional (single-stage) input-queued multicast switches. As compared with the work in [12], our approach is more systematic and the new load balancing mechanism we proposed is more effective.

In the next section, we review the related work on multicast switches. Our load-balanced two-stage switch is detailed in Section III, and the stability proof of 100% throughput is given in Section IV. In Section V, the delay-throughput performance of our design is compared with other approaches [12] by simulations. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Supporting multicast traffic is fundamentally more difficult than unicast. The number of multicast flows is increased from

This work was supported in part by Zhejiang Provincial Natural Science Foundation of China (No. Y1100388), Zhejiang Provincial Public Technology Research of China (No. 2010C31071), Fundamental Research Funds for the Central Universities (No. 2010QNA5032), National Science and Technology Major Project (No. 2011ZX03003-003-03).

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

N^2 of unicast to $N(2^N-1)$. The key objective in designing a multicast switch, same as that for unicast, is to guarantee 100% throughput for any admissible multicast traffic pattern. A multicast packet should be replicated before reaching its outputs. Existing multicast switches either replicate packets at input ports [12-13] or inside the switch fabric [5-11].

A. Packet Replication at Input Ports [12-13]

If packet replication takes place at input ports, a multicast packet is cloned to become a set of unicast packets upon its arrival, and each copy is stored in the corresponding VOQ (virtual output queue) according to its fanout. Then, N VOQs per input port are sufficient to carry all unicast copies after replication. The rest of the switch operation is the same as a unicast switch. But one should not overlook the complexity of packet replication. In general, it requires a memory speedup of up to N times, e.g. cloning a single broadcast packet to N unicast VOQs in a single time slot. Another issue is that after replication, the amount of unicast packets leaving an input port is much larger than the amount of multicast packets arrived. Even an input port is not overloaded by multicast packet arrival, it can be easily overloaded by unicast packet departure. Since an input-queued switch can transmit at most one packet from each input port in each time slot, the excess amount of unicast packets will be dropped even before they can enter the switch fabric. This alone makes 100% throughput impossible.

B. Packet Replication inside Switch Fabric [5-11]

With multicast switch fabric, packet replication can take place inside the switch fabric. A multicast switch fabric can be implemented using 2×2 switching elements [4], each with two additional states to support multicast. As compared to packet duplication at inputs, it requires no speedup for packet replication, and lowers the chance of an input to be overloaded by packet departure process. Accordingly, recent research efforts [5-11] have all assumed multicast switch fabric.

In [5], the multicast packet scheduling problem was proved to be NP-hard and two heuristic algorithms were studied in [6]. (Note that unicast scheduling can be optimally solved by MWM algorithm [3], which has a complexity of $O(N^{2.5}\log N)$.) In [7], *fanout splitting* is introduced to "split" a multicast packet for sending copies over multiple time slots. In doing so, packet replication at input ports (via splitting) is (unintentionally) allowed. In fact, a main contribution of [7] is that it proved 100% throughput with multicast switch fabric and fanout splitting is not possible.

More recently, network coding [8] was exploited [9-11] to further expand the achievable rate region of an input-queued switch with fanout splitting under specific traffic patterns. Notably, only *intra-session* network coding was studied because the adopted switch architecture does not facilitate packets from different flows to interact. In [9], the volumes of the achievable and admissible multicast traffic rate regions of a 2×3 switch were obtained by exhaustive search. Let the normalized volume of admissible traffic rate region be 1. The volumes of achievable rate region without fanout splitting, with fanout splitting, and with fanout splitting + network coding were found to be 0.460, 0.937 and 0.952, respectively. We can see that a) 100% throughput is still not possible with fanout splitting + network coding; and b) the gain of using fanout splitting is significant, but the gain of network coding is marginal. It was claimed in [9] that the coding gain will be more pronounced in larger switches, due to more traffic patterns favoring coding. However, there will be a comparable growth (if not more) in traffic patterns that do not favor coding. If network coding is used for traffic patterns that do not benefit from coding, the switch performance (i.e. coding delay and complexity) suffers, and such performance degradation cannot be captured by the volume of rate region. Last but not the least, network coding approach assumes the traffic pattern is known in advance, which is difficult to justify in practice.

III. Load Balanced Multicast Switch (LBMS)

A. Our Approach

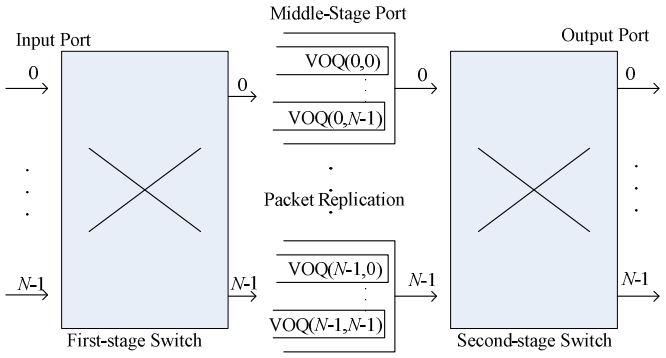

Fig. 1 A load-balanced multicast switch architecture.

It is our belief that adopting multicast switch fabric cannot solve the fundamental problem faced by input-queued switch in supporting multicast traffic, but likely shifts the problem to elsewhere, e.g. making the multicast scheduling algorithm NP-hard and switch fabric more expensive. In this section, we provide a solution based on the load-balancing concept and only using unicast switch fabric. With unicast switch fabric, packet replication is carried outside switch fabric. As we have pointed out in Section II, two problems for packet replication at input ports are a) the replication process requires a potential speedup of N times; and b) the amount of the replicated unicast packets may overload the link connecting input port to switch fabric even though the multicast traffic is admissible.

With the above two problems in mind, we design a two-stage switch architecture, as shown in Fig. 1. The first stage switch fabric is for evenly distributing multicast packets to different middle-stage ports, and the second stage is for delivering packets to their correct outputs. Both stages use unicast switch fabric. Since a multicast packet is switched to its connected middle-stage port as soon as it arrives, no input port buffer or scheduler is required. Packet replication is carried out at middle-stage ports (only). It can be implemented by storing a multicast packet once, and reading it out multiple times, according to the scheduling algorithm used by the second stage switch (e.g. MWM algorithm [3]). Accordingly, the VOQs at each middle-stage port only store the pointers to the memory address where the multicast packet locates. Note that buffer space required for storing pointers is much smaller than storing packets. Although each multicast packet arrival will still trigger multiple updates to the pointer-based VOQs, such an operation will be much more efficient than physically replicating the same packet multiple times.

Indeed, if the multicast traffic pattern is admissible, each input/output port in Fig. 1 will have a loading less than or equal to 1. Obviously, the rate for multicast packets arrived at each middle-stage port will be less than or equal to 1 as well. The key issue is: after packet replication at middle-stage ports, can we ensure the amount of replicated unicast packets at each middle-stage port is still less than or equal to 1? This hinges on the load-balancing mechanism in the first stage switch.

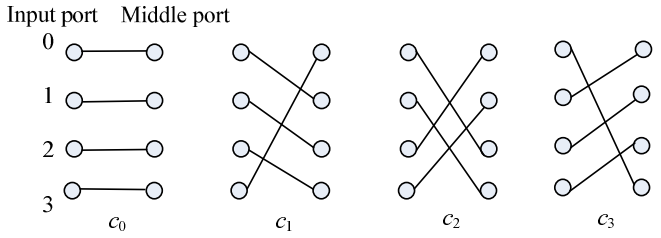

Fig. 2 A set of switch configurations for load-balanced multicast switch.

We propose a very simple load balancing mechanism for the first stage switch. Let $\boldsymbol{C}$ be a set of switch configurations, which is initialized to the pre-determined $\{c_0, c_1 \ldots c_{N-1}\}$. Note that each configuration c_m (m=0, 1…N-1) is represented by a permutation matrix. An example for $\{c_0, c_1 \ldots c_{N-1}\}$ in a 4×4 switch is given in Fig. 2. The only requirement for $\{c_0, c_1 \ldots c_{N-1}\}$ is that each input is connected to each middle-stage port exactly once. At any time slot t, randomly select one from $\boldsymbol{C}$, say c_m, to configure the first-stage switch fabric. Then update $\boldsymbol{C} = \boldsymbol{C} - c_m$. If the running set $\boldsymbol{C}$ is empty, reset $\boldsymbol{C} = \{c_0, c_1, \ldots c_{N-1}\}$. We show in Section IV that for any admissible multicast traffic pattern, this simple load-balancing mechanism is powerful enough to ensure that a) all replicated unicast packets are uniformly distributed to each middle-stage port; and b) the amount of replicated unicast packets entering each middle-stage port is no greater than 1. Then 100% throughput can be comfortably guaranteed by configuring the second-stage switch with a pre-determined and periodic sequence of N configurations. Without loss of generality, we can reuse the same set $\{c_0, c_1 \ldots c_{N-1}\}$ generated for the first stage. The major difference is that each configuration in $\{c_0, c_1 \ldots c_{N-1}\}$ is now used in a round-robin order, instead of random selection from the running set $\boldsymbol{C}$. Specifically, at any time slot t, the second-stage switch fabric is configured by c_m, where $m = t \bmod N$. When a middle-stage port j connects to an output port k based on c_m, the HOL packet from VOQ(j,k) is sent to output k.

In summary, we propose a simple load-balanced multicast switch (LBMS) in this section, where the first stage is configured by randomly selecting a configuration from a running set $\boldsymbol{C}$, and the second stage is configured according to a pre-determined and periodic sequence of configurations. Both two stages only need to realize N configurations, which is much simpler than a conventional unicast switch fabric that requires to realize N! configurations. In LBMS, there is no input port scheduler and the middle-stage port sends the HOL packet to its connected output port based on the current second-stage configuration. The computation complexity overall is $O(1)$ only, which is greatly lower than the NP-hard of multicast switch fabric. Despite its simple design, we prove that the LBMS can guarantee 100% throughput for any admissible multicast traffic pattern in Section IV.

B. Birkhoff-von Neumann Switch [12]

Admittedly, the load-balanced multicast switch (LBMS) we proposed above, even though independently, is similar to the Birkhoff-von Neumann switch in [12]. Birkhoff-von Neumann switch was designed with unicast traffic in mind, but it can be applied to multicast traffic as well. The major/only difference is that the different load-balancing mechanisms are used in the first-stage switch. In [12], two load-balancing mechanisms were studied, random and fixed. The random approach (denoted as BN-random) is that in each time slot, the first-stage switch is configured by randomly selecting a configuration from the *original* set of pre-determined configurations $\{c_0, c_1 \ldots c_{N-1}\}$, where our approach is to select one from the running set $\boldsymbol{C}$. Following the fixed approach (i.e. BN-fixed), the first-stage switch is configured in the same way as the second stage, i.e. using the same/another fixed sequence in $\{c_0, c_1 \ldots c_{N-1}\}$.

It was shown in [12] that BN-random can guarantee 100% throughput for any admissible multicast traffic pattern (same as our LBMS), whereas BN-fixed can only guarantee 100% throughput if the incoming traffic is stationary and weakly mixing. Note that weakly mixing sequence is a proper subset of ergodic sequence and describes how fast a sequence loses its memory. In [12], BN-fixed is preferred due to its better delay performance. In terms of hardware complexity, BN-fixed is simpler than BN-random due to its predetermined and periodic configurations. Our LBMS can also be implemented at a comparable low-cost. In LBMS, its first-stage configuration can be randomly obtained in advance and then confirmed for using at a future time slot. For example, a random order for all elements in $\{c_0, c_1 \ldots c_{N-1}\}$ is generated using N slots from time t+1 to t+N. Then such "predetermined" sequence of configurations can be adopted between time slot t+N+1 to t+2N.

IV. 100% THROUGHPUT PROOF FOR LBMS

In this section, we first show that the load-balancing mechanism in the first stage of LBMS is effective. Specifically, we prove that the sequence of configurations generated from the running set $\boldsymbol{C}$ is weak mixing. Upon such first stage, the replicated unicast packets are uniformly distributed to each middle-stage port with an arrival rate no greater than 1. Afterwards, 100% throughput for the second stage and thus whole switch can be derived. Let $D_1(t)$ and $D_2(t)$ be the N×N permutation matrices at time slot t in our first and second-stage switch fabrics respectively.

Statement 1: $D_1 = \{D_1(t), t \geq 0\}$ is a *stationary* and *weakly mixing* stochastic sequence.

Here, we omit the proof of statement 1 due to its straightforward progress and the page limitation. Clearly, weak mixing implies ergodicity [14], so time averages of D_1 are equal to its ensemble averages:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{s=1}^{t} D_1(s) = \mathbf{E}\, D_1(1) = \frac{1}{N}\,\underline{\mathbf{e}} \qquad (1)$$

where $\underline{\mathbf{e}}$ is the N×N matrix with all its elements being 1. D_2 is a stationary and ergodic stochastic sequence due to its pre-determined and periodic priorities [12], so we also have:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{s=1}^{t} D_2(s) = \mathbf{E}\, D_2(1) = \frac{1}{N}\,\underline{\mathbf{e}} \qquad (2)$$

In the following, we select any one output port k and study how its traffic distributed among the middle-stage ports. At time slot t, if a multicast packet arrives at input port i and output k is in its fanout, we let $a_i(t) = 1$. Otherwise, $a_i(t) = 0$. Then at time t, the arriving traffic at all N input ports with destination output k can be accurately described by an $N\times 1$ vector $\underline{a}(t) = (\, a_i(t)\,)$, $i = 0,1\ldots N\text{-}1$. If a sequence is not ergodic, this implies that its time averages are not equal to its ensemble averages. This kind of non-ergodic traffic is of no concern with the switch stability [15]. Therefore, we make the same reasonable and primary assumption as BN-random in [12]:

Assumption 1: $\{\underline{a}(t),\ t \ge 0\}$ is a stationary and ergodic stochastic sequence.

Then the time averages of $\{\underline{a}(t),\ t \ge 0\}$ are equal to its ensemble averages:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{s=1}^{t} \underline{a}(s) = \mathbf{E}\,\underline{a}(1) = \underline{\lambda} \qquad (3)$$

where $\underline{\lambda} = (\lambda_{i,k})$, $i = 0,1\ldots N\text{-}1$, is an $N\times 1$ vector and $\lambda_{i,k}$ is the mean rate of $a_i(t)$. In fact, $\lambda_{i,k}$ is the traffic rate for unicast flow(i,k) (after duplication) and $\sum_i \lambda_{i,k}$ is the traffic rate to output port k. Based on the condition of admissible traffic,

$$0 \le \lambda_{i,k} \le 1, \qquad 0 \le \sum_i \lambda_{i,k} \le 1 \qquad (4)$$

Statement 2: the unicast traffic obtained after packet replication is uniformly distributed to each middle-stage port with an arrival rate less than or equal to 1.

Proof: All multicast packets are duplicated to become unicast packets upon their arrivals at middle-stage ports. If an middle-stage port j receives a unicast packet destined to output k at time slot t, we let $b_j(t) = 1$. Otherwise, $b_j(t) = 0$. Then at time t, an $N\times 1$ vector $\underline{b}(t) = (\, b_j(t)\,)$, $j = 0,1\ldots N\text{-}1$, can present all unicast packets with the destination output k received by all middle-stage ports. Recall that an input port switches a packet to its connected middle-stage port as soon as the packet arrives:

$$\underline{b}(t) = D_1(t)\,\underline{a}(t) \qquad (5)$$

Both $\{D_1(t),\ t \ge 0\}$ and $\{\underline{a}(t),\ t \ge 0\}$ are stationary and independent to each other, so $\{\underline{b}(t),\ t \ge 0\}$ is stationary from (5). Moreover, since $\{D_1(t),\ t \ge 0\}$ is weak mixing (Statement 1) and $\{\underline{a}(t),\ t \ge 0\}$ is ergodic (Assumption 1), we get that $\{\underline{b}(t),\ t \ge 0\}$ is an ergodic sequence [15]. Combine (5) with (1) and (3),

$$\mathbf{E}\,\underline{b}(t) = \mathbf{E}\, D_1(t)\underline{a}(t) = \mathbf{E}\, D_1(t)\ \mathbf{E}\,\underline{a}(t) = \frac{1}{N}\,\underline{\mathbf{e}}\,\underline{\lambda}$$

We can see that any middle-stage port j receives the unicast packets destined to any output k with the mean rate

$$\mathbf{E}\, b_j(t) = \frac{1}{N}\sum_i \lambda_{i,k} \qquad (6)$$

From (4) and (6), we have:

$$\mathbf{E}\, b_j(t) \le \frac{1}{N} \qquad (7)$$

The meaning of (7) is that the generated unicast packets going to any output k arrive at any middle-stage port j with the mean rate less than or equal to $\frac{1}{N}$. In other words, the unicast traffic obtained after packet replication is uniformly distributed to each middle-stage port. Then the total unicast packets received by any middle-stage port j can be calculated by:

$$N \times \mathbf{E}\, b_j(t) \le N \cdot \frac{1}{N} \le 1$$

The arrival rate for the unicast traffic obtained after packet replication is less than or equal to 1. #

Statement 2 ensures that a) all replicated unicast packets are uniformly distributed to each middle-stage port; and b) the amount of the replicated unicast packets will not overload the link connecting any middle-stage port to the second-stage switch fabric. Then 100% throughput for any admissible multicast traffic pattern is achievable.

Theorem 1: the load-balanced multicast switch guarantees 100% throughput for any admissible multicast traffic pattern.

Proof: We focus on any one middle-stage port VOQ(j,k). Its mean arrival rate is $\mathbf{E}\, b_j(t) \le \frac{1}{N}$ from (7), while its service rate is $\frac{1}{N}$ from (2). Therefore, there is no traffic accumulation at VOQ(j,k) in a long term. The second-stage switch is stable and thus the whole load-balanced multicast switch guarantees 100% throughput for any admissible multicast traffic pattern. #

V. Performance Evaluation

As the key objective in designing an input-queued switch, 100% throughput already makes LBMS clearly superior to the multicast switches that cannot provide 100% throughput, e.g. adopting multicast switch fabric with fanout splitting and network coding [9]. As such, we concentrate on comparing the delay performance between LBMS and other 100% throughput switches BN-random and BN-fixed [12]. Output-queued switch is also implemented as a lower bound. Although we only present simulation results for switch with size 16×16 below, the same conclusions and observations apply for other sizes.

A. Uniform Bursty Traffic

Uniform bursty traffic is generated as follows. If i is an even number, no packet arrives to input port i for all time slots. Otherwise, packet arrival is modeled by the ON/OFF traffic model. In the OFF state, no packet arrives. In the ON state, a packet arrival is generated in every time slot, which has equal probability of being unicast or multicast. If the packet is unicast, it destines to each output with equal probability. If the packet is multicast, its fanout size is randomly selected between [2, 16], and the identity of each output in the fanout is randomly selected from all output ports. Given the average input load λ and average burst size q, the state transition probability from OFF to ON is $\lambda/[q(1\text{-}\lambda)]$, and from ON to OFF is $1/q$. Simulation results in Fig. 3 are based on $q = 30$ packets.

The superiority of BN-fixed and LBMS on load-balancing is clearly described by this bursty traffic in Fig. 3. LBMS and BN-fixed beat BN-random even when traffic is light. For example when throughput is 0.6, with BN-random packets experience an average delay of 45.64 time slots, whereas for

LBMS is just 31.71, cutting down the delay by 30.5%. The curves of LBMS and BN-fixed almost overlap with each other. But it should be noted that BN-fixed cannot guarantee 100% throughput for any admissible multicast traffic pattern.

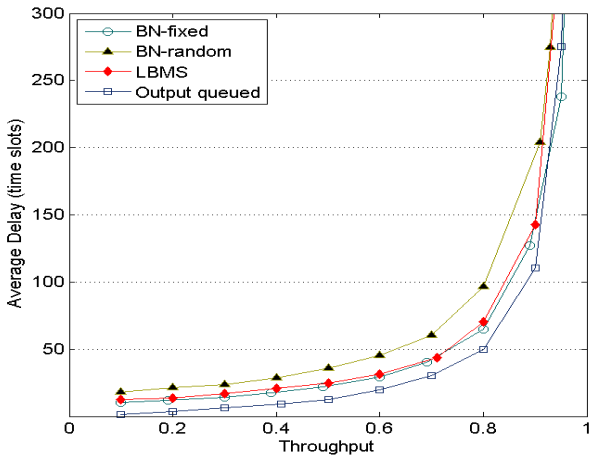

Fig. 3: Delay vs throughput, under uniform bursty traffic

B. *Non-weakly Mixing Traffic*

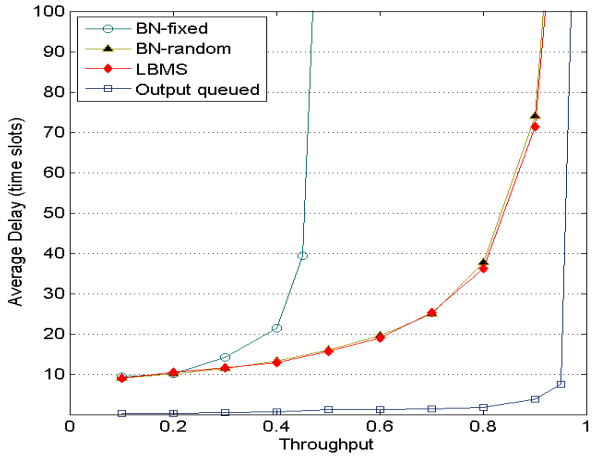

Fig. 4: Delay vs throughput, under non-weakly mixing traffic

If t is an odd number, no packet arrives to all input ports at time slot t. Otherwise, packet arrives at input port i only if i is an even number. The arriving packet is confirmed to be a multicast packet with the fixed fanout size 4. The identity of each output in the fanout is also randomly selected from all output ports. Then this traffic is non-weak mixing but ergodic and admissible. From Fig. 4, BN-fixed only provides 50% throughput, while LBMS and BN-random guarantee 100% throughput.

In summary, for any admissible multicast traffic pattern LBMS and BN-random can guarantee 100% throughput but BN-fixed cannot [12]. Among the only two 100%-throughput multicast switches, LBMS always yields a lower delay than BN-random under various traffic conditions.

VI. CONCLUSIONS

In this paper, we proposed an input-queued multicast switch under the light of load-balancing. It consists of two stages switches. Both stages use unicast switch fabric and only need to realize N switch configurations. Since a packet is switched to a middle-stage port as soon as it arrives, no input port buffer or scheduler is required. The middle-stage port carries out packet replication and sends the HOL packet to its connected output port based on the current second-stage configuration. Then the computation complexity in our switch is $O(1)$ only, which can be implemented at a much lower hardware cost than the NP-hard of multicast switch fabric. We proved that the load-balanced multicast switch (LBMS) guarantees 100% throughput for any admissible multicast traffic pattern. The simulation results also showed that LBMS yields a much lower delay than the 100%-throughput BN-random [12] multicast switch under various traffic conditions. This can be attributed to the more effective load balancing mechanism in LBMS.

REFERENCES

[1] B. Hu and K. L. Yeung, "Feedback-based scheduling for load-balanced two-stage switches," *IEEE/ACM Transactions on Networking,* Vol. 18, Issue. 4, pp. 1077-1090, Aug. 2010.

[2] B. Hu and K. L. Yeung, "Load-balanced optical switch for high-speed router design," *IEEE/OSA Journal of Lightwave Technology,* Vol. 28 , Issue. 13, pp. 1969-1977, July 2010.

[3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input queued switch," *IEEE INFOCOM*, April 1996, San Francisco, USA.

[4] T. Lee, "Nonblocking copy networks for multicast packet switching," *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, pp. 1455–1467, Dec. 1988.

[5] M. Andrews, S. Khanna, and K. Kumaran, "Integrated scheduling of unicast and multicast traffic in an input-queued switch," *IEEE INFOCOM*, March 1999, pp. 1144–1151, New York, NY, USA.

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

[7] 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 Transactions on Networking*, Vol. 11, No. 3, 2003.

[8] S. Li, R. Yeung, and N. Cai, "Linear Network Coding", *IEEE Transactions on Information Theory*, Vol. 49, No. 2, pp. 371-381, 2003.

[9] J. Sundararajan, M. M´edard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter, "Network coding in a multicast switch," *IEEE INFOCOM*, May 2007, pp.1145–1153, Anchorage , Alaska , USA.

[10] M. Kim, J. K. Sundararajan, and M. M´edard, "Network coding for speedup in switches," *IEEE International Symposium on Information Theory (ISIT)*, June 2007, pp. 1086–1090.

[11] M. Kim, J. K. Sundararajan, M. Médard, A. Eryilmaz and R. Kötter, "Network coding in a multicast switch" *IEEE Transaction on Information Theory*, Vol. 57, No. 1, January 2011.

[12] C.S. Chang, D.S. Lee and Y.S. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *Computer Communications*, Vol. 25, pp. 611-622, 2002.

[13] B. Hu and K. L. Yeung, "Multicast scheduling in feedback-based two-stage switch," *IEEE Workshop on High Performance Switching and Routing*, June 2009, Paris, France.

[14] K. Petersen, "Ergodic Theorem," *Cambridge: University press*, 1983.

[15] C. S. Chang, "Performance guarantees in communication network," *Springer*, London, 2000.