Distribute and Match – The DM Switch for High Speed Packet Switching Networks

Zhenghao Zhang
Computer Science Department
Florida State University
Tallahassee, FL 30306, USA
zzhang@cs.fsu.edu

***Abstract*—High speed switch is one of the key components in modern networks. In this paper, we propose the Distribute and Match (DM) switch, a novel switch that achieves 100% throughput while maintaining packet order without requiring speedup of the switching fabric. The DM switch has two switching stages, where the first stage distributes the packets to the middle nodes and the second stage forwards packets either according to a matching algorithm or based on reservation. The DM switch achieves good performance because the matching algorithm is usually capable of forwarding packets with small delay up to medium load, while under high load, the DM switch activates the reservation mode and can sustain high throughput. Our simulation confirm that the DM switch achieves overall better performance than other compared implementable switches.**

I. Introduction

High speed switch is one of the key components in modern networks. A switch has multiple input and output lines, connected by the switching fabric, which forwards the packets from the input side to the output side. Typically, at each input port, packets are buffered and organized as Virtual Output Queues (VOQ), one queue for each output port. To resolve output contention, i.e., multiple inputs sending packets to the same output, a scheduling algorithm can be employed to find a contention-free schedule [7], [8]. For switch with only input buffers, the bottleneck is usually the scheduling algorithm because the running time of the algorithm is constrained by the length of a time slot and optimal algorithms cannot be employed due to their complexities. With practical algorithms, the input-buffered switch may incur long delay when the load is high under certain types of traffic.

Another type of switch is the two-stage switch [1], [3], [4], [5], [6], which has two switching fabrics operating at the line rate, both following a fixed, round-robin schedule to connect the inputs to the outputs. The two-stage switch basically employs no scheduling algorithm, and can achieve 100% throughput because the switch can be configured based on reservation such that no capacity is lost due to contention. However, precisely because of the lack of a scheduling algorithm, the exiting two-stage switches rely heavily on reservation which, although guarantees 100% throughput, suffers unnecessarily long delays when the traffic load is not high.

The DM switch is based on the simple idea of combining the strengths of both the input-buffered switch and the two-stage switch. As shown in Fig. 1, it uses two switching fabrics operating at the line rate and introduces the *middle nodes* between the switching fabrics to buffer packets. Packets are first transfered from the input ports to the middle nodes then to the output ports, either as *contention packets* or *reservation packets*. The first stage is called the *distributing stage* and follows the same fixed connection schedule as the two-stage switch. The second stage is called the *matching stage*, and is so named because under low or medium loads, it usually works in the contention mode, i.e., allowing packets to compete for the access of the output ports according to a matching algorithm. The synergy between the distributing stage and the matching stage leads to a good performance. Comparing to the input-buffered switch, the matching stage accepts randomized input which facilitates the discovery of good schedules. Comparing to the two-stage switch, employing a matching algorithm improves the success rate of contention resolution, such that the DM switch starts the reservation mode at a much higher load hence improving the delay performance under low or medium traffic load. Under high loads, the matching stage switches to the reservation mode, i.e., sends packets according to synchronized schedules, and the DM switch enjoys the performance guarantees based on reservation.

The DM switch is a good option for cases when the length of a time slot is long enough to allow a scheduler to find a matching, such as those currently employing the input-buffered switches. It improves the performance over the input-buffered switches by borrowing the advantages from the two-stage switches. The design of the DM switch faces the following key challenges. First, due to the distribution stage, packets in the same flow may be randomly stored at different middle nodes; however, to maintain packet order, only the oldest packet in a flow should be sent. Therefore, a mechanism should be designed to maintain packet order with both low

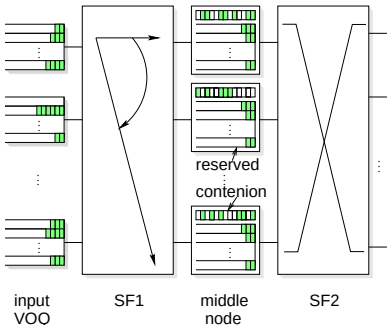

Fig. 1. The DM switch.

time complexity and low hardware complexity. Second, the DM switch supports two modes of packet forwarding, either through contention or through reservation. The interaction of the two modes should be carefully designed to achieve good performance under all loads. In this paper, we address these challenges, and our simulation show that the DM switch achieves desirable performance.

The rest of the paper is organized as follows. Section II describes the DM switch. Section III proves that the DM switch achieves 100% throughput if the matching is maximal. Section IV evaluates the DM switch with simulations. Section V discusses the related works. Section VI concludes the paper.

II. The DM Switch

We describe the design of the DM switch in this section.

A. Preliminaries and Notations

We consider the switches operate in a time-slotted manner, where the length of a time slot is the time to send a packet. The switch size is denoted as N. Input port i, middle node r, and output port j are denoted as I_i, M_r, and O_j respectively, for $0 \leq i, r, j \leq N-1$. Flow $F(i,j)$ refers to the packets arrived at I_i destined to O_j. A packet is a *head-of-flow packet*, if it is in the buffer and is the oldest packet in a flow.

The decision making elements often has to make selections from a set of candidates, represented by a binary vector where '1' means a candidate and '0' otherwise. In the DM switch, all such decisions are made based on *round-robin*. That is, a decision making unit keeps a round-robin pointer, and always selects the candidate with an index closest to the pointer clockwise, wrapping around if necessary. The round-robin pointer, if needs to be updated, is updated to be the index clockwisely next to the last selected candidate, wrapping around if necessary. For example, suppose $N = 6$. If the vector is $[010100]$ while the pointer at 4, the selected candidate will be 1 and the pointer will be updated to 2.

B. Hardware Setup and Packet Forwarding Modes

The DM switch is shown in Fig. 1. Each input port has a buffer organized into N VOQs. The input ports are connected to the middle nodes via the first switching fabric of the DM switch, referred to as SF1. SF1 follows a fixed, round-robin switching pattern: if the current time slot is t, I_i will be connected to M_r where $r = (i+t) \mod N$.

An input port may operate in two modes, the *reservation mode* and the *contention mode*. The reservation mode always starts at the time slot when the input port is connected to M_0. At this time slot, if the input port has at least one VOQ with length no less than N, it selects one according to round-robin and starts the reservation mode and will stay in this mode for N time slots. That is, it continuously sends packets in this VOQ as *reservation packets* for N time slots, one to each middle node due to the connection of SF1. The N packets are called a *frame*, and the packet stored at M_0 is called the *head-of-frame* packet. After a frame is sent, the input port is connected to M_0 again and may start another cycle of reservation mode. A middle nodes receives the reservation packets and stores them in its *reservation buffer*, which is also organized into N queues referred to as *RVQ*s, one queue for each output port. Packets arrived at the reservation buffer are served according to the first-in-first-out rule.

At any time slot, if an input port is not in the reservation mode, it is in the contention mode. A packet sent while the input is in the contention mode is called a *contention packet*. Suppose the current time slot is t and I_i is connected to M_r. If there is at least one non-empty VOQ, I_i will attempt to send a contention packet to M_r. When there are multiple non-empty VOQs, I_i chooses a VOQ based on round-robin. However, unlike the reservation packets that can always be sent, whether a contention packet can be sent is determined by M_r. Note that in addition to the reservation buffer, each middle node holds a separate buffer called the *contention buffer* denoted as *CTB*, that stores only the contention packets. Unlike the reservation buffer that has no constraint on size, a contention buffer can hold exactly W packets. We call location $t \mod W$ the *current location* of the contention buffer. Basically, *I_i can send a packet to M_r at time slot t if there is no packet stored at the current location; and if a packet is sent, it will be stored at the current location.* For this reason, the contention buffer is said to be *timestamp-indexed*.

The middle nodes are connected to the output ports through switching fabric 2, denoted as SF2, which is a crossbar and is used to send both the reservation packets and the contention packets. Like the input ports, an output port can also be in two modes, the reservation mode and the contention mode. O_j starts reservation mode always at time slot $kN - j$, called the *frame checking point*, for some integer k. It checks whether M_0 has a reservation packet to it that is a head-of-flow packet. If so, it will be operating in reservation mode and will continuously receive reservation packets from the middle nodes for N time slots, one from each middle node, in an ascending order according to the indices of the middle nodes. All such packets belong to a same frame planted sequentially in the middle nodes by the input port.

If an output port is not in the reservation mode, it is in the contention mode. When a middle node is not sending a reservation packet, it is in the contention mode. Conceptually, a bipartite graph exists between the middle nodes and the output ports that are in the contention mode. If M_r holds a contention packet destined to O_j that is a head-of-flow

packet, there is an edge exists between them, i.e., a contention packet can be transfered from M_r to O_j. The packet transfer schedule is determined by a matching algorithm calculated in parallel between N middle node schedulers and N output port schedulers. In every time slot, it finds a matching and the middle nodes send packets to the output ports according to the edges in the matching. The algorithm can be any parallel matching algorithm, such as iSLIP [7].

C. The Information Queue and Maintaining the Packet Order

To maintaining packet order, a packet can be sent only if there is no older packets belonging to the same flow stored in the buffers of the switch. In the DM switch, this is achieved by maintaining Information Queues (IFQ) at each output port. Basically, O_j maintains IFQ_{ij} for $0 \leq i < N$, where IFQ_{ij} stores the information about the packets arrived at I_i destined to O_j that are currently buffered in the middle nodes, including reservation buffer and the contention buffer. The information includes: 1) whether it is a reservation or contention packet, 2) the index of the middle node storing this packet, and 3) the time stamp of the packet. This information can be maintained by asking a middle node to send a message to the output port each time slot. As a middle node can receive at most one packet per time slot, the message is about only one packet and can be encoded into only 2 bits, where one bit indicates whether a packet is received or not and the other indicates whether it is a reservation or contention packet. The output port, based on the index of the middle node, can find out the input port where this packet came from, because SF1 follows a fixed connection pattern. Note that to run the scheduling algorithm, each middle node scheduler is already connected by N wires to all the output port schedulers and such wires can be used to carry the messages.

With IFQs, at the frame checking point, the output port knows if the head packet at the reservation buffer is a head-of-flow packet, and will start to receive the frame only if it is. Packets belonging to the same frame are sent to the middle nodes in order and are received by the output port in order.

For the contention packets, an output port scheduler sends a request to the middle node scheduler if and only if there exists an IFQ whose head packet is stored in the contention buffer at this middle node. Therefore, if M_r is matched to O_j, there must be a packet that can be transfered. If the algorithm schedules a transfer from M_r to O_j, O_j first selects a packet according to round-robin, because there could be multiple IFQs whose head packets are stored in M_r. O_j then sends the lower $\log_2 W$ bits of the time stamp of the selected packet to M_r with which M_r can locate the packet in constant time.

D. The Interaction between Reservation and Contention

The DM switch sends packet either in the reservation mode or the contention mode. We use two simple rules for coordinating between the two modes:

- *Global Block:* No input port should send contention packets if there is a frame of reservation packets whose head-of-frame packet is not a head-of-flow packet.
- *Individual Block:* No input port should send contention packets destined to O_j if there is a reservation packet destined to O_j still in the buffer.

We prove in Section III that these two simple rules guarantee achieving 100% throughput in the DM switch.

In practice, each output port checks its IFQs and if an IFQ contains reservation packets but its head packet is not a reservation packet, a Global Block should be activated by generating a bit. The bits from all output ports are "or"ed and can be announced. Each middle node knows the state of its reservation buffer, and can generate an N-bit vector to indicate whether an individual block is needed for an output port, a bit being '1' means needed and '0' otherwise. This vector can be bitwise "or"ed with the vectors from all other middle nodes, and then can be sent to the input ports via the connection between the input ports and the middle nodes.

E. Discussions

The DM switch, compared to a typical two-stage switch, employs a different switching fabric in the second stage capable of dynamic configurations. However, the switching fabric is the same as that in an input-buffered switch and may employ the same scheduling algorithm. In each time slot, the DM switch may require control message passing: 1) a middle nodes needs to send the 2-bit packet information to an output port to maintain the IFQs, 2) an output port needs to send the lower $\log_2 W$ bits of the time stamp to a middle node to select a packet, 3) the output ports need to send 1 bit for Global Block to the input side of the switch, and 4) the middle nodes needs to send N bits for Individual Block to the input side of the switch. While this may increase the complexity of the switch, we believe the cost is far from prohibitive. We also note that with simple hardware, all such control messages can be generated in constant time.

F. An Example of the Operation

The major principles of the DM switch can be illustrated with a simple example shown in Fig. 2 on a small 3×3 DM switch for 4 time slots where $W = 3$. The contention buffer is shown at the top of each middle node.

At time slot 0, I_i is connected to M_r by SF1 where $r = i$. I_0 finds a reservation frame for O_1 in its buffer, and sends a reservation packet. I_1 and I_2 both have buffered packets, destined to O_1 and O_2, respectively. However, the current contention buffer location is assumed to be the right most slot, and both of the buffer locations in M_1 and M_2 are occupied, hence the packets cannot be sent. O_0's frame check point is time slot 0, and it finds a reservation frame and receives the packet stored at M_0. The rest of the middle nodes and output ports, i.e., M_1, M_2, O_1 and O_2 run in the connection mode. M_1 is storing two packets belonging to two flows $F(1,2)$ and $F(2,1)$, where both packets are head-of-flow packets. M_2 is storing two packets belonging to two flows $F(2,1)$ and $F(2,2)$, where the packet of $F(2,2)$ is a head-of-flow packet while the packet of $F(2,1)$ is not. Therefore, there

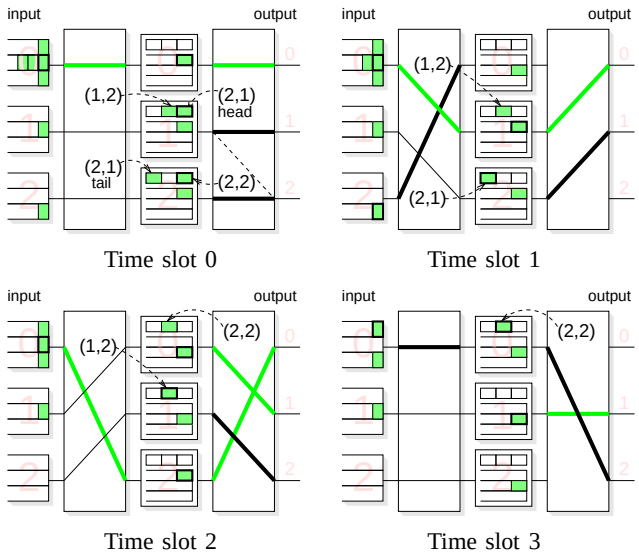

Fig. 2. An example of a 3×3 DM switch. A highlighted cell represents a packet being transfered. A bold line represents a packet transfer, where the green and black line represent the transfer of a reservation packet and a contention packet, respectively. A non-bold solid line represents a connection. A dashed line represents a request from a middle node to an output port.

exist edges between (M_1, O_1), (M_1, O_2), and (M_2, O_2). The matching algorithm finds a matching with two edges.

At time slot 1, I_i is connected to M_r by SF1 where $r = (i+1) \mod 3$. I_0 continues to send the reservation packet. I_1 is connected to M_2 where the contention buffer location is free; however, the packet in its buffer still cannot be sent due to Individual Block because it is destined to O_1 while there are reservation packets in the buffers for O_1. I_2 sends a contention packet to M_0. O_0 receives the reservation packet from M_1. M_0, M_2, O_1 and O_2 run in the connection mode, and the packet belonging to $F(2,1)$ stored in M_2 becomes the head-of-flow packet, and is sent.

At time slot 2, I_i is connected to M_r by SF1 where $r = (i+2) \mod 3$. I_0 continues to send the reservation packet. I_1 still cannot send the packet due to Individual Block. O_0 receives the reservation packet from M_2. As this time slot the frame checking point of O_1, O_1 finds a reservation frame and starts to receive the reservation packet. M_1 and O_2 run in the connection mode, and the contention packet in M_1 is sent.

At time slot 3, I_i is connected to M_r by SF1 where $r = i$. I_0 sends a contention packet. I_1 still cannot send the packet due to Individual Block. O_1 receives the reservation packet from M_1. M_0, M_2, O_0 and O_2 run in the connection mode, and the packet in M_0 is sent.

III. Achieving 100% Throughput

In this section, we prove important theoretical properties of the DM switch. We show that if the matching algorithm is capable of finding a maximal matching, the number of buffered packets in the DM switch for any output port is never more than a constant away from the number of buffered packets for the same output port in an output-buffered switch subject to the same arrival process, therefore the DM switch achieves 100% throughput.

First, we note that

Lemma 1: The number of packets in the buffer of an input port is never more than N^2.

Proof: We use induction. Consider an input port I_i. The claim is clearly true at the beginning of time slot i when I_i is first connected to M_0. Suppose this is still true till the beginning of time slot $i + kN$, we will prove that it is still true till the beginning of time slot $i + (k+1)N$. To see this, note that if there is a VOQ whose size is no less than N at the beginning of time slot $i + kN$, the input will start to send a reservation frame. As there is one departure and at most one arrival per time slot, the total number of buffered packets from time slot $i+kN$ to the beginning of time slot $i+(k+1)N$ will not be more than that at the beginning of time slot $i + kN$, and our claim is true due to the induction hypothesis. If there is no VOQ whose size is no less than N at the beginning of time slot $i + kN$, the total number of packets cannot exceed $N(N-1)$, therefore it cannot be greater than N^2 after only N time slots. [1] ■

Consider a conceptual queue U_j storing only the reservation packets in the DM switch for O_j, and let $U_j(t)$ be the length of it at time slot t. Let U_j^* denote the output queue for the same output port in an output-buffered switch, whose input process is the same as the arrival process of the reservation packets at the middle nodes of the DM switch, and let $U_j^*(t)$ be the length of it at time slot t. The key of our proof is the following lemma.

Lemma 2: $U_j(t) \leq U_j^*(t) + N^2 W + N$ for any t if the scheduling algorithm always finds a maximal matching.

Proof: Suppose U_j becomes nonempty at time slot T_0 and stays nonempty until time slot T_1. We claim that $U_j(t) \leq U_j^*(t) + N^2 W + N$ for any $T_0 \leq t \leq T_1$. The same fact holds for any non-empty periods of U_j and the bound can be established.

We say U_j is *idle* when O_j is not receiving reservation packets. We prove this claim by arguing that there can be at most $N^2 W + N$ time slots when U_j is idle from T_0 to T_1. Let $A_j(t)$ and $L_j(t)$ be the number of packets arrived at U_j and the number of idle slots of U_j till time slot t since T_0, respectively. Clearly,

$$U_j(t) = A_j(t) - (t - T_0) + L_j(t).$$

On the other hand, as U_j and U_j^* are subject to the same input process,

$$U_j^*(t) \geq A_j(t) - (t - T_0),$$

because the output-buffered queue is work-conservative. Therefore, if $L_j(t) \leq N^2 W + N$ for all t, our claim is true.

To see this, we note that the first frame check point of O_j starts at most N time slots after T_0. We show that after the first frame checking point, U_j is idle for a total of no more than $N^2 W$ time slots, hence proving the claim. Note that if the packet at the head of U_j is a head-of-flow packet, the transmission of a frame can begin at a frame checking

[1] The same result as Lemma 1 has been proved in [1] for the CR switch. The proof here is different and is provided for completeness.

point. The transmission cannot begin, i.e., U_j is forced to be idle, only in the case when the packet at the head of U_j is not a head-of-flow packet. In this case, there must be a contention packet belonging to the same flow stored in the contention buffer of some middle nodes. We prove that if U_j is idle for $N^2 W$ time slots after the first frame checking point, all contention packets destined to O_j are sent during these time slots. Note that because of the Individual Block, from time T_0, the DM switch will refrain all inputs from sending contention packets destined to O_j into the middle nodes until T_1. Therefore, there will be no new contention packets arrived at the middle nodes and U_j will not be idle for the rest of the time slots till T_1.

Note that at the frame checking point of O_j, if the frame cannot be sent, U_j will be idle for the next N time slots. We call such N idle time slots an *idle frame*. The idle periods of U_j will be a set of idle frames. Consider a middle node, say M_r. During each idle frame, there is at least one time slot in which M_r is not sending reservation packet, because U_j is idle. Therefore, M_r will have the opportunity to send a contention packet. If the total idle period after the first frame check point is no less than N^2W, there are at least NW idle frames and M_r will have a total of no less than NW opportunities. During these opportunities, M_r is either free, or is sending contention packets to O_j, or is sending contention packets to output ports other than O_j. Denote the time spent in each category before all contention packets to O_j stored in M_r are sent as ω_0, ω_1, and ω_2, respectively. As a middle node holds at most W contention packets, we have $\omega_1 + \omega_2 \leq W$. ω_0 is at most $(N-1)W$, because first, if there are still contention packets to O_j at M_r while M_r is idle, it must be the case that O_j is receiving a contention packet, as the matching is maximal. Second, there can be at most $(N-1)W$ contention packets stored in other middle nodes, because with Global Block, before all contention packets to O_j that are older than the reservation packets to O_j are sent, the DM switch is refraining all contention packets be sent to the middle nodes. As a result, after N^2W idle time slots, there cannot be any contention packets to O_j left in M_r. The claim is proved as the same fact holds for all middle nodes. ■

With Lemma 2, we may prove Lemma 3 using a very similar approach as Lemma 11 in [1]. The discussion is provided here for completeness. Basically, let $A_1(t)$, $A_2(t)$, $A_3(t)$ be the number of arrived packets destined to O_j up to time t at the DM switch, the middle nodes of the DM switch, and U_j, respectively. Let $Y_g(t)$ be the number of buffered packets for O_j at an ideal output-buffered switch when the arrival process is $A_g(t)$ for $g = 1, 2, 3$. Lemma 2 establishes that $U_j(t) \leq Y_3(t) + N^2 W + N$. Because the number of arrival at U_j is no more than the number of arrivals at the middle nodes, Lemma 11 in [1] shows that $Y_3(t) \leq Y_2(t)$. Also, Lemma 1 shows that the number of buffered packets in an input buffer cannot exceed N^2, which leads to $A_2(t) \geq A_1(t) + N^3$, and Lemma 11 in [1] shows that $Y_2(t) \leq Y_1(t) + N^3$. Therefore, $U_j(t) \leq Y_1(t) + N^3 + N^2 W + N$. Noting that the DM switch can buffer at most $N^3 + NW$ packets besides the reservation packets, we have

Lemma 3: The total number of packets buffered in the DM switch destined to O_j can be at most $2N^3 + N^2 W + NW + N$ more than that in the output queue of O_j in an ideal output-buffered switch subject to the same arrival process if the matching algorithm always finds maximal matchings.

Therefore,

Theorem 1: The DM switch achieves 100% throughput if the matching algorithm always finds maximal matchings.

IV. Simulations

We tested the performance of the DM switch with extensive simulations on a 32 by 32 switch where $W = 4096$. In the simulation, the arrival of packets at an input port is independent of other input ports. Two arrival processes are considered, Bernoulli and bursty. If Bernoulli, a packet arrives at an input port with a certain probability independent of other time slots. If bursty, a burst of packets arrive consecutively at an input port, the length of which is random and follows truncated Pareto distribution, where the probability that the burst length is s is $C/s^{2.5}$ for $1 \leq s \leq 1000$ and $C = 1/(\sum_{s=1}^{1000} s^{2.5})$ [1]. Two methods are simulated to determine the destination of a packet: uniform across all output ports or non-uniform. If non-uniform, a packet arrives at I_i is destined to O_i with probability $\mu + (1-\mu)/N$ and is destined to O_j with probability $(1-\mu)/N$ for $j \neq i$. μ is the "unbalance factor" and is set to be 0.5 [9]. The destinations of packets under Bernoulli arrival are chosen independently; the destinations of packets belonging to the same burst are the same. We therefore have four types of traffic: the Bernoulli uniform traffic (BERUNI), the Bernoulli non-uniform traffic (BERNUN), the bursty uniform traffic (BURUNI), and the bursty non-uniform traffic (BURNUN). For comparison, the results of three other switches are also shown: the input-buffered switch with no speed up running the iSLIP algorithm for 4 iterations (IQ), the ideal output-buffered switch (OQ), and the CR switch (CR) [1].

Fig. 3 shows performance of the switches measured by the packet delay. We note that first, *the DM switch achieves better performance than the CR switch when the traffic load is not high and achieves similar performance when the traffic load is high for all four types of traffic.* This is not a surprise because when the traffic load is low, both switches run in the contention mode, but the DM switch employs the matching algorithm which is more efficient than the contention method of the CR switch. When the load is medium, the DM switch continues to operate in contention mode while the CR switch may have activated the reservation mode which incurs longer delay. When the traffic is high, both switches operate in the reservation mode thus have similar performances.

Second, *under non-uniform traffic, the DM switch achieves similar or better performance than the IQ switch.* When the traffic load is not high, two switches has similar performance. The DM switch shows advantage when the traffic load is high due to the activation of the reservation mode, while the IQ switch cannot keep up with the load because the iSLIP algorithm cannot converge to best matchings when the traffic

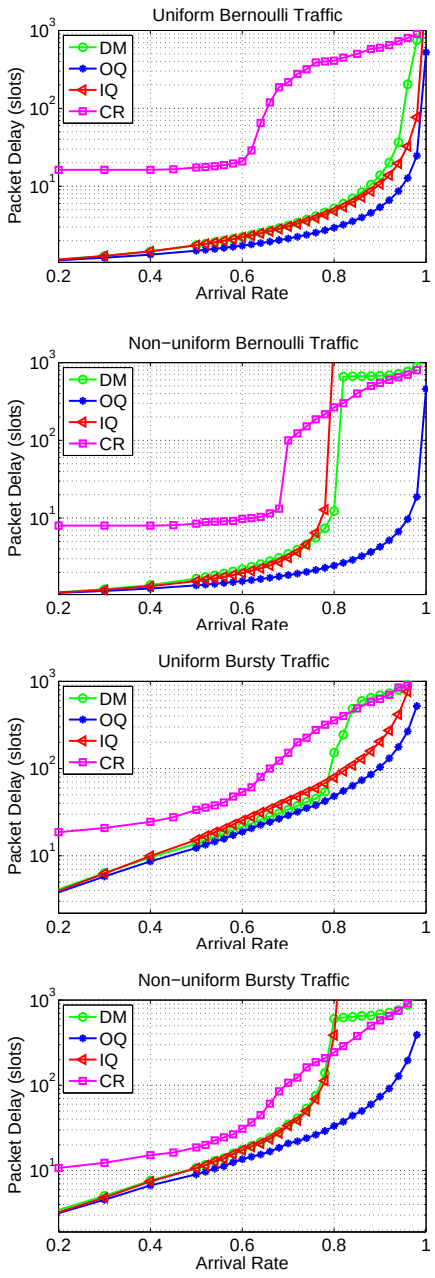

Fig. 3. The performance of the switches.

is not uniform. *Under uniform traffic, the DM switch achieves similar performance as the IQ switch when the traffic load is not high; however, the IQ switch has better performance when the traffic load is high.* It is well-known that the iSLIP algorithm is capable of finding good matchings when the traffic is uniform. The DM switch runs in the contention mode when the load is not high, hence has similar performance as the IQ switch. It activates the reservation mode when the load is high, which incurs longer delay than the IQ switch.

V. Related Work

The DM switch is inspired by the two-stage switch [1], [2], [3], [4], [5], [6], because both employs two switching fabrics. The key difference is that the second stage in the DM switch is configured according to a matching algorithm for the contention packets, while both stages of the two-stage switch follow a fixed connection pattern. Adding more intelligence to the configuration of the switching fabric, the DM switch does not have to invoke reservation mode until high load. As a result, unlike switches adopting a fixed switching schedule at both stages such as switches proposed in [1], [2], the DM switch does not suffer the $N/2$ delay at low and medium loads.

The CIOQ switch achieves 100% throughput [10], [11]. It uses only one switching fabric which requires a certain speedup. The DM switch employs two switching fabrics without speedup. As of today, the DM switch may offer a more attractable tradeoff between cost and performance, because maintaining the speedup may become increasingly more difficult as the line rate continues to increase.

The DM switch uses the IFQ and the timestamp-indexed contention buffer for maintaining packet order. Similar IFQ and timestamp-indexed buffer were adopted in the OpCut switch [12]. However, the OpCut switch does not support the reservation mode and cannot guarantee 100% throughput.

VI. Conclusions

In this paper, we propose the Distribute and Match (DM) switch that achieves 100% throughput while maintaining packet order without requiring speedup of the switching fabric. The DM switch has two switching stages, the distribution stage and the matching stage, and forwards packets either according to a matching algorithm or based on reservation. It achieves good performance because the matching stage is very efficient in forwarding packets up to medium load; while under high load, it relies on reservation and can sustain 100% throughput. Our simulations confirm that the DM switch achieves overall better performance than compared implementable switches.

References

[1] C.-L. Yu, C.-S. Chang, and D.-S. Lee "CR switch: a load-balanced switch with contention and reservation," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1659-1671, 2009.

[2] B. Hu and K.L. Yeung, "Feedback-based scheduling for load-balanced two-stage switches," *IEEE/ACM Transactions on Networking*, vol.18, no.4, pp. 1077-1090, Aug. 2010.

[3] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," *ACM SIGCOMM*, 2003.

[4] C.-S. Chang, D.-S. Lee, Y.-J. Shih and C.-L. Yu, "Mailbox switch: a scalable two-stage switch architecture for conflict resolution of ordered packets," *IEEE Transactions on Communications*, vol. 56, pp. 136-149, 2008

[5] 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, Apr. 2002.

[6] J.-J. Jaramillo, F. Milan, and R. Srikant, "Padded frames: a novel algorithm for stable scheduling in load balanced switches," Proceedings of *CISS*, Princeton, NJ, March 2006.

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

[8] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319352, Nov. 1993.

[9] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Chao, “CIXB-1: Combined input-one-cell-crosspoint buffered switch,” *HPSR 2001*, pp. 324329, Dallas, TX, May 2001.

[10] S.-T. Chuang, A. Goel, N. McKeown, B. Prabhakar, “Matching output queuing with a combined input output queued switch,” *IEEE Journal on Selected Areas in Communications*, vol.17, pp. 1030-1039, 1999.

[11] B. Prabhakar and N. McKeown, “On the speedup required for combined input and output queued switching,” *Automatica*, vol. 35, no. 12, Dec. 1999.

[12] L. Liu, Z. Zhang, and Y. Yang, “Packet scheduling in a low latency optical packet switch,” *HPSR 2010*.