

Mailbox Switch: A Scalable Two-stage Switch Architecture for Conflict Resolution of Ordered Packets

Cheng-Shang Chang, Duan-Shin Lee, and Ying-Ju Shih
Institute of Communications Engineering
National Tsing Hua University
Hsinchu 300, Taiwan, R.O.C.

Email: cschang@ee.nthu.edu.tw lds@cs.nthu.edu.tw yjshih@gibbs.ee.nthu.edu.tw

Abstract—Traditionally, conflict resolution in an input-buffered switch is solved by finding a matching between inputs and outputs per time slot. To do this, a switch not only needs to gather the information of the virtual output queues at the inputs, but also uses the gathered information to compute a matching. As such, both the communication overhead and the computation overhead make it difficult to scale. Recent works on the two-stage switch architecture in [6], [7], [12], [8] showed that conflict resolution can be easily solved over time and space without communication and computation overhead. However, the main problem of such a two-stage switch architecture is that packets might be out of sequence. The main objective of this paper is to propose a scalable solution, called the mailbox switch, that solves the out-of-sequence problem in the two-stage switch architecture. The key idea of the mailbox switch is to use a set of *symmetric* connection patterns to create a feedback path for packet departure times. With the information of packet departure times, the mailbox switch can schedule packets so that they depart in the order of their arrivals. Despite the simplicity of the mailbox switch, we show via both the theoretical models and simulations that the throughput of the mailbox switch can be as high as 75%. With limited resequencing delay, a modified version of the mailbox switch achieves 95% throughput. We also propose a recursive way to construct the switch fabrics for the set of symmetric connection patterns. If the number of inputs, N , is a power of 2, we show that the switch fabric for the mailbox switch can be built with $\frac{N}{2} \log_2 N \times 2 \times 2$ switches.

Index Terms—Birkhoff-von Neumann switches, input-buffered switches, conflict resolution, two-stage switches

This research is supported in part by the National Science Council, Taiwan, R.O.C., under Contract NSC-91-2219-E007-003, and the program for promoting academic excellence of universities 89-E-FA04-1-4.

I. INTRODUCTION

As the parallel input buffers of input-buffered switches provide the needed speedup for memory access speed, input-buffered switches are known to be more scalable than shared memory switches. However, synchronized parallel transmissions among parallel input buffers in every time slot require careful coordination to avoid conflicts. Thus, finding a scalable method (and architecture) for conflict resolution becomes the fundamental design problem of input-buffered switches.

Traditionally, conflict resolution is solved by finding a matching between inputs and outputs per time slot (see e.g., [11], [1], [22], [17], [18], [19], [9], [16]). Two steps are needed for finding a matching.

- (i) Communication overhead: one has to gather the information of the buffers at the inputs.
- (ii) Computation overhead: based on the gathered information, one then applies a certain algorithm to find a matching.

Most of the works in the literature pay more attention to reducing the computation overhead by finding scalable matching algorithms, e.g., wavefront arbitration in [22], PIM in [1], SLIP in [17], and DRRM in [16]. However, in our view, it is the communication overhead that makes matching per time slot difficult to scale. To see this, suppose that there are N inputs/outputs and each input implements N virtual output queues (VOQ). If we use a single bit to indicate whether a VOQ is empty, then we have to transmit N bits from each input (to a central arbiter or to an output) in every time slot. For instance, transmitting such N bit information in PIM and SLIP is implemented by an independent circuit that sends out parallel requests. Suppose that the packet size is chosen to be 64 bytes. Then building a switch with more than 512 inputs/outputs will have more communication overhead than transmitting the data itself.

To reduce the communication overhead, one approach is to gather the long term statistics of the VOQs, e.g., the average arrival rates, and then use such information to find a sequence of pre-determined connection patterns (see e.g., [1], [14], [10], [4], [5], [2]). Most of the works along this line are based on the well-known Birkhoff-von Neumann algorithm [3], [23] that decomposes a doubly substochastic matrix into a convex combination of (sub)permutation matrices. For an $N \times N$ switch, the computation complexity for the Birkhoff-von Neumann decomposition is $O(N^{4.5})$ and the number of permutation matrices produced by the decomposition is $O(N^2)$ (see e.g., [4], [5]). The need for storing the $O(N^2)$ number of permutation matrices in the Birkhoff-von Neumann switch makes it difficult to scale for a large N . Even though there are decomposition methods that reduce the number of permutation matrices (see e.g., [13]), they in general do not have good throughput. For instance, the throughput in [13] is $O(1/\log N)$ and it tends to 0 when N is large. Another problem of using long term statistics is that the switch does not adapt too well to traffic fluctuation.

It would be ideal if there is a switch architecture that yields good throughput without the need for gathering traffic information (no communication overhead) and computing connection patterns (no computation overhead). Recent works on the two-stage switches (see e.g., [6], [7], [12], [8]) shed some light along this direction. The switch architecture in [6], called the load balanced Birkhoff-von Neumann switch, consist of two crossbar switch fabrics and parallel buffers between them. In a time slot, both the crossbar switch fabrics sets up connection patterns corresponding to permutation matrices that are periodically generated from a one-cycle permutation matrix. By so doing, the first stage performs load balancing for the incoming traffic so that the traffic coming into the second stage is uniform. As such, it suffices to use the same periodic connection patterns as in the first stage to perform switching at the second stage. In the load balanced Birkhoff-von Neumann switch, there is no need to gather the traffic information. Also, as the connection patterns are periodically generated, no computation is needed at all. More importantly, it can be shown to achieve 100% throughput for any *non-uniform* traffic under a minor technical assumption. However, the main drawback of the load balanced Birkhoff-von Neumann switch in [6] is that packets might be out of sequence. To solve the out-of-sequence problem in the two-stage switches, two approaches have been proposed. The first one uses sophisticated scheduling in the buffers between the two switch fabrics (see e.g., [7], [12]) and hence it may require complicated hardware implementation and

non-scalable computation overhead. The second one is to use the rate information for controlling the traffic entering the switch (see e.g., [8]). However, this requires communication overhead and it also does not adapt too well to large traffic fluctuation.

One of the main objectives of this paper is to solve the out-of-sequence problem in the two-stage switch without non-scalable computation and communication overhead. For this, we propose a switch architecture, called the mailbox switch. The mailbox switch has the same architecture as the load balanced Birkhoff-von Neumann switch. Instead of using an arbitrary set of periodic connection patterns generated by a one-cycle permutation matrix, the key idea in the mailbox switch is to use a set of *symmetric* connection patterns. As an input and its corresponding output are usually built on the same line card, the symmetric connection patterns set up a feedback path from the central buffers (called mailboxes in this paper) to an input/output port. Since everything inside the switch is pre-determined and periodic, the scheduled packet departure times can then be fed back to inputs to compute the waiting time for the next packet so that packets can depart in sequence. Thus, the communication overhead incurred by this is the transmission of the information of the packet departure time, which is constant in every time slot for every input port. This communication overhead in every time slot for every input port is independent of the size of the switch. On the other hand, the computation overhead incurred by this is the computation of the waiting time, which also requires only a constant number of operations. Simplicity comes at the cost of throughput. The throughput of the mailbox switch is no longer 100%. Both our theoretical models and simulations show that it can achieve more than 75% throughput. Our theoretical results also show that a special case of the mailbox switch reduces to the classical head-of-line blocking switch in [11] that yields 58% throughput. By allowing limited resequencing delay, a modified version of the mailbox switch can achieve more than 95% throughput.

In this paper, we also propose a recursive way to construct the switch fabrics for the set of symmetric connection patterns. If the number of inputs, N , is a power of 2, we show that the switch fabric for the mailbox switch can be built with $\frac{N}{2} \log_2 N$ 2×2 switches.

II. THE SWITCH ARCHITECTURE

A. Generic mailbox switch

In this paper, we assume that packets are of the same size. Also, time is slotted and synchronized so that a packet can be transmitted within a time slot. As in

the load balanced Birkhoff-von-Neumann switch, the $N \times N$ mailbox switch consists of two $N \times N$ crossbar switch fabrics (see Figure 1) and buffers between the two crossbar switch fabrics. The buffers between the two switch fabrics are called mailboxes. There are N mailboxes, indexed from 1 to N . Each mailbox contains N bins (indexed from 1 to N), and each bin contains F cells (indexed from 1 to F). Each cell can store exactly one packet. Cells in the i^{th} bin of a mailbox are used for storing packets that are destined for the i^{th} output port of the second switch. In addition to these, a First In First Out (FIFO) queue is added in front of each input port of the first stage.

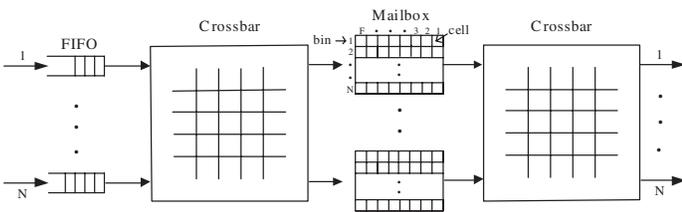


Fig. 1. The switch architecture

Now we describe how the connection patterns of these two crossbar switch fabrics are set up. In every time slot, both crossbar switches in Figure 1 have the same connection pattern. During the t^{th} time slot, input port i is connected to the output port j if

$$(i + j) \bmod N = (t + 1) \bmod N. \quad (1)$$

In particular, at $t = 1$, we have input port 1 connected to output port 1, input port 2 connected to output port $N-1$, ..., and input port N connected to output port 2. Clearly, such connection patterns are periodic with period N . Moreover, each input port is connected to each of the N output ports exactly once in every N time slot. Specifically, input port i is connected to output port 1 at time i , output port 2 at time $i + 1$, ..., output port N at time $i + N - 1$. Also, we note from (1) that such connection patterns are *symmetric*, i.e., input port i and output port j are connected if and only if input port j and output port i are connected. As such, we call a switch fabric that implements the connection patterns in (1) a *symmetric Time Division Multiplexing (TDM) switch*. Note that one can solve j in (1) by the following function

$$j = h(i, t) = \left((t - i) \bmod N \right) + 1. \quad (2)$$

Thus, during the t^{th} time slot the i^{th} input port is connected to the $h(i, t)^{th}$ output port of these two crossbar switch fabrics.

As input port i of the first switch and output port i of the second switch are on the same line card, the symmetric property then enables us to establish a bi-directional communication link between a line card and a mailbox. As we will see later, such a property plays an important role in keeping packets in sequence.

As the connection patterns in the mailbox switch is a special case of the load-balanced Birkhoff-von Neumann switch with one-stage buffering [6], one might expect that it also approaches 100% throughput if we use the FIFO policy for each bin and increase the bin size F to ∞ . However, we also suffer from the out-of-sequence problem by doing this. Packets that have the same input port at the first switch and the same output port at the second switch may be routed to different mailboxes and depart in a sequence that is different from the sequence of their arrivals at the input port of the first switch.

To solve the out-of-sequence problem, one may add a resequencing buffer and adapt a more careful load balancing mechanism as in the load balanced Birkhoff-von Neumann switch with multi-stage buffering [7]. However, such an approach requires complicated scheduling and jitter control in order to have a bounded resequencing delay. Here we take a much simpler approach. The idea is that we do know the packet departure time once it is placed in a mailbox as the connection patterns are deterministic and periodic. Also, as an input port of the first switch and the corresponding output port of the second switch are in general built on the same line card, the information of packet departure times can be fed back to the inputs so that packets can be scheduled in the order of their arrivals.

To be specific, define flow (i, j) as the sequence of packets that arrives at the i^{th} input port of the first switch and are destined for the j^{th} output port of the second switch. Let $V_{i,j}(t)$ be the number of time slots that a packet of flow (i, j) has to wait once it becomes the head-of-line (HOL) packet at the FIFO queue of the i^{th} input port of the first stage at time t . Following the terminology in queueing theory, we call $V_{i,j}(t)$ the virtual waiting time of flow (i, j) . Now we describe how the mailbox switch works to keep packets of the same flow in sequence.

- (iA) **Retrieving mails:** at time t , the i^{th} output port of the second switch is connected to the $h(i, t)^{th}$ mailbox. The packet in the first cell of the i^{th} bin is transmitted to the i^{th} output port. Packets in cells $2, 3, \dots, F$ of the i^{th} bin are moved forward to cells $1, 2, \dots, F - 1$.
- (iiA) **Sending mails:** suppose that the HOL packet of the i^{th} input port of the first switch is from flow (i, j) . Note that the i^{th} input port of the

first switch is also connected to the $h(i, t)^{th}$ mailbox. In order to keep packets in sequence, this HOL packet is placed in the first empty cell of the j^{th} bin of the $h(i, t)^{th}$ mailbox such that it will depart not earlier than $t + V_{i,j}(t)$. If no such empty cell can be found, the HOL packet is blocked and it remains the HOL packet of that FIFO queue.

(iiiA) **Updating virtual waiting times:** all the flows that do not send mails (packets) at time t decrease their virtual waiting time by 1. This includes flows that have blocked transmissions. To update the virtual waiting time for flow (i, j) , suppose that the HOL packet is placed in the f^{th} cell of the j^{th} bin of the $h(i, t)^{th}$ mailbox. As the connection patterns are deterministic and periodic, one can easily verify that the $h(i, t)^{th}$ mailbox will be connected to the j^{th} output port of the second switch at $t + ((j - i - 1) \bmod N) + 1$. Thus, the departure time for this packet is simply $t + (f - 1)N + ((j - i - 1) \bmod N) + 1$. As such, the number of time slots that has to be waited at $t + 1$ for flow (i, j) is $(f - 1)N + ((j - i - 1) \bmod N)$ and we have

$$V_{i,j}(t + 1) = (f - 1)N + ((j - i - 1) \bmod N). \quad (3)$$

B. Mailbox switch with cell indexes

In view of (3), there is a simple way to represent the virtual waiting time of a flow. The virtual waiting time $V_{i,j}(t + 1)$ can be written as a sum of two components: $(f - 1)N$ and $((j - i - 1) \bmod N)$. The first term is only a function of the cell index f and the second term is a number between 0 and $N - 1$. This leads to a much easier way to implement the mailbox switch. Define $f_{i,j}(t)$ to be the smallest index of the cell such that the HOL packet will not depart earlier than $t + V_{i,j}(t)$ if the HOL packet is placed in that cell. To simplify our representation, we call $f_{i,j}(t)$ the cell index of $V_{i,j}(t)$. Now we modify the second and the third phase as follows:

(iiB) **Sending mails:** suppose that the HOL packet of the i^{th} input port of the first switch is from flow (i, j) . This HOL packet is sent to the $h(i, t)^{th}$ mailbox along with $f_{i,j}(t)$. This packet is then placed in the first empty cell of the j^{th} bin with the cell index not smaller than $f_{i,j}(t)$. If successful, the index of that cell, say f , is transmitted to the i^{th} output port of the second switch. If no such empty cell can be found, an error message, say $f = 0$, is transmitted to the

j^{th} output port of the second switch to indicate a HOL blocking.

(iiiB) **Updating virtual waiting times:** in addition to the cell index of the virtual waiting time $f_{i,j}(t)$, we also keep a counter $g_{i,j}(t)$ for flow (i, j) . If flow (i, j) has a successful transmission of a packet at time t , then $f_{i,j}(t + 1)$ is set by the index f returned by the mailbox at time t and $g_{i,j}(t + 1)$ is reset to N . On the other hand, if flow (i, j) does not have a successful transmission of a packet at time t , then $f_{i,j}(t + 1) = f_{i,j}(t)$ and $g_{i,j}(t + 1) = g_{i,j}(t) - 1$. If $g_{i,j}(t + 1)$ is reduced to 0, then we reset $g_{i,j}(t + 1)$ back to N . When this happens and $f_{i,j}(t + 1) > 1$, we decrease $f_{i,j}(t + 1)$ by 1.

In view of (3) and (iiiA), the virtual waiting time $V_{i,j}(t)$ can be represented by $f_{i,j}(t)$ and $g_{i,j}(t)$ as follows:

$$V_{i,j}(t) = \max[(f_{i,j}(t) - 1)N + ((j - i - 1) \bmod N) - (N - g_{i,j}(t)), 0]. \quad (4)$$

The main advantage of the scheme that uses cell indexes is that there is no need to transmit the whole information of the virtual waiting times. Instead, only cell indexes are transmitted. This greatly reduces the communication overhead needed in the mailbox switch. Also, it is easier to place a HOL packet in a mailbox by using the cell index of its virtual waiting time.

C. Mailbox switch with a limited number of forward tries

Note that the mailbox switch resolves conflict implicitly over *time* and *space*. First, packets are distributed evenly to the N mailboxes via the symmetric TDM switch at the first stage. Intuitively, one may view this as conflict resolution over space. Once a packet is transmitted to a mailbox, the mailbox switch has to find an empty cell with its cell index not smaller than the cell index of the virtual waiting time of the packet. As cells in the same bin are ordered in the FIFO manner, this can be viewed as conflict resolution over time. In the search for an empty cell to place the packet, there might be several tries until an empty cell is found. For each unsuccessful try, it may be viewed as a ‘‘collision,’’ and each collision leads to back off N time slots for the packet departure time. Such a backoff not only affects the packet being placed, but also affects all the subsequent packets that belong to the same flow because the virtual waiting time of that flow is also increased by N time slots. If there are many collisions, the increase of the virtual waiting time will be large and eventually packets

will be distributed over time *sparsely*. This will result in low throughput and large delay. To avoid such an event, it might be better to block the packet by putting a limit on the amount of virtual waiting time that can be increased for each placement. This leads to the following modified scheme.

(iiC) **Sending mails:** let δ be the maximum increment of the cell index of the virtual waiting time. We only search for an empty cell from the cell $f_{i,j}(t)$ to the cell $\min[f_{i,j}(t) + \delta, F]$. If successful, the index of that cell, say f , is transmitted to the i^{th} output port of the second switch. If no such empty cell can be found, an error message, say $f = 0$, is transmitted to the i^{th} output port of the second switch to indicate a HOL blocking.

D. Mailbox switch with limited numbers of forward and backward tries

To perform conflict resolution more efficiently over *time*, we may also search for an empty cell with a limited number of *backward* tries. By so doing, packets in the mailbox switch might be out of sequence. But resequencing delay is bounded.

(iiD) **Sending mails:** let δ_b be the maximum number of backward tries. We only search for an empty cell from the cell $\max[f_{i,j}(t) - \delta_b, 1]$ to the cell $\min[f_{i,j}(t) + \delta, F]$. If successful, the index of that cell, say f , is transmitted to the i^{th} output port of the second switch. If no such empty cell can be found, an error message, say $f = 0$, is transmitted to the i^{th} output port of the second switch to indicate a HOL blocking.

(iiiD) **Updating virtual waiting times:** the case without a successful transmission of a packet is the same as (iiiB). For the case with a successful transmission of a packet, we have to deal with the following two subcases. If the returned index f is not smaller than $f_{i,j}(t)$, then it is the same as before. That is, we set $f_{i,j}(t+1) = f$ and reset $g_{i,j}(t+1)$ to N . On the other hand, if the returned index f is smaller than $f_{i,j}(t)$, then the packet being placed will depart earlier than its previous one. As such, it is treated in the same way as the case without a successful transmission of a packet.

Note that the resequencing delay in the scheme with backward tries is bounded by $N\delta_b$ time slots.

III. THEORETICAL MODELS

To further explain the mailbox switches, in this section we provide two theoretical models. In these models, we

assume that both the bin size F and the buffers for the FIFO queues at the input ports of the first switch are infinite. Also, we do not allow backward tries, i.e., $\delta_b = 0$.

A. Uniform i.i.d. traffic model

In our models, we consider the well-known uniform i.i.d. traffic model as in [11], [6]. Assume that the arrival processes to the input ports of the first switch satisfy the following conditions.

- (A1) Arrivals at each input port are independent and identical Bernoulli processes.
- (A2) All arrival processes have the same arrival rate ρ_a and every arrival process is independent of others.
- (A3) The destination of every arrival at each input port is uniformly distributed over N outputs.
- (A4) N is large.

In the following sections, we provide two theoretical models for the mailbox switches with $\delta = 0$ and $\delta > 0$.

B. Exact analysis for the throughput with $\delta = 0$

In this section, we consider the mailbox switch with $\delta = 0$. Since $\delta = 0$, the cell index of the virtual waiting time will never be increased. As such, there is no need to keep track of the virtual waiting times at all! Moreover, even though we assume that $F = \infty$ in our model, only the *first* cell in every bin is used. As such, it can be implemented with $F = 1$. Since $F = 1$, there is no need to transmit and feedback the cell index of the virtual waiting time. However, we still need to feedback a single bit information to indicate whether a HOL packet is successfully placed, i.e., $f = 0$ for a HOL blocking and $f = 1$ for a successful placement.

Our objective of this section is to show that this special case of the mailbox switch with $\delta = 0$ yields the same throughput as the classical HOL blocking switch in [11], i.e., it achieves 58% throughput. In fact, the mailbox switch with $\delta = 0$ can be viewed as a HOL blocking switch with distributed and pipelined conflict resolution.

As the traffic is uniform, we only need to consider a particular output port of the second switch, say, the first output. At time t , it is connected to the $h(1, t)^{\text{th}}$ mailbox, and this mailbox is also connected to the first input port of the first switch by symmetry. If the first bin of the $h(1, t)^{\text{th}}$ mailbox is occupied at the beginning of the t^{th} time slot, then the packet is retrieved by the first output port and the first bin becomes empty at time t . In any event, we know that the first bin of the $h(1, t)^{\text{th}}$ mailbox is empty at time t .

Let $Y_i(t) = 1$ if the HOL packet of the FIFO queue of the i^{th} input port is destined for the first output port of the second switch at time t , and $Y_i(t) = 0$ otherwise. Let

$$q(t) = \sum_{i=1}^N Y_i(t+i-1). \quad (5)$$

As the $h(1, t)^{th}$ mailbox is connected to the first input port at time t , it will be connected to i^{th} input port at time $t+i-1$. Thus, $q(t)$ is the total number of HOL packets that can be placed in the first bin of the $h(1, t)^{th}$ mailbox from t to $t+N-1$. If $q(t) \geq 1$, then there is exactly one HOL packet that will be placed in the first bin of the $h(1, t)^{th}$ mailbox as the bin is empty at time t . Those blocked HOL packets remain the HOL packets and they can be placed in the first bin of the $h(1, t+1)^{th}$ mailbox from $t+1$ to $t+N$. Thus, we have

$$q(t+1) = (q(t) - 1)^+ + a(t), \quad (6)$$

where $a(t)$ is the number of packets that becomes the HOL packets and can be placed in the first bin of the $h(1, t+1)^{th}$ mailbox from $t+1$ to $t+N$. Once we have the recursive equation in (6), we can follow the standard argument to show that the maximum throughput is $2 - \sqrt{2}$ (see e.g., [11], [21]). Specifically, we first assume that $\rho_a = 1$ in the uniform i.i.d. traffic model. As such, once a HOL packet is placed in a mailbox, it will be replaced by another packet that chooses its destination *uniformly and independently*. When N is large, $a(t)$ is the sum of a large number of Bernoulli random variables and may be viewed as a Poisson random variable. To have a stable system, the mean rate of $a(t)$ should be the same as the throughput ρ_d , i.e.,

$$E[a(t)] = \rho_d. \quad (7)$$

From the well-known result for the discrete-time $M/G/1$ queue, we then have in steady state

$$P(q(t) > 0) = \rho_d, \quad (8)$$

and

$$E[q(t)] = \frac{1}{2} \frac{2\rho_d - \rho_d^2}{1 - \rho_d}. \quad (9)$$

To find ρ_d , we note that the expected total number of blocked HOL packets is $NE(q(t)-1)^+$ and the expected number of departing HOL packets is $N\rho_d$. Since the total number of HOL packets is N , it follows that

$$NE(q(t) - 1)^+ + N\rho_d = N. \quad (10)$$

Since

$$E(q(t) - 1)^+ = E[q(t)] - P(q(t) > 0), \quad (11)$$

using (11), (8) and (9) in (10) yields $\rho_d = 2 - \sqrt{2}$.

C. Approximation for the throughput with $\delta > 0$

In this section, we provide a theoretical model for the mailbox switch with $\delta > 0$. The objective is to find a simple (approximation) formula for the maximum throughput of the mailbox switch. Unlike the case $\delta = 0$ in the previous section, the cell indexes of virtual waiting times $f_{i,j}(t)$'s might go to infinity once we have $\delta > 0$. Thus, in order to have a stable system, we have to make sure that both the FIFO queues at the input ports of the first switch and the cell indexes of the virtual waiting times do not go to infinity.

First, let us consider an FIFO queue, say the i^{th} queue, at the input port of the first switch. In order to have a stable queue, we need to make sure that the arrival rate to the queue is smaller than the service rate of the queue. From the uniform i.i.d. traffic model described in Section III-A, the arrival rate to the queue is simply ρ_a . To compute the service rate, consider a HOL packet of the queue at time t . Suppose the HOL packet is destined for the j^{th} output port of the second switch. The HOL packet is blocked only if there is no empty cell among the cells $f_{i,j}(t), f_{i,j}(t)+1, \dots, f_{i,j}(t)+\delta$. Let ρ_d be the throughput of the mailbox switch. As a packet eventually leaves the mailbox switch once it is placed in a cell, the throughput ρ_d is also the probability that a cell is occupied. To simplify our analysis, we make the following assumption on the independence of cell occupancy:

- (A5) Every cell is occupied *independently* with probability ρ_d . This is independent of everything else.

From (A5), the probability that the HOL packet is blocked is $\rho_d^{\delta+1}$. Thus, the service rate is $1 - \rho_d^{\delta+1}$. This leads to the following condition for the FIFO queue to be stable:

$$\rho_a < 1 - \rho_d^{\delta+1}. \quad (12)$$

Now we consider the cell index of the virtual waiting time for a particular flow, say flow (i, j) . In order for $f_{i,j}(t)$ to be stable, we need to make sure that the increase rate of $f_{i,j}(t)$ is smaller than the decrease rate of $f_{i,j}(t)$. To compute the increase rate, note that $f_{i,j}(t)$ is increased by k for some $0 \leq k \leq \delta$ if the following three conditions hold: (i) the HOL packet at the i^{th} input port of the first switch is a packet from flow (i, j) , (ii) the cells in the j^{th} bin with the indexes $f_{i,j}(t), f_{i,j}(t)+1, \dots, f_{i,j}(t)+k-1$ are occupied, and (iii) the cell with the index $f_{i,j}(t)+k$ is empty. From the uniform i.i.d. traffic model, the probability that the HOL packet at the i^{th} input port of the first switch is a packet from flow (i, j) is simply ρ_a/N . As everything

is assumed to independent in (A5), the probability that $f_{i,j}(t)$ is increased by k is

$$\frac{\rho_a}{N} \cdot \rho_d^k \cdot (1 - \rho_d).$$

Thus, the increase rate of $f_{i,j}(t)$ is

$$\begin{aligned} & \sum_{k=0}^{\delta} k \cdot \frac{\rho_a}{N} \cdot \rho_d^k \cdot (1 - \rho_d) \\ &= \frac{\rho_a}{N} \frac{\delta \rho_d^{\delta+2} - (\delta + 1) \rho_d^{\delta+1} + \rho_d}{1 - \rho_d}. \end{aligned} \quad (13)$$

To compute the decrease rate, note that $f_{i,j}(t)$ is decreased by 1 if the following two conditions hold: (i) there is no successful transmission of a packet from flow (i, j) , and (ii) the counter $g_{i,j}(t) = 1$. The event that there is no successful transmission of a packet from flow (i, j) can be decomposed as the union of the two disjoint events: the HOL packet at the i^{th} input port of the first switch is *not* a packet from flow (i, j) or the HOL packet at the i^{th} input port of the first switch is a *blocked* packet from flow (i, j) . Thus, the probability that there is no successful transmission of a packet from flow (i, j) is

$$1 - \frac{\rho_a}{N} + \frac{\rho_a}{N} \cdot \rho_d^{\delta+1}.$$

To compute the probability that $g_{i,j}(t) = 1$, we make the following assumption.

(A6) The counter $g_{i,j}(t)$ is uniformly distributed over $\{1, 2, \dots, N\}$.

As such, the probability that $g_{i,j}(t) = 1$ is simply $1/N$. Thus, the decrease rate of $f_{i,j}(t)$ is

$$\left(1 - \frac{\rho_a}{N} + \frac{\rho_a}{N} \rho_d^{\delta+1}\right) \frac{1}{N}. \quad (14)$$

Using (13) and (14) and letting $N \rightarrow \infty$, we have the following condition for the $f_{i,j}(t)$ to be stable:

$$\rho_a \frac{\delta \rho_d^{\delta+2} - (\delta + 1) \rho_d^{\delta+1} + \rho_d}{1 - \rho_d} < 1. \quad (15)$$

As the throughput ρ_d cannot be larger than the arrival rate ρ_a , it follows from (12) and (15) that throughput ρ_d is limited by the following two inequalities:

$$\rho_d + \rho_d^{\delta+1} < 1, \quad (16)$$

$$\rho_d \frac{\delta \rho_d^{\delta+2} - (\delta + 1) \rho_d^{\delta+1} + \rho_d}{1 - \rho_d} < 1. \quad (17)$$

In Figure 2, we use the bound obtained by (16) and (17) to plot the maximum throughput as a function of δ . For $\delta < 5$, the inequality in (16) sets the limit on the maximum throughput. On the other hand, for $\delta \geq 5$, the inequality in (17) sets the limit on the maximum throughput. From these, it is interesting to see that the

curve is peaked when $\delta = 4$ and that gives the maximum throughput of 0.755. The intuition behind this is that if we set δ too small, it is quite likely that the HOL blocking will become a problem. On the other hand, if we set δ too large, then packets will be distributed over time *sparse* and that also results in a low throughput. We also note that when $\delta \rightarrow \infty$, the mailbox switch has the maximum throughput 0.618, which is higher than $2 - \sqrt{2} \approx 0.58$ for the case with $\delta = 0$.

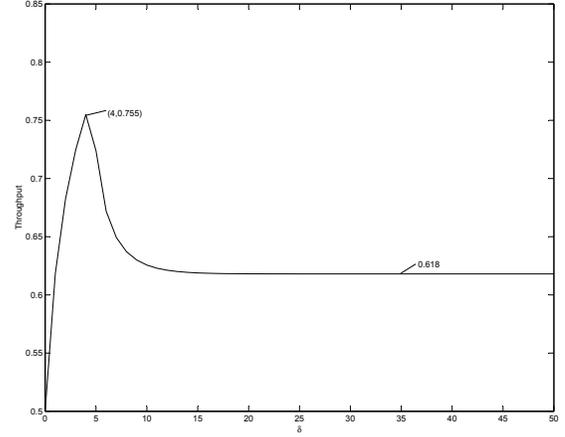


Fig. 2. The maximum throughput as a function of δ

IV. SIMULATION STUDY

In this section, we perform various simulations to verify our theoretical results in the previous section. In all our simulations, we consider 100×100 mailbox switches, i.e., $N = 100$. Our first experiment is to find the maximum throughput of the mailbox switch. To achieve this, the arrival rate of each input port is set to 1, i.e., a packet arrives at each input port in every time slot. In Figure 3, we plot the simulation results (along with the theoretical results) for the maximum throughput as a function of δ under the uniform i.i.d. traffic.

Note that the curve from the simulation results in Figure 3 is similar to that from the theoretical results. Both curves show that the throughput can be increased by increasing δ at the beginning, and it then starts to decrease if δ is increased further. As explained in our theoretical model, this is because the throughput is limited by the HOL blocking at the FIFO queues of the first switch when δ is small. On the other hand, when δ is large, the throughput is limited by the stability of the virtual waiting times. Thus, the throughput model based on the stability of the FIFO queues and the virtual waiting times seems to be valid (at least qualitatively).

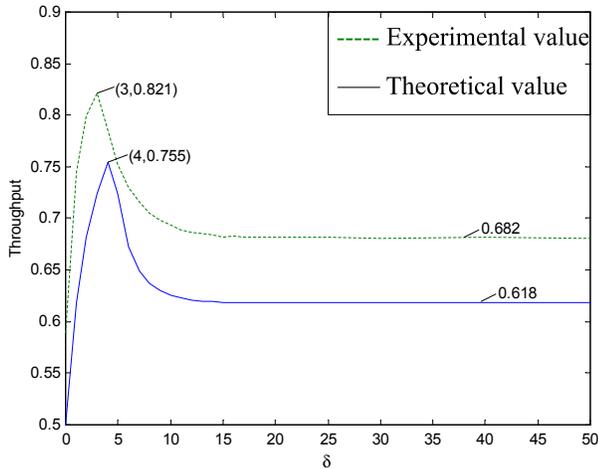


Fig. 3. The maximum throughput as a function of δ

However, the simulation results show that the mailbox switch achieves better throughput than predicted. In particular, as $\delta \rightarrow \infty$, the simulation results show that the mailbox switch has the maximum throughput 0.682, which is higher than 0.618 predicted by the theoretical model. The main reason behind this is that the independence assumption for cell occupancy in (A5) is an over simplified assumption. In fact, we expect that nonempty cells are more likely to be clustered together as we always search for the first empty cell. As such, packets destined for the same output are more well packed and the increase rate of the cell indexes of the virtual waiting times is not as large as predicted in (13). We also note that for the case $\delta = 0$ the simulation result shows the maximum throughput is 0.58 as predicted by our theoretical model.

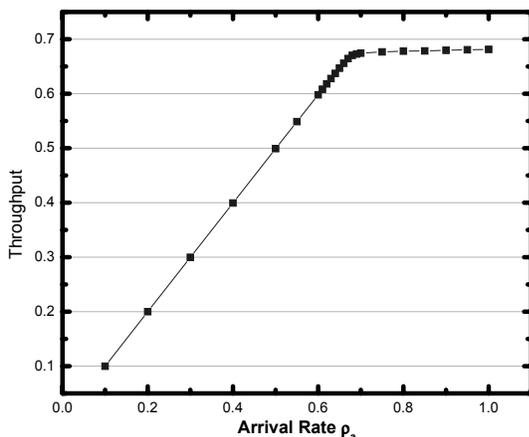


Fig. 4. Throughput as a function of the arrival rate ρ_a

In our second experiment, we measure the throughput by increasing the arrival rate ρ_a . For this experiment, we choose $\delta = 50$. In Figure 4, we plot the throughput as a function of the arrival rate ρ_a . Note that the throughput of the mailbox switch increases linearly as a function of the arrival rate ρ_a until it reaches its maximum throughput near 0.682. From that point on, the throughput remains almost unchanged even if the arrival rate ρ_a is increased further. This shows that the mailbox switch does not have the undesired catastrophic behavior in some random conflict resolution algorithms such as ALOHA and CSMA (see e.g., [20]), where the throughput decreases as the load is increased further.

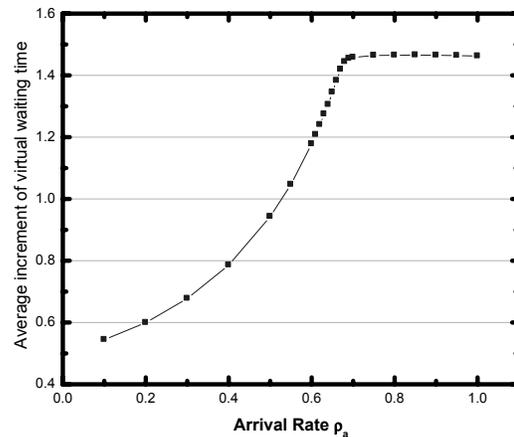


Fig. 5. Normalized average increment of the virtual waiting time as a function of the arrival rate ρ_a

In this experiment, we also measure the normalized average increment of the virtual waiting time when a packet is placed successfully in a mailbox. The normalized average increment of the virtual waiting time is obtained by the ratio of the average increment of the virtual waiting time to the number of input ports N . In Figure 5, we plot the normalized average increment of the virtual waiting time as a function of the arrival rate. Note from Figure 5 that the curve can be broken into two different segments near the maximum throughput 0.682: the stable segment with $\rho_a < 0.682$ and the overloaded segment with $\rho_a > 0.682$. In the stable segment, the product of the arrival rate and the normalized average increment of the virtual waiting time is less than 1. As the virtual waiting time is decreased by 1 per time slot, the virtual waiting time remains finite in the stable segment. To verify this, we plot in Figure 6 a sample path of the cell index of the virtual waiting time for flow (1, 50) when $\rho_a = 0.55$. In Figure 6, the cell index of the virtual waiting time exhibits the typical behavior

of a stable queue, i.e., the virtual waiting time returns to zero recurrently. Note that when the virtual waiting time is zero, its cell index is one. Thus, on average every HOL packet only needs to wait for a finite amount of time after being placed in a mailbox. As such, every HOL packet departs from the mailbox switch within a finite average delay. In this case, the throughput is the same as the arrival rate (since the HOL blocking is not a constraint when $\delta = 50$). This is consistent with the throughput plot in Figure 4. We also note that the normalized average increment of the virtual waiting time is an increasing convex function in this stable segment. This is intuitive as the number of “collisions” that occur in finding an empty cells is increased at a much faster rate than the linear increase of the arrival rate.

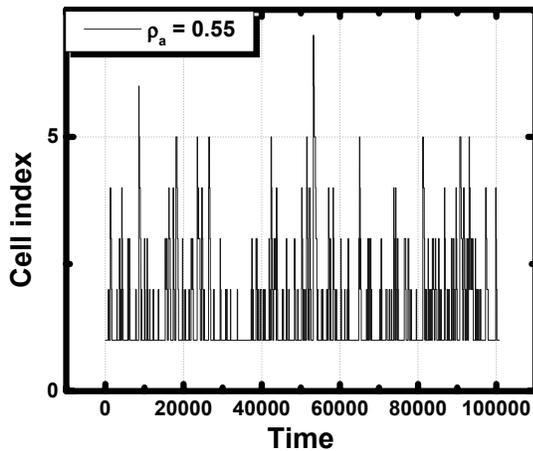


Fig. 6. A sample path of the cell index of the virtual waiting time of flow (1, 50) when arrival rate is 0.55

As the virtual waiting time is decreased by 1 per time slot, the virtual waiting time in the overloaded segment will go to ∞ eventually. In Figure 7, we plot a sample path of the cell index of the virtual waiting time for flow (1, 50) when $\rho_a = 0.69$. Note that the cell index of the virtual waiting time in this figure increases almost linearly to ∞ with respect to time. This is also consistent with the well known behavior of a unstable queue. As ρ_a exceeds the maximum throughput of the mailbox switch, it is intuitive to see that late arrivals of HOL packets have to wait much longer than early arrivals. As such, the virtual waiting time increases with respect to time.

In the third experiment, we consider the mailbox switch with limited numbers of forward and backward tries. As discussed in Section II-D, there are two parameters for such a mailbox switch: δ and δ_b . The search for an empty cell for flow (i, j) is started from the cell with the index $\max[1, f_{i,j}(t) - \delta_b]$ to the cell with

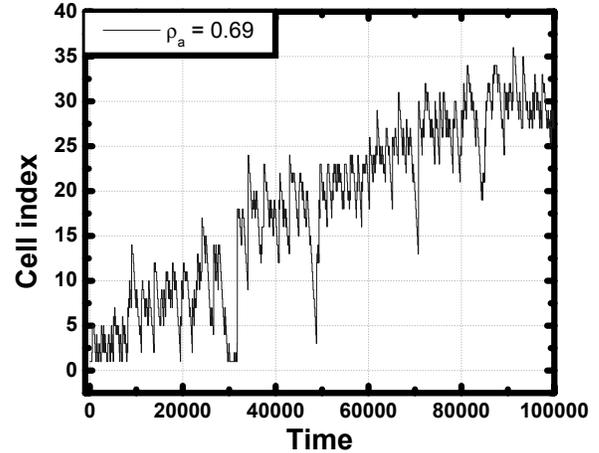


Fig. 7. A sample path of the cell index of the virtual waiting time of flow (1, 50) when the arrival rate is 0.69

the index $\min[F, f_{i,j}(t) + \delta]$. The resequencing delay for such a mailbox switch is bounded by $N\delta_b$ slots. In Figure 8 we plot the throughput as a function of δ_b for $\delta = 5, 6$, and 7. From Figure 8, we note that the mailbox switch can achieve more than 95% throughput with small δ and δ_b . The throughput is an increasing function of δ_b as placing a packet in a cell with the index smaller than the cell index of its virtual waiting time does not result in the increase of its virtual waiting time. Another interesting observation is that increasing δ does increase the throughput when δ_b is large. In the case that $\delta_b = 0$, we have known from Figure 3 that the throughput decreases as δ increases when $\delta \geq 4$. This is because a large δ tends to increase a large amount of the virtual waiting time when backward tries are not allowed ($\delta_b = 0$). However, this is not the case when δ_b is large. Even though a large δ tends to increase a large amount of the virtual waiting time, a large δ_b allows packets to be repacked in the cells that are “wasted” by a large increase of the virtual waiting time. Thus, the constraint is shifted from the stability of the virtual waiting time to the HOL blocking of FIFO queues. As a large δ tends to have a small probability of HOL blocking, this explains why the mailbox switch with a large δ has better throughput than that with a small δ when δ_b is large.

V. RECURSIVE CONSTRUCTION OF THE SYMMETRIC TDM SWITCHES

In order to construct an $N \times N$ mailbox switch, one needs to construct two $N \times N$ symmetric TDM switches. Even though an $N \times N$ symmetric TDM switch can be implemented by an $N \times N$ crossbar switch, we show in

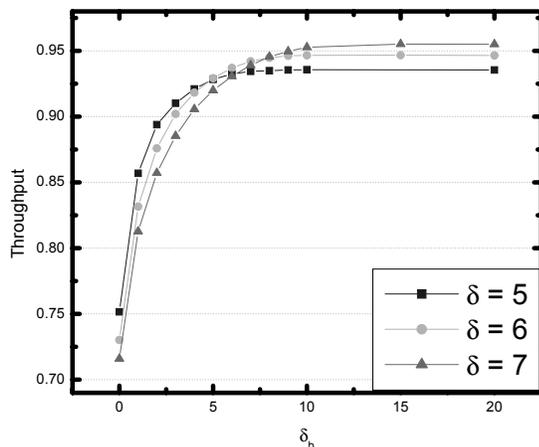


Fig. 8. The maximum throughput as a function of δ_b

the section that an $N \times N$ symmetric TDM switch can be easily constructed with $O(N \log N)$ complexity.

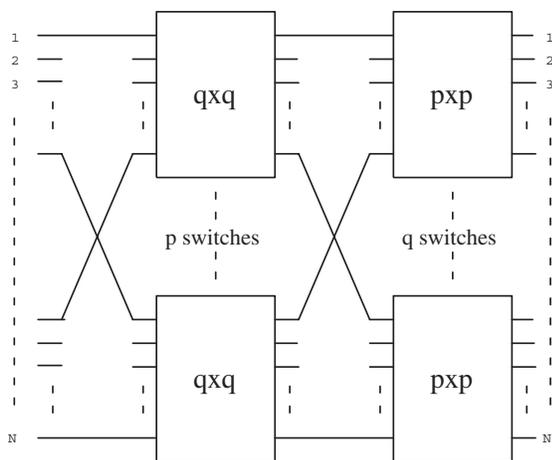


Fig. 9. A two-stage construction of an $N \times N$ symmetric TDM switch

In Figure 9, we show a two-stage construction of an $N \times N$ symmetric TDM switch (with $N = pq$). The first stage consists of p $q \times q$ symmetric TDM switches (indexed from $1, 2, \dots, p$) and the second stage consists of q $p \times p$ symmetric TDM switches (indexed from $1, 2, \dots, q$). These two stages of switches are connected by the perfect shuffle, i.e., the ℓ^{th} output of the k^{th} switch at the first stage is connected to the k^{th} input of the ℓ^{th} switch at the second stage. Also, index the N inputs and outputs from 1 to N . The N inputs of the $N \times N$ switch are connected to the inputs of the switches at the first stage by the perfect shuffle. To be precise, let

$$\ell(i) = \lfloor \frac{i-1}{p} \rfloor + 1, \quad (18)$$

and

$$k(i) = i - (\ell(i) - 1) * p. \quad (19)$$

Note that for $i = 1, 2, \dots, N$, $\ell(i)$ is an integer between 1 and q and $k(i)$ is an integer between 1 and p . Then the i^{th} input of the $N \times N$ switch is connected to the $\ell(i)^{th}$ input of the $k(i)^{th}$ switch at the first stage. Also, we note that the j^{th} output of the $N \times N$ switch is the $k(j)^{th}$ output of the $\ell(j)^{th}$ switch at the second stage.

The symmetric TDM switches at these two stages are operated at different time scales. The connection patterns of the symmetric TDM switches at the second stage are changed every time slot. However, the connection patterns of the symmetric TDM switches at the first stage are changed every *frame* with each frame containing p time slots. To be specific, we define the m^{th} frame of the k^{th} switch at the first stage in Figure 9 to be the set of time slots $\{(m-1)p+k, (m-1)p+k+1, \dots, mp+k-1\}$. Then every symmetric TDM switch at the first stage is operated according to its own frames. Note that the p symmetric TDM switches at the first stage do not change their connection patterns at the same time as the m^{th} frames of these switches contain different sets of time slots.

Lemma 1 *The two-stage construction in Figure 9 is an $N \times N$ symmetric TDM switch.*

Proof. In order for the $N \times N$ switch to be a symmetric TDM switch, we need to show that the i^{th} input port is connected to the j^{th} output at time t when

$$(i + j) \bmod N = (t + 1) \bmod N. \quad (20)$$

From the topology in Figure 9, we know there is a unique routing path from an input of the $N \times N$ switch to an output of the $N \times N$ switch. To be precise, the i^{th} input is connected to the $\ell(i)^{th}$ input of the $k(i)^{th}$ switch at the first stage. Also, the $\ell(j)^{th}$ output of the $k(i)^{th}$ switch at the first stage is connected to the $k(i)^{th}$ input of the $\ell(j)^{th}$ switch at the second stage. Note that the j^{th} output of the $N \times N$ switch is the $k(j)^{th}$ output of the $\ell(j)^{th}$ switch at the second stage. Thus, in order for the i^{th} input of the $N \times N$ switch to be connected to the j^{th} output of the $N \times N$ switch at time t , one must have

- (i) the $\ell(i)^{th}$ input of the $k(i)^{th}$ switch at the first stage is connected to its $\ell(j)^{th}$ output at time t , and
- (ii) the $k(i)^{th}$ input of the $\ell(j)^{th}$ switch at the second stage is connected to its $k(j)^{th}$ output at time t .

As the switches at the first stage are $q \times q$ symmetric TDM switches that change their connection patterns

every frame, we have from (i) that t must be in the m^{th} frame of the $k(i)^{\text{th}}$ switch at the first stage, where m satisfies

$$(\ell(i) + \ell(j)) \bmod q = (m + 1) \bmod q. \quad (21)$$

From (21), it follows that for some integer m_2

$$(\ell(i) - 1) + (\ell(j) - 1) = (m - 1) + m_2q. \quad (22)$$

Similarly, as the switches at the second stage are $p \times p$ symmetric TDM switches that change their connection patterns every time slot, we have from (ii) that

$$(k(i) + k(j)) \bmod p = (t + 1) \bmod p. \quad (23)$$

Since t is in the m^{th} frame of the $k(i)^{\text{th}}$ switch, t is one of the p time slots $\{(m - 1)p + k(i), (m - 1)p + k(i) + 1, \dots, mp + k(i) - 1\}$. Thus, we have from (23) that

$$t = (m - 1)p + k(i) + k(j) - 1. \quad (24)$$

Note from (19), (22) and (24) that

$$\begin{aligned} & (i + j) \bmod N \\ &= \left((\ell(i) - 1)p + k(i) + (\ell(j) - 1)p + k(j) \right) \bmod N \\ &= \left((m - 1)p + m_2pq + k(i) + k(j) \right) \bmod N \\ &= (t + 1 + m_2N) \bmod N \\ &= (t + 1) \bmod N. \end{aligned} \quad (25)$$

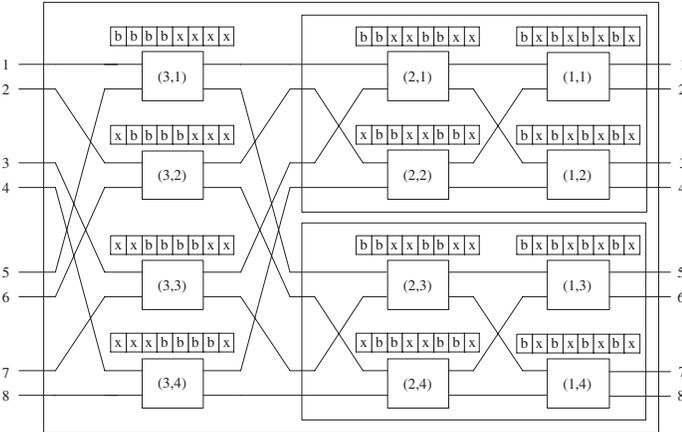


Fig. 10. An 8×8 symmetric TDM switch via 2×2 switches

Note that a 2×2 switch only has two connection patterns and it is a symmetric TDM switch if it alternates its two connection patterns every time slot. If N is a power of 2, then one can recursively expand the two-stage construction by 2×2 switches. The number of 2×2 switches needed for an $N \times N$ symmetric TDM switch is

then $\frac{N}{2} \log_2 N$. This shows that one can build an $N \times N$ symmetric TDM switch with $O(N \log N)$ complexity. In Figure 10, we show an 8×8 symmetric TDM switch that uses the recursive construction. The eight connection patterns of each 2×2 switch are represented by a sequence of 8 characters in “b” and “x”, where “b” denotes the bar connection and “x” denotes the cross connection of a 2×2 switch. To find out the connection patterns of the 2×2 switches in the general case, we index the stage from left to right by $1, 2, \dots, \log_2 N$, and index the switch in each stage from top to bottom by $1, 2, \dots, N/2$ as in Figure 10. Then the connection pattern of the m^{th} switch at the ℓ^{th} stage at time t is determined by the function $\psi(\ell, m, t)$:

$$\psi(\ell, m, t) = \left\lfloor \frac{(t - \phi(\ell, m)) \bmod 2^\ell}{2^{\ell-1}} \right\rfloor, \quad (26)$$

where

$$\phi(\ell, m) = ((m - 1) \bmod 2^{\ell-1}) + 1. \quad (27)$$

We set the bar connection pattern if $\psi(\ell, m, t) = 0$, and the cross connection pattern if $\psi(\ell, m, t) = 1$.

VI. CONCLUSIONS

In this paper, we proposed the mailbox switch as a scalable two-stage switch architecture for conflict resolution of ordered packets. The mailbox switch has the following nice features:

- (i) Low communication overhead: only the cell indexes need to be transmitted between input/output ports and mailboxes. This requires $\log_2(F + 1)$ bits for each placement of a HOL packet into a mailbox.
- (ii) Low computation overhead: one only needs to keep track of the cell index of the virtual waiting times $f_{i,j}(t)$ and the associated counter $g_{i,j}(t)$. The connection patterns of the two symmetric TDM switch fabrics are independent of the traffic.
- (iii) Low hardware implementation complexity: the symmetric TDM switches can be constructed recursively. An $N \times N$ symmetric TDM switch can be constructed with $\frac{N}{2} \log_2 N$ 2×2 switches.
- (iv) In order delivery: packets of the same flow are delivered in the order of their arrivals.
- (v) High throughput: more than 75% throughput can be achieved. When allowing limited resequencing delay, the mailbox switch can achieve 95% throughput.

Though not reported here, our simulations also show that the throughput can be higher if the traffic is bursty. The

intuition behind this is that packets of the same burst tend to be distributed *evenly* to the mailboxes and they are placed in the cells with the same index. As such, it is less likely to have a large increase of the virtual waiting time for a placement of a HOL packet.

In the following, we discuss possible extension of the mailbox switches.

- (i) In addition to returning the cell indexes of the virtual waiting times, one can also send out the information for the occupancy of the cells. With this additional information, one can implement VOQs at the inputs so that one can select packets from various VOQs to reduce the probability of HOL blocking. This corresponds to doing pipelined matching in a distributive manner. Our preliminary results show that even for the case $\delta = 0$, the throughput can be very close to 100% if the longest VOQ is selected. However, it is not clear how much information is needed for the occupancy of the cells in order to achieve high throughput.
- (ii) For the case with $\delta = \infty$, the throughput predicted by our theoretical model is too conservative. This is because the assumption in (A5) is over simplified. It would be of interest to refine the assumption to find the theoretical throughput.
- (iii) Even though the number of forward tries δ is fixed in our original design of the mailbox switch, it can be made to be adaptive to the FIFO queue at each input. It is intuitive that one should choose a large δ to avoid HOL blocking when the number of packets in the FIFO queue is large. On the other hand, one should reduce δ to minimize the increase of the virtual waiting time when the number of packets in the FIFO queue is small. The tradeoff is not clear.

REFERENCES

[1] T. Anderson, S. Owicki, J. Saxes and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. on Computer Systems*, Vol. 11, pp. 319-352, 1993.

[2] M. Andrews and M. Vojnovic, "Scheduling reserved traffic in input-queued switches: new delay bounds via probabilistic techniques," *Proceedings of IEEE INFOCOM*, 2003.

[3] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, Vol. 5, pp. 147-151, 1946.

[4] C.-S. Chang, W.-J. Chen and H.-Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," *IEEE IWQoS'99*, pp. 79-86, London, U.K., 1999.

[5] C.-S. Chang, W.-J. Chen and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *Proceedings of IEEE INFOCOM*, pp. 1614-1623, Tel Aviv, Israel, 2000.

[6] 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.

[7] C.-S. Chang, D.-S. Lee and C.-M. Lien, "Load balanced Birkhoff-von Neumann switch, part II: Multi-stage buffering," *Computer Communications*, Vol. 25, pp. 623-634, 2002.

[8] C.-S. Chang, D.-S. Lee, and C.-Y. Yue, "Providing guaranteed rate services in the load balanced Birkhoff-von Neumann switches," *Proceedings of IEEE INFOCOM*, 2003.

[9] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *Proceedings of IEEE INFOCOM*, pp. 556-564, Tel Aviv, Israel, March, 2000.

[10] A. Hung, G. Kesidis and N. McKeown, "ATM input-buffered switches with guaranteed-rate property," *Proc. IEEE ISCC'98*, Athens, pp. 331-335, 1998.

[11] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Trans. Commun.*, Vol. COM35, NO.12, Dec. 1987.

[12] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," *Proceedings of IEEE INFOCOM*, New York, 2002.

[13] I. Keslassy, M. Kodialam, T.V. Lakshman, D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches," *Proceedings of IEEE INFOCOM 2003*.

[14] T.-T. Lee and C.-H. Lam, "Path switching-a quasi-static routing scheme for large scale ATM packet switches," *IEEE Journal on Selected Areas of Communications*, Vol. 15, pp. 914-924, 1997.

[15] S. Li and N. Ansari, "Input-queued switching with QoS guarantees," *Proceedings of IEEE INFOCOM*, pp. 1152-1159, New York, 1999.

[16] Y. Li, S. Panwar and H.-J. Chao, "On the performance of a dual round-robin switch," *Proceedings of IEEE INFOCOM*, pp. 1688-1697, 2001.

[17] N. McKeown, "Scheduling algorithms for input-queued cell switches," *PhD Thesis. University of California at Berkeley*, 1995.

[18] N. McKeown, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Proceedings of IEEE INFOCOM*, pp. 296-302, 1996.

[19] A. Mekittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," *Proceedings of IEEE INFOCOM*, 1998.

[20] R. Nelson, "Stochastic catastrophe theory in computer performance modeling," *Journal of the Association for Computing Machinery*, Vol. 34, pp. 661-685, 1987.

[21] M. Schwartz, *Broadband Integrated Networks*. New Jersey: Prentice Hall, 1996.

[22] Y. Tamir and H.-C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, pp. 13-27, 1993.

[23] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, Vol. 2, pp. 5-12, Princeton University Press, Princeton, New Jersey, 1953.