Constructing an Optical Router with Minimum Complexity

Hadas Kogan, Isaac Keslassy
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
{rhadas@tx,isaac@ee}.technion.ac.il

***Abstract*—In the past years, electronic routers have had trouble keeping up with the increase in optical fiber capacity. As their power consumption has grown exponentially and already exceeds standards, it seems that an alternative solution is mandatory. Many have suggested all-optical routers as an alternative. However, these are deemed too complex, especially given the need to implement both switching and buffering, even though their fundamental complexity has apparently never been analyzed.**

In this paper, we study the number of fundamental optical components (2×2 switches and fiber delay lines) needed to emulate ideal routers. We first demonstrate that an $N \times N$ router with a buffer size of B per port needs at least $\Theta(N \log(NB))$ components, and then build a construction that achieves this lower bound. Finally, we generalize this result to different router architectures and scheduling disciplines.

I. Introduction

Router processing, switching and buffering are done today using electronics. Hence, the growth in router capacity is closely linked to Moore's law and to memory bandwidth growth, which are historically slower than the growth in optical link capacities [1], [2]. This is why routers have increasingly become a bottleneck for Internet communications. To keep up with incoming fiber traffic, network operators now need to piece together many routers per backbone node. In fact, it is not unusual to find more than ten routers in a single POP (Point Of Presence) [3]. Further, routers need to be packed more densely. In the past years, the power consumption per core router rack has increased exponentially, and now reaches more than 10kW, while standard accepted power consumption is about 2kW per rack [1], [4]. Clearly, this trend cannot continue indefinitely. There needs to be some future alternative.

A natural approach is to fully harness the power of optics by using an all-optical router. However, there are many obstacles on the road to optics, essentially due to the diverse functions needed in a router. In fact, a router can be seen as implementing three main stages. First, a *lookup stage* processes each arriving packet and determines its appropriate output link. Then, a *switching stage* transfers the packet from its input link to its output link. Finally, since different arriving packets might contend for the same output link, a *buffering stage* queues packets that lose in the contention. These three stages currently need millions of gates and extensive electronic memory in core routers, and therefore make all-optical core routers seem too complex and out of reach.

We can wonder whether there is any fundamental reason behind the complexity of these three stages. In fact, if we consider the lookup stage (including the header processing), it could be argued that its complexity is essentially due to protocol choices, and would therefore be much smaller in the future if network operators use simpler routing protocols such as MPLS [5], [6]. In this paper, we will analyze whether the complexity of the two other stages is protocol-related as well, or whether there exists some fundamental lower bound to this complexity.

There have been two main approaches to emulating electronic buffers using optics. The first approach is to slow – or stop – light [7]. However, this approach requires gas environments with tight temperature and pressure constraints, and currently seems impractical. The second approach, which we will follow, is to let optical packets circulate on switched fiber delay lines until their turn to depart [8]. An architecture implementing this approach only requires fiber delay lines with fixed propagation delays, and basic 2×2 switches to orient packets through them.

Using this approach, the switching and buffering stages of a router can be emulated with a set of simple 2×2 switches connected by fiber delay lines. The fundamental complexity of an all-optical router can then be provided by the minimum number of basic components (i.e., 2×2 switches and fiber delay lines) necessary to construct such a router. The goal of this paper is to analyze this fundamental complexity, and to provide constructions with as few basic components as possible.

Several works have made significant progress in the design of optical systems, with the switching as well as the buffering stages. On the switching side, Shannon showed in its seminal work that to realize all $N!$ possible permutations, an $N \times N$ unbuffered switch fundamentally needs at least $\Theta(\log(N!))$ basic components [9]. Subsequent research also focused on constructing minimum-complexity unbuffered switches [10]–[14] and time-space switches [15]–[17]. On the buffering side, there has been extensive research on FIFO multiplexers [18], FIFO queues [19], priority queues [20], time-slot interchanges [16], [21] and flexible delay lines [22].

In this paper, we focus on building an optimal OQ (Output-Queued) PIFO (Push-In-First-Out) router. An OQ router is a router in which arriving packets are placed immediately in the queue of their destination output. It is a work-conserving router, and therefore is often considered as the standard ideal router. Further, we allow the departure policy to be any PIFO policy, thus enabling the use of non-FIFO policies (such as WFQ, GPS and strict priority), so as to provide QoS (Quality-of-Service) guarantees.

Our main finding is that the fundamental complexity of an $N \times N$ OQ-PIFO router with a port buffer size B grows like $\Theta(N \log(NB))$. We further present a construction that achieves this fundamental complexity lower bound, and thus prove that it is achievable. Likewise, we show that it is also the fundamental complexity of a PIFO shared memory router, and that the same OQ-PIFO router construction can be used to emulate a PIFO shared memory router as well. We also present two non-optimal exact emulations of an OQ-FIFO switch.

Further, on the way to determine the fundamental complexity of an OQ-PIFO switch, we also prove an interesting result: the complexity of an optical buffer is $\Theta(\log(B))$. We present a construction that emulates such an optical buffer.

This paper is organized as follows: Section II presents definitions related to systems and complexity of systems. Section III reviews several optical constructions, which we later use. In section IV, a lower bound on the practical complexity of an OQ-PIFO switch is found. In order to emulate an OQ-PIFO switch, we introduce in section V an optical buffer. We present a construction of an optical buffer and show that it is optimal. Then, Section VI presents a construction emulating an optical OQ-PIFO switch. In section VII, the more general case of a PIFO shared memory is presented and emulated. Finally, Section VIII presents two exact emulations of an OQ-FIFO switch.

II. DEFINITIONS

A. Systems

We will refer to a *system* as an ideal network element that has input links, output links and inner states. The outputs of the system are uniquely determined as a function of the inputs and the inner states of the system during the entire time of operation. Packet size is fixed, time is slotted and it takes one time-slot to transmit a packet. If the packets have variable sizes, they are segmented into fixed size blocks during arrival and reassembled at departure. We distinguish systems *with memory*, i.e., where the packets on the output links may have arrived during previous time-slots, and systems that are *without memory*, i.e., where the packets on the output links have arrived on the current time-slot. Throughout this paper we consider optical constructions consisting only of fiber delay lines (FDLs) and optical 2×2 switches. The length of the fiber delay line determines its delay. For example, if a packet is transmitted through a FDL with length 5 at time t, it will depart from the other side at time $t + 5$. Finally, for simplicity, we consider that all FDLs use the same wavelength, even though we believe that these results could be extended to WDM fibers as well (when considering the action of wavelength selection as dual to switching in space).

Definition 1: An optical construction *emulates* an ideal system if, with identical arrivals, identical packets depart from the construction with some bounded delay compared to the packets departing from the ideal system [23]–[25]. If the delay is fixed we say that the construction *strongly emulates* the system. If there is no delay we say that the construction *exactly emulates* the system.

Definition 2: A *frame F(N,B,D)* is the collection of packets arriving on N links during B time-slots, where D is some constant time offset. For example, the k^{th} frame is composed of packets arriving on N links during times $t \in [(k-1)B + D + 1, kB + D]$.

We will denote a frame of packets arriving on the input links as an *input frame*, and a frame of packets departing on the output links as an *output frame*.

Definition 3: A system is *independent between frames* if the k^{th} output frame is composed only of the packets of the k^{th} input frame.

Definition 4: A *frames switch* [15], [17] with N inputs, N outputs, buffer size B and delay D is a network element that operates on frames such that the elements of the k^{th} output frame $F_O(N, B, d+D)$ are a permutation both in time and in space of the elements of the k^{th} input frame $F_I(N, B, d)$, for every k.

Figure 1 illustrates such a frames switch. Packets arrive on two input links. The frames switch performs

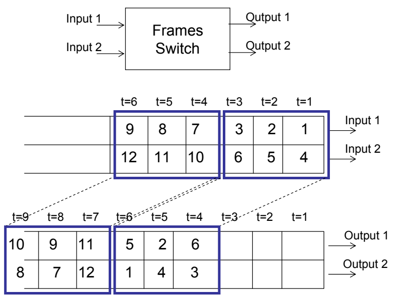


Fig. 1. A frames switch with $N = 2$, $B = 3$

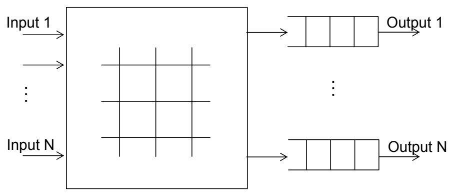


Fig. 2. An output-queued switch

a permutation along the time and space axis, where the frame size is $B = 3$ and $N = 2$. The permuted packets depart on the two output links. The operation of interchanging is performed after all the packets of the k^{th} input frame arrived to the system, and before the departure of the output frame starts. Therefore, we say that there is an inherent delay of B time-slots.

Example 1: A Time-Slot Interchange (TSI) ([11], [16]) is a network element with one input link and one output link that permutes the arriving packets in time. It is a special case of a frames switch with $N = 1$, as it operates only on one input link.

Example 2: An $N \times N$ switch has N input links and N output links, where the outputs are a permutation along the space of the inputs. It is a special case of a frames switch with $B = 1$.

Definition 5: An *Output-Queued (OQ) Switch* ([24], [25]) with N input links, N output links and buffer size B is a switch in which arriving packets are placed immediately in the queue of their destination output. An output buffer of size B can contain at most B packets destined to this output, and is forced to drop packets upon the arrival of additional packets.

An example of an OQ switch is shown in Figure 2. Packets arrive on N input links. They are switched into the buffers according to their destination, where they wait for their turn to depart. Of course, if $B \geq N$, each output buffer must be able to accept up to N packets at any time-slot, since all inputs might have a packet destined to this output.

Definition 6: A *departure policy* determines the order in which the packets depart from the queue. The departure policy may be first-in first-out (FIFO) or push-in first-out (PIFO) [24], we call it an OQ-FIFO switch or an OQ-PIFO switch respectively.

A PIFO departure policy is necessary for implementing a scheduling algorithm that schedules the packets to depart in an order different from their arrival order. As such, OQ-PIFO switch can be used to implement a variety of QoS scheduling disciplines such as WFQ, GPS and strict priorities.

B. Complexity

In this section, we will introduce notions of complexity and optimality of systems, expanding the original definitions by Shannon [9], [11], [15], [26].

Definition 7: The *practical complexity* [26] of a system is a measure of the minimal number of basic elements, i.e., 2×2 switches, necessary to compose the system. The practical complexity is given by

$$C^* = \lim_{t\to\infty} \lceil \frac{\log(\#states_t)}{t} \rceil,$$

where $states_t$ is the set of all the possible states of the system during time interval $[0, t]$.

Example 3: At each time-slot, an $N \times N$ switch can choose between $N!$ permutations. Therefore, the number of states of an $N \times N$ switch during interval $[0, T]$ is given by $(N!)^T$, because there are $N!$ possible states at the end of each of the T time-slots, and the system is independent between time-slots. Therefore, the practical complexity of the switch is

$$C^* = \lim_{T\to\infty} \frac{\log((N!)^T)}{T} = \log(N!),$$

which is exactly Shannon's result of the complexity of an $N \times N$ switch [9].

It is not always possible to emulate a system with a construction that achieves the practical complexity. For instance, consider the Example 3 above: even with this simple example, when $N \geq 3$, it has been argued that no construction can ever achieve the lower bound provided by the switch practical complexity [27]. Consequently, we will say that for a construction to be optimal, it suffices that its complexity does not grow faster with N than the practical complexity lower bound.

Definition 8: Denote by C the number of 2×2 switches composing a specific construction emulating a system S, and by C^* the practical complexity of system S. We say that a construction is *optimal* if $C = \Theta(C^*)$, i.e., there is some constant k such that $C^* \leq C \leq kC^*$.

Example 4: A Benes network has $N \log N$ 2×2 switches. According to the Stirling formula,

$$C = N \log N = \Theta(\log(N!)) = \Theta(C^*)$$

Therefore the Benes network is said to be optimal.

Note again that the number of 2×2 switches does not have to be identical to the theoretical minimum in order to be called optimal. In this last example, a Benes network has $N \log N$ switches, which is more than the practical complexity $\log(N!)$ for $N \geq 3$. Nevertheless, it will be said to be optimal.

Our goal is to build an optimal construction of an OQ-PIFO switch. We will first find the practical complexity of an OQ-PIFO switch, and then we will build a construction that achieves the practical complexity lower bound.

III. Previous constructions

In this section, we review several optical constructions, which have already been introduced in the literature and will be used throughout this paper.

- *Memory cell:* An optical memory cell is capable of receiving and storing one packet. When a read request arrives the stored packet departs on the output link. Reference [18] shows an implementation of an optical memory cell with a 2×2 switch and a fiber delay line (FDL) with length 1. The state of the switch determines if it is in a "read"/"write" state or in a "store" state.
- *Priority queue:* An optical priority queue of size B is a network element with one input link, one output link and one lost link. When a departure request arrives, the packet with the highest priority is sent on the output link. When there is an arrival and the queue is full, the packet with the lowest priority among the arriving packet and the packets stored in the buffer is sent on the lost link. References [19], [20] prove that a construction containing a switch of size $\sqrt{B} \times \sqrt{B}$ and $\sqrt{B}$ feedback delay lines suffices to exactly emulate a priority queue.
- *FIFO multiplexer:* An optical FIFO multiplexer with buffer B is a network element that has N input links, one output link and $N-1$ lost links. The multiplexer is capable of receiving up to N packets in one time-slot and store them in the buffer. If the buffer is full, arriving packets are lost. The multiplexer is non-idling: whenever there is at least one packet stored in the buffer there is a departure according to FIFO policy. Reference [18] suggests a recursive construction of a multiplexer $N \rightarrow 1$ with buffer B using $\log_N(B)$ $N \times N$ switches.
- *FIFO queues:* An optical FIFO queue has one input link, one output link and one control link. If there are packets stored in the buffer and the control link is enabled packets can depart from the queue. Reference [19] presents a recursive construction of an optical FIFO queue with $2\log_N(B+1)$ 2×2 switches, as well as a construction of a $2 \rightarrow 1$ FIFO queue.
- *A time-slot interchange:* A time-slot interchange (TSI) [11] is a network element with one input link and one output link, that is capable of interchanging the positions of B arriving time-slots according to any permutation. Throughout this paper a time-slot interchange with frame size B will be denoted as π_B. [16] presents optical implementations of a TSI, with $2\log B - 1$ 2×2 switches.

IV. A lower bound on the practical complexity of an OQ-PIFO switch

Our goal is to find the practical complexity of an OQ-PIFO switch, and then to build an optimal construction with a number of 2×2 switches that grows like its practical complexity. However, it appears that finding the practical complexity of an OQ-PIFO switch is far from straightforward. At each time-slot, its output state depends on the previous time-slots — contrarily to systems that are independent between frames or time-slots, such as TSIs and unbuffered switches. Therefore, it is not possible to divide the time into frames and to count the number of different states within one frame. Thus, a straightforward calculation of the practical complexity will require the identification of all the possible groups of states during the entire time of operation. However, building such a state space seems extremely hard, as there are many parameters that we should consider, such as the relative priorities, arrival ordering and lost packets in cases where some queues are full.

In the following, we will introduce an alternative way to find the practical complexity of an OQ-PIFO switch, without any need to find the whole state space. First, we will introduce a lower bound on the practical complexity. Then we will find a construction that achieves that lower bound, proving that the practical complexity is equal to the lower bound.

The lower bound is introduced by showing that a frames switch is a special case of an OQ-PIFO switch. Therefore the practical complexity of an OQ-PIFO switch is no less than the practical complexity of a frames switch, which constitutes a lower bound on the practical complexity of an OQ-PIFO switch[1]. (For the

[1]Of course, a frames switch cannot serve as an OQ-PIFO switch. Consider for example the case of a packet that is stored in one of the output queues, and continuously suffers from arriving packets with higher priority that keep coming and pushing it backwards. In an ideal OQ-PIFO switch, that packet will suffer from an unbounded delay, which is impossible using a frames switch.

sake of presentation, all proofs in this paper are presented in the Appendix.)

Lemma 1: Denote by C_P^* the practical complexity of an output-queued PIFO switch with N inputs, N outputs and buffer size B, and by C_F^* the practical complexity of a frames switch with N inputs, N outputs and frames size B. It holds that $C_P^* \geq C_F^*$.

The practical complexity of a frames switch is the logarithm of the number of permutations in space and time, divided by the time it takes to perform a permutation [15], [17]. A fixed delay does not change the practical complexity, as the delay divided by time goes to zero as the time goes to infinity.

Lemma 2: The practical complexity of a frames switch with delay D is given by $C_F^* = \Theta(N \log(NB))$.

Example 5: A Time-Slot Interchange (TSI) [11], [16] is a special case of a frames switch with $N = 1$. The practical complexity of a TSI is $\Theta(\log B)$ as it performs one of $B!$ permutations during B time-slots.

Example 6: An $N \times N$ switch is a special case of a frames switch with $B = 1$. The practical complexity of an $N \times N$ switch is $\Theta(N \log N)$ as it performs one of $N!$ permutations during one time-slot.

By combining Lemma 1 and Lemma 2 we get a lower bound on the practical complexity of an OQ-PIFO switch.

Theorem 1: Denote by C_P^* the practical complexity of an output-queued PIFO switch with N inputs, N outputs, buffer size B and delay D, it holds that $C_P^* = \Omega(N \log(NB))$.

These results can be intuitively explained as follows: An OQ-PIFO switch contains at least one $N \times N$ switch, which routes the arriving packets to the appropriate output queue, and N PIFO output queues of size B that store the packets. The practical complexity of the switch is $\Theta(N \log N)$; and the practical complexity of each of the N PIFO queues is no less than the practical complexity of a TSI, $\Theta(\log B)$. Therefore, the practical complexity of the OQ-PIFO switch will be no less than the sum of practical complexities, $\Theta(N \log N + N \cdot \log B)$, which is exactly the derived lower bound.

Now, we want to find a construction that has a number of switches that equals the lower bound derived in this section. If we find such a construction, it will prove that the practical complexity of an OQ-PIFO switch equals the lower bound. In order to do that, we will first introduce in the following section a new construction of an optical buffer.

V. Optimal emulation of an optical buffer

In this section, we will define and emulate an optical buffer. We will use our construction of an optical buffer in constructing an optimal emulation of an OQ-PIFO switch.

A. Definition of an optical buffer

A buffer in electronics is an element that receives packets and stores them. When a departure request for a specific packet arrives, the buffer sends the desired packet on the departure link. We will now define and implement an optical buffer with a similar functionality to that of an electronic buffer.

Definition 9: An *optical buffer B* is a network element with one input link and one output link that can store up to B packets. When the buffer receives a departure request for a certain packet, the packet is sent on the departure link.

In this general definition, it is interesting to note that the relative departure order of packets that are stored in an optical buffer can arbitrarily change *after* the packets arrive to the buffer, unlike other buffering structures presented earlier such as priority queues and FIFO queues.

In addition, from a more practical standpoint, it is worth noting that practical electronic buffers need a delay of several clock cycles between the arrival of a read request and the read operation itself. Similarly, in our constructions of optical elements emulating electronic buffers, we will allow for some bounded delay between the arrival of a departure request and the departure of the desired packet. This bounded delay is included in the emulation definition (Definition 1).

B. The practical complexity of an optical buffer

We will present here two constructions emulating an optical buffer, both with $O(\log B)$ 2×2 switches. We claim that those constructions are optimal. In order to prove that, we will use arguments similar to those used with an OQ-PIFO switch.

We will first show that a lower bound on the practical complexity of an optical buffer is given by $C_B^* = \Omega(\log B)$. A lower bound on the practical complexity of an optical buffer is derived by showing that a TSI is a special case of an optical buffer. Therefore, the practical complexity of an optical buffer is not lower than the practical complexity of a TSI. Note that a bounded emulation delay does not change the practical complexity.

Theorem 2: Denote by C_B^* the practical complexity of an optical buffer with capacity B, it holds that $C_B^* = \Omega(\log(B))$.

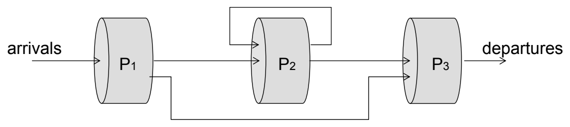


Fig. 3. Illustration of the optical buffer emulation principle

Next, we will present two constructions of an optical buffer, both with $O(\log B)$ switches. Since the constructions have a number of switches that grows like the lower bound, the practical complexity equals the lower bound

$$C_B^* = \Theta(\log(B)),$$

and the constructions will be said to be optimal.

C. Construction overview

The idea of the construction is demonstrated in Figure 3. Consider three buckets P_1, P_2 and P_3, where P_1 holds the packets that arrived during the last time frame, P_2 holds the packets that were stored in the buffer at the beginning of the last time frame, and P_3 holds the departing packets. A central controller divides the packets stored in P_1 and P_2 into two groups: packets that should remain in the buffer, and packets that should depart from the buffer. The packets that should remain in the buffer are written into P_2 and the packets that should depart from the buffer are written into P_3. The packets that are stored in P_3 are rearranged before departure according to the desired departure ordering.

D. Construction I

One straightforward emulation of optical buffer following the above overview is presented in Figure 4. The construction is composed of an FDL of length B storing the packets that arrived during the last B time-slots (P_1). A FIFO queue with 2 inputs stores the packets that are in the buffer (P_2). A FIFO multiplexer with 2 inputs followed by a time-slot interchange hold and rearrange the departing packets according to the desired ordering before departure (P_3). There is a central controller that controls the 2×2 switches. The central controller is aware of all the packets that are stored in the construction and their priorities, and receives the departure requests arriving from an external source. The central controller decides for each packet if it should remain in the buffer or depart. (For the sake of presentation, all proofs in this paper are presented in the Appendix.)

Theorem 3: The construction presented in Figure 4 emulates an optical buffer with delay $D = 3B$.

P_2 might need to store more than B packets: although the number of stored packets is at most B at the start

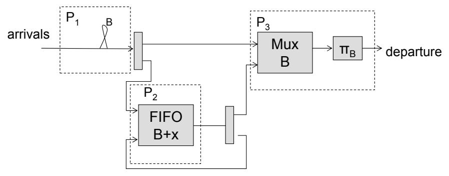


Fig. 4. Emulating an optical buffer - construction I

and end of each time interval $[kB+1, (k+1)B]$, it might exceed B inside the time interval. The following lemma specifies what the size of P_2 should be.

Lemma 3: The capacity of P_2 in Figure 4 should be at least $\frac{3B}{2}$ (i.e., $x = \frac{B}{2}$) in order to maintain the correct behavior of the construction.

E. Construction II

A second construction of an optical buffer is presented in Figure 5. It includes one FDL of length B that delays the arriving packets (P_1). The FDL is followed by a TSI of size B that rearranges the arriving packets before switching the packets either into the buffer or to the departure line. The TSI is followed by a memory cell with capacity B that holds the stored packets (P_2), and a second TSI of size B that rearranges the departing packets before sending them on the departure link (P_3). Stages A-E demonstrate the way the construction works. Consider the dark packets as the packets that should depart in the next time frame and the white packets as the packets that should remain in the buffer. The numbers displayed on the dark packets stand for the desired departure ordering. The first TSI rearranges the frame of arriving packets (A) to create frame B. The packets in frame B are arranged such that no collision occurs: each dark packet in B arrives to the 2×2 switch simultaneously with a white packet that is stored in the buffer (C), similarly, white packets in B arrive simultaneously with dark packets in C. As a consequence, the packets that depart from the central switch (D) are all dark and the packets that remain in the buffer are all white. But the departing packets are still not organized by the desired departure ordering, therefore the second TSI arranges the packets such that their ordering is exactly the desired departure ordering (E).
This construction is a strong emulation of an optical buffer, i.e., it has a fixed delay that is composed of the sum of delays of the two TSIs.

Theorem 4: The construction presented in Figure 5 strongly emulates an optical buffer, with a fixed delay $D = 3B$.

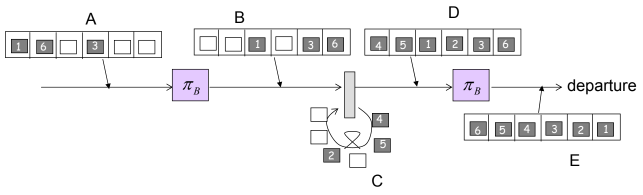


Fig. 5. Emulating an optical buffer - construction II

F. Proof of optimality

Both constructions include elements such as FIFO queues, FIFO multiplexers and time-slot interchanges. In [19] a summary of the complexities of all these elements can be found. All these elements can be implemented with $O(\log B)$ 2×2 switches. Therefore, $O(\log B)$ is an upper bound on the practical complexity of an optical buffer. Moreover, we have shown earlier that a lower bound on the practical complexity of an optical buffer is given by $C_B^* = \Omega(\log B)$. The combination of the lower and upper bounds results in a practical complexity of $\Omega(\log B)$. (Note that a fixed additional delay does not change the practical complexity in the definition, since the delay divided by time goes to zero as time goes to infinity.)

Theorem 5: The practical complexity of an optical buffer is $C_B^* = \Theta(\log(B))$.

Since the two constructions of an optical buffer presented earlier have $C = \Theta(C_B^*)$ 2×2 switches, they are said to be optimal according to Definition 8.

Corollary 1: The two constructions of an optical buffer presented in Figures 4 and 5 are optimal.

To conclude, we have presented two optimal constructions emulating an optical buffer, both with delay $D = 3B$. The second construction is a strong emulation, i.e., it has a fixed delay. Next, we will build an optical OQ-PIFO switch that uses our second construction of an optical buffer.

VI. Optimal complexity OQ-PIFO switch

We have found a lower bound on the practical complexity of an OQ-PIFO switch $C_P^* = \Omega(N \log(NB))$. In this section we will find the practical complexity of an OQ-PIFO switch and present an optimal construction. First, we present two naive constructions that exactly emulate an OQ-PIFO switch. These naive constructions will introduce a loose upper bound on the practical complexity of an OQ-PIFO switch. Then, we will present a construction emulating an OQ-PIFO switch, using the pigeonhole principle and the optical buffer constructions presented above. We will show that the construction of an OQ-PIFO switch by the pigeonhole principle is optimal.

A. Naive constructions of an output-queued PIFO switch

We describe here two naive constructions that exactly emulate an output-queued PIFO switch with output queue size B. The first construction implements separately each of the NB memory cells of the OQ-PIFO switch, while the second construction implements separately N^2 buffers corresponding to the N^2 (input,output) pairs. We will see that both naive constructions have a high complexity because they do not fully use the output buffer multiplexing properties.

The first construction is presented in Figure 6(a). Packets arrive on N input links. Then, they are sent to empty memory cells [18] through an $N \times NB$ switch. Finally, when packets should depart, they are read from the memory cells, and an $NB \times N$ switch sends them on the corresponding output link. There are NB parallel memory cells, as there should be up to B packets stored for each buffer.

Lemma 4: The construction in Figure 6(a) is a naive emulation of an OQ-PIFO switch, with $O(NB\log(NB))$ 2×2 switches

The second construction is presented in Figure 6(b). Packets arrive on N input links. Then, they are sent to priority queues [20] through an $N \times N^2$ switch. There are N^2 priority queues, one for each (input,output) pair. Finally, when a packet should depart, it is read from the corresponding priority queue, through an $N^2 \times N$ switch that sends it on the correct output link.

Lemma 5: The construction in Figure 6(b) is a naive emulation of an OQ-PIFO switch, with $O(N^2\sqrt{B}\log(B))$ 2×2 switches.

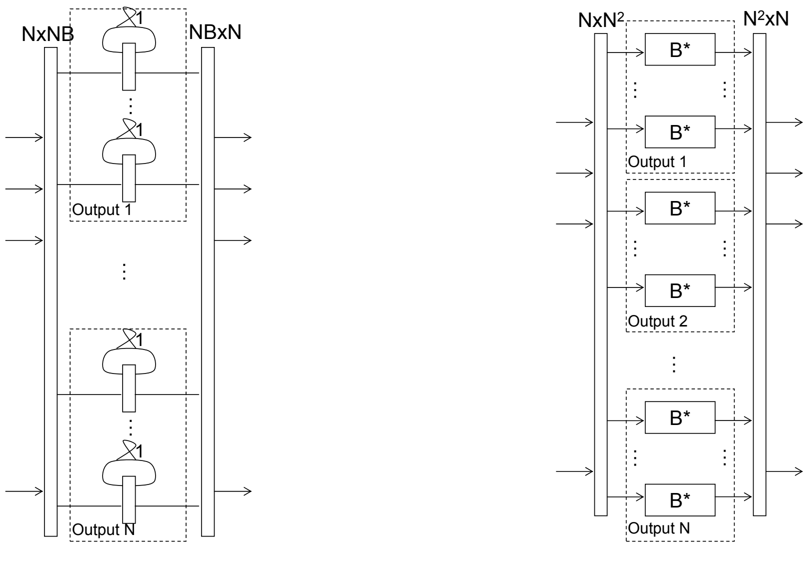
The naive constructions present an upper bound on the practical complexity.

Lemma 6: An upper bound on the practical complexity of an OQ-PIFO switch is given by $C_P^* = O(\min\{N^2\sqrt{B}\log(B), NB\log(NB)\})$

B. Emulating an OQ-PIFO switch by the pigeonhole principle

As shown in [28], it is possible to use the pigeonhole principle to emulate an OQ-PIFO switch. Pigeons represent packets, pigeonholes represent buffers, and packets follow a set of rules called the *Constraint Set*.

The basic rule is that only one packet (pigeon) can enter or depart a buffer (pigeonhole) at each time. Therefore, a packet scheduled to leave at time D cannot enter a buffer in which there is already a packet about to depart at time D. It introduces a certain difficulty on the straightforward emulation of a PIFO discipline: since the departure time is not known in advance with a PIFO



(a) Construction I (b) Construction II

Fig. 6. Naive constructions of an OQ-PIFO switch

discipline, it is not possible to prevent conflicts in the departure process.

A possible solution presented in [28] is to modify the departure process to prevent memory conflicts amongst packets destined to different outputs. Instead of sending only one packet to each output every time-slot, choose one output every time-slot and send several packets to that output. The outputs are chosen in a round robin fashion. More formally, consider an OQ-PIFO switch with N ports. Assume that $O_k(t)$ is the packet departing on output link k at time t, and $P_l(t)$ is the packet stored in the buffer that should depart to destination l at time t, $1 \leq k, l \leq N$. With the original departure process,

$$O_k(t) = P_k(t), \forall 1 \leq k \leq N,$$

while with the modified departure process,

$$O_k(t) = P_{t \bmod N}(\lfloor \frac{t}{N} \rfloor N + k), \forall 1 \leq k \leq N.$$

With the modified departure process it is possible to implement an OQ-PIFO switch according to the pigeonhole principle. Now, every packet can potentially collide only with the $N-1$ previous packets and the $N-1$ consequent packets scheduled to the same output, and not with packets scheduled to other output links. In order to avoid these collides during departure, when a packet arrives to the buffer, it cannot enter the buffers in which the $N-1$ previous packets or the $N-1$ consequent packets are stored.

There are no conditions on the packets that are stored in the buffers. Their departure time is not known in advance, it is not even bounded. Their relative departure ordering does not have to remain fixed once they are already stored in the buffers. The optical buffer we have presented earlier is flexible enough to match these requirements. Therefore, we will use our construction of an optical buffer in an emulation of an OQ-PIFO switch. We will find the practical complexity of an OQ-PIFO switch and show that an emulation of an OQ-PIFO switch by the pigeonhole principle with our emulation of optical buffers is optimal.

C. Optimal emulation of an OQ-PIFO switch

Finally, we present a construction that emulates the behavior of an OQ-PIFO switch with $\Theta(N \log(NB))$ switches. The construction will give us a tight upper bound on the practical complexity, and we will show that it is optimal. The construction uses the pigeonhole principle and it includes parallel optical buffers. We will use the construction presented in Figure 5 as an emulation of an optical buffer.

There are two main differences between the behavior of the optical buffer presented here and the electronic buffers in [28]. The first difference is that the packets arrive to the optical buffer and depart from it on different links, therefore read and write operations can be made simultaneously. The second difference is that for an optical buffer there is a storage limitation that presents an additional constraint, whereas for electronic buffers the storage capacity is huge and there is no need to take it into account. The constraint set is therefore slightly

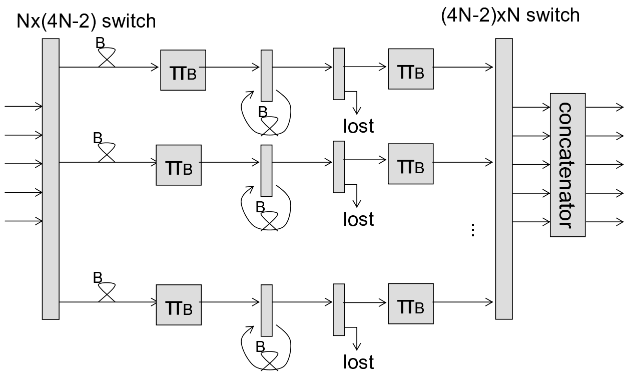


Fig. 7. An emulation of an OQ-PIFO switch

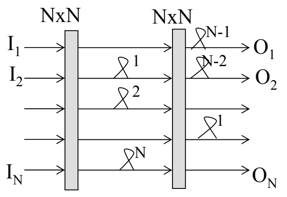


Fig. 8. A concatenator

different from the constraint set introduced in [28] for an OQ-PIFO switch. The first difference relaxes N constraints, and the second difference presents additional N constraints.

A construction of an optical OQ-PIFO switch is presented in Figure 7. Arriving packets are sent through an $N \times (4N-2)$ switch to optical buffers. The optical buffers are implemented as in Figure 5. Departing packets are read from the corresponding optical buffers through an $(4N-2) \times N$ switch and are sent on the output links. If there are already B packets with the same destination that are stored in the buffer, the packets with the lowest priority with that destination are lost through the lost links. The packets depart from the optical buffers according to the modified departure process described in section VI-B, through a concatenator that rearranges them according to the original departure process. The concatenator is presented in Figure 8.

(For the sake of presentation, all proofs in this paper are presented in the Appendix.)

Theorem 6: The construction presented in Figure 7 is a strong emulation of an optical OQ-PIFO switch with $O(N\log(NB))$ 2×2 switches and delay $D = 3B + N$.

The delay of the construction is the sum of the delay resulting from the modified departure process and the delay of the optical buffer. The practical complexity is a lower bound on the number of 2×2 switches necessary to build a construction. Therefore, the number of 2×2 switches in our construction of an OQ-PIFO switch is an upper bound on the practical complexity of an OQ-PIFO switch. We will now find the practical complexity of an OQ-PIFO switch, by combining the lower bound derived in Theorem 1, and the upper bound introduced by the number of 2×2 switches in the construction in Figure 7.

Theorem 7: The practical complexity of an OQ-PIFO switch is $C_P^* = \Theta(N\log(NB))$.

Since the construction of an OQ-PIFO switch in Figure 7 has $C = \Theta(C_P^*)$ 2×2 switches, it is said to be optimal according to Definition 8, as stated in the following corollary.

Corollary 2: The construction of an OQ-PIFO switch presented above is optimal.

To conclude, we have found the practical complexity of an OQ-PIFO switch, and presented an optimal construction emulating it.

VII. Optimal complexity shared memory

Until now we have discussed the emulation of an OQ-PIFO switch with N output queues, each with capacity B. Now we would like to expand our discussion also to the case of a router with shared memory. We consider the case of an optical shared memory with capacity NB. Arriving packets are written to the shared memory if it is not full, and wait there for their turn to departure. The difference between this shared memory architecture and an OQ architecture is that there is no further limitation on the number of packets destined to the same output link that can be stored in the shared memory. However, it is interesting to note that our construction of an OQ-PIFO switch also does not impose a limitation on the maximum number of packets with the same destination that are stored in the buffer. Therefore, it is also possible to emulate a shared memory with the construction presented in Figure 7.

Theorem 8: The construction presented in Figure 7 is an emulation of a shared memory with PIFO departure policy.

An OQ-PIFO switch with N output buffers, each with capacity B, can be emulated using a shared memory with a PIFO departure policy and capacity NB. Therefore, the practical complexity C_S^* of the shared memory is lower bounded by the practical complexity of an OQ-PIFO switch: $C_S^* = \Omega(C_P^*)$. Further, since we have shown that the same optimal construction of an OQ-PIFO switch can be used to implement a shared memory as well, we also get an upper bound on the practical complexity of a shared memory.

Corollary 3: The practical complexity of a shared memory with a PIFO departure policy is $C_S^* = \Theta(N\log(NB))$.

Corollary 4: The construction of a a shared memory with a PIFO departure policy presented above is optimal.

VIII. Exact emulation of an OQ-FIFO switch

In section VI we have shown three different emulations of an OQ-PIFO switch. The first two were naive constructions, exactly emulating an OQ-PIFO switch, but with high complexity. The third was an optimal emulation, with a number of switches that equals in growth to the practical complexity, but with a certain delay. Note that an OQ-FIFO switch is a special case of an OQ-PIFO switch, therefore all three constructions are valid also for the case of an OQ-FIFO switch. In addition, the practical complexity of an OQ-FIFO switch is not higher than the practical complexity of an OQ-PIFO switch.

The case of an OQ switch with a FIFO discipline seems easier to emulate than the case of an OQ switch with a PIFO discipline. In this section we present two constructions based on the pigeonhole principle, exactly emulating an OQ-FIFO switch, both with lower complexity than the naive constructions presented in section VI. The constructions that we present exactly emulate emulate a non-idling OQ-FIFO switch: whenever there are packets destined to some output stored in the buffer, there is a departure to that output.

By the pigeonhole principle, it is possible to construct an output-queued FIFO switch that is non-idling (i.e., $c_i(t) = 1$ for all t and for $1 \leq i \leq N$) with $2N-1$ parallel buffers [28]. Note that now the relative ordering of the packets stored in the buffers does not change once written to the buffer. Instead of using our optical buffer emulation, which has low complexity but non-zero delay bound, it is possible to use one of the following structures, so as to *exactly* emulate an OQ-FIFO router with no additional delay. In each case, we use $2N-1$ parallel buffers, each time implementing the buffers using a different structure.

- *Using priority queues*: The construction of a non-idling OQ-FIFO switch using $2N-1$ parallel priority queues has
$$C = O(N \log N + N\sqrt{B}\log B)$$
2×2 switches.
- *Using flexible delay lines*: A flexible delay line is a network element that can realize different mappings in time without delay. It can be emulated using Cantor networks with $O(\log^2(B))$ 2×2 switches [22]. Thus, we could use a construction that uses $2N-1$ parallel flexible delay lines, as was also independently proposed in [29]. The number of 2×2 switches in such a construction is
$$C = O(N \log N + N \log^2(B))$$

Both these constructions are *not optimal*, because their complexities are higher than the practical complexity of an OQ-FIFO switch, and they also cannot emulate any OQ-PIFO switch. Nevertheless, as these constructions are without delay, they can be a good choice when the implementation of a non-idling FIFO switch is needed and delay is critical.

IX. Conclusion

This paper dealt with the fundamental number of components required to build all-optical routers. We showed that an $N \times N$ optical router with a port buffer size B needs at least $\Theta(N \log(NB))$ components. Further, we presented a construction that achieves this fundamental lower bound, and thus proved that it is achievable.

This paper has two main consequences. First, it proves that the fundamental number of components of optical routers cannot be arbitrarily low. Therefore, advances in routing protocols cannot reduce the number of components to arbitrarily low levels. This extends the observation already made by Shannon for unbuffered switches over fifty years ago [9]. On the other hand, this paper also quantifies the effect of reducing the buffer size on the number of components, and thus sheds a ray of light on the possibility of implementing basic optical routers with small buffers. In fact, it shows how the recent papers on reducing the buffer size B in routers have a dramatic influence on the expected fundamental complexity of all-optical routers [30], [31].

Acknowledgments

The authors would like to thank Ami Litman and C.-S. Chang for their helpful comments, and to acknowledge the support of the ATS-WD Career Development Chair.

References

[1] N. McKeown, *Internet routers: past, present and future,* BCS Ada Lovelace Lecture, London, 2006.

[2] I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard and N. McKeown, *Scaling Internet routers using optics,* ACM SIGCOMM, Karlsruhe, Germany, 2003.

[3] N. Spring, R. Mahajan, D. Wetherall and T. Anderson, *Measuring ISP topologies with rocketfuel,* IEEE/ACM Transactions on Networking, Vol. 12, No. 1, pp. 2–16, 2004.

[4] C. Minkenberg, R.P. Luijten, F. Abel, W. Denzel and M. Gusat, *Current issues in packet switch design,* Computer Communication Review, Vol. 33, No. 1, pp. 119–124, 2003.

[5] W. Wang, L. Rau and D. J. Blumenthal, *All-optical label switching/swapping of 160 Gbps variable length packets and 10 Gbps labels using a WDM Raman enhanced-XPM fiber wavelength converter with unicast/multicast operation,* Optical Fiber Communications Conference and Exhibit (OFC), postdeadline paper PDP8, 2004.

[6] R. Takahashi, T. Nakahara, H. Takenouchi and H. Suzuki, *40-Gbit/s label recognition and* 1×4 *self-routing using self-serial-to-parallel conversion,* IEEE Photonics Technology Letters, Vol. 16, pp. 692–694, 2004.

[7] L. V. Hau, S. E. Harris, Z. Dutton and C. H. Behroozi, *Light speed reduction to 17 metres per second in an ultracold atomic gas,* Nature, Vol. 397, pp. 594–597, 1999.

[8] E. F. Burmeister and J.E. Bowers, *Integrated gate matrix switch for optical packet buffering,* Photonics Technology Letters, Vol. 18, No. 1, 2006.

[9] C. E. Shannon, *Memory requirements in a telephone exchange*, Bell System Technical Journal, Vol. 29, pp. 343–349, 1950.

[10] C. Clos, *A study of non-blocking switching networks*, Bell System Technical Journal, Vol. 32, no. 2, pp. 406–424, 1953.

[11] M. J. Marcus, *New approuches to the analysis of connecting and sorting networks*, Technical Report, 1972.

[12] V. E. Benes, *Mathematical theory of connecting networks and telephone traffic*, New York: Academic Press, 1965.

[13] J. Hui, *Switching and traffic theory for integrated broadband networks,* Boston: Kluwer Academic Publishers, 1990.

[14] N. Pippenger, *The complexity theory of switching networks*, Technical Report, 1973.

[15] D. K. Hunter and D. G. Smith, *New architectures for optical TDM switching*, IEEE J. Lightwave Technology, Vol. 11, pp. 459–511, 1993.

[16] F. Jordan, D. Lee, K.Y. Lee and S.V. Ramanan, *Serial array time slot interchangers and optical implementations*, IEEE Transactions on Computers ,Vol. 43, No. 11, pp. 1309–1318, 1994.

[17] R. Kannan, D. Lee, K.Y. Lee and H.F. Jordan, *Optical TDM sorting networks for high-speed switching*, IEEE Transactions on Communications, Vol. 45, Issue 6, pp. 723–736, 1997.

[18] C.S. Chang, D.S. Lee and C.K. Tu, *Recursive construction of FIFO optical multiplexers with switched delay lines*, IEEE Transactions on Information Theory, Vol. 50, pp. 3221–3233, 2004.

[19] C. S. Chang, Y. T. Chen and D. S. Lee, *Construction of optical FIFO queues and non-overtaking delay lines*, Technical Report, 2005.

[20] A. D. Sarwate and V. Anantharam, *Exact emulation of a priority queue with a switch and delay lines*, submitted to Queueing Systems Theory and Applications.

[21] C. S. Chang, J. Cheng and D. S. Lee, *A simple proof for the constructions of optical priority queues*, submitted to Queueing Systems, 2005.

[22] C. S. Chang, Y.T Chen, J. Cheng and D.S. Lee, *Constructions of linear compressors, non-overtaking delay lines, and flexible delay lines for optical packet switching*, submitted to IEEE/ACM Transactions on Networking, 2006.

[23] S. Chuang, A. Goel, N. McKeown and B. Prabhakar, *Matching output queueing with a combined input/output-queued switch*, IEEE Sel. Areas in Communications, Vol. 17, no. 6, pp. 1030–1039, 1999.

[24] S. Iyer and N. McKeown, *Techniques for fast shared memory switches*, HPNG Technical Report – TR01–HPNG–081501, Stanford, CA, Aug 2001.

[25] H. Attiya, D. Hay and I. Keslassy, *Packet-mode emulation of output-queued switches*, ACM SPAA 06, Cambridge, MA, 2006.

[26] H. Kogan and I. Keslassy, *Fundamental complexity of optical systems*, IEEE Infocom '07 minisymposium, Anchorage, AK, 2007.

[27] A. Waksman, *A permutation network*, J. Assoc. for Comput. Mach., Vol. 15, No. 1, pp. 159-163, 1968.

[28] S. Iyer, R. Zhang and N. McKeown, *Routers with a single stage of buffering*, ACM SIGCOMM 02, Pittsburgh, USA, 2002.

[29] C. S. Chang, *Packet switch architectures course*, Chapter 4g, http://gibbs.ee.nthu.edu.tw/COM5353.htm

[30] G. Appenzeller, I. Keslassy and N. McKeown, *Sizing router buffers,* ACM SIGCOMM '04, Portland, Oregon, 2004.

[31] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown and T. Roughgarden, *Routers with very small buffers,* IEEE Infocom '06, Barcelona, Spain, 2006.

APPENDIX

A. Proof of Lemma 1

The proof is given by showing that it is possible to use an output-queued PIFO switch with a certain strategy as an implementation of a frames switch. Assume that every packet has a departure label and a priority label. The departure label indicates the packet's desired output link. The priority label dictates the departure ordering from the queues. The packets do not depart from the output queues until all the packets that belong to the current time frame arrive. The strategy is composed of the following rules:

- During the first B time-slots of operation, the packets do not depart from the queues.
- Set the departure label of each packet to be the desired output link.
- Set the priority label of each packet according to the desired departure time.

Since given an output-queued PIFO switch it is possible to implement a frames switch, the practical complexity of a PIFO switch is lower bounded by the practical complexity of a frames switch, $C_P^* \geq C_F^*$.

B. Proof of Lemma 2

In order to calculate the complexity of a frames switch we have to take the logarithm of the number of different states during time interval $[0, T]$, and divide it by T. By taking $T = KB$ and given the delay D, the total number of states $\#S_F$ is bounded as follows:

$$(NB)!^{K - \lceil \frac{D}{B} \rceil B} \leq \#S_F \leq (NB)!^{K - \lfloor \frac{D}{B} \rfloor B} \qquad (1)$$

Stirling's formula indicates that:

$$\log(N!) = N \log N - N + O(\log N).$$

Thus there exist N_0, C_1, C_2 such that

$$C_1 N \log N \leq \log N! \leq C_2 N \log N,$$

for every $N \geq N_0$. In order to find the complexity we will take the logarithm of the number of states and divide it by the time. Therefore, the minimal complexity of the combined space time permutation is given by:

$$C_1\alpha_1 N(\log(NB)) \leq C_F^* \leq C_2\alpha_2 N(\log(NB)),$$

where

$$\alpha_1 = \frac{K - \lceil \frac{D}{B} \rceil B}{K}$$

and

$$\alpha_2 = \frac{K - \lfloor \frac{D}{B} \rfloor B}{K}.$$

Taking $K \to \infty$ leads to

$$C_F^* = \Theta(N(\log(NB))).$$

C. Proof of Theorem 2

First, we will show that a TSI can be emulated using an optical buffer. This is done the following way: while the input frame is being written into the buffer, no packet that belongs to the current input frame is departing. Once all the packets have been written to the buffer, they start to depart according to the desired departure ordering.

We have found in Lemma 2 that the practical complexity of a frames switch with a certain delay is given by $C_F^* = \Theta(N\log(NB))$. The practical complexity of a TSI is given by the practical complexity of a frames switch where $N = 1$:

$$C_T^* = \Theta(\log B).$$

Since a TSI is a special case of an optical buffer, the practical complexity of an optical buffer is lower bounded by the practical complexity of a TSI:

$$C_B^* = \Omega(\log B).$$

D. Proof of Theorem 3

We will prove by induction that the construction presented in Figure 4 is an emulation of an optical buffer. For simplicity we will mark the fiber delay line of length B by P_1, the FIFO queue of size $B+x$ by P_2 and the FIFO multiplexer of size B by P_3.

First step of the induction: Assume that at time 0, P_1, P_2 and P_3 are empty. At time B, the packets that arrived until now are stored in P_1. During time interval $[B+1, 2B]$ the packets in P_1 pass through the 2×2 switch, and the central controller decides for each packet whether to direct it to P_2 or to P_3. The central controller sorts the packets using the departure requests during time interval $[1, B]$ and the priorities of the packets. The packets that should have departed from the ideal buffer during time interval $[1, B]$ are switched to P_3. The packets that should have remained in the ideal buffer and switched to P_2. Until time $2B$, all the packets that should have departed from an ideal buffer during time interval $[1, B]$ are sent to P_3 and are reordered before departing. To conclude, at time $2B$, P_2 contains the packets that should have been stored in an ideal buffer at time B and P_1 contains the new packets that arrived during time interval $[B+1, 2B]$.

Induction assumption for time kB, where $k > 2$: Assume that P_1 holds the packets that arrived during time interval $[(k-1)B+1, kB]$, P_2 holds the packets that should have been stored in the ideal buffer at time $(k-1)B$, and P_3 holds the packets that should have departed from the ideal buffer at times $[(k-2)B+1, (k-1)B]$.

The induction assumption holds at time $(k+1)B$: During time interval $[kB+1, (k+1)B]$, the central controller sorts the packets that are stored in P_1 and P_2: packets that should have remained in the buffer until time $kB+1$ are written to P_2. Packets that should have departed from the buffer during time interval $[(k-1)B+1, kB]$ are written to P_3. The length of time interval in which the sorting procedure is held is B. Since there are no more than B packets in P_1 and B packets in P_2 at the beginning of the time interval, all the packets are being sorted. By the end of the time interval, all the packets that should have departed from the buffer during time interval $[(k-1)B+1, kB]$ are either in the multiplexer or in the time-slot interchange. P_2 contains the packets that arrived during time interval $[kB+1, (k+1)B]$, and P_1 contains the packets that should have been stored in the ideal buffer at time kB.

The delay introduced by this construction is a sum of three factors, $D = d_1 + d_2 + d_3$:

- $d_1 = B$: at time $kB+1$ P_3 starts to receive the packets that should have departed from the ideal buffer at time interval $[(k-1)B, kB]$
- The FIFO multiplexer P_3 is not necessarily empty at time $kB+1$. There might still be packets from time interval $[(k-2)B+1, (k-1)B]$ that had not departed yet. The maximum number of packets from that time interval that are still stored in P_3 is $\frac{B}{2}$. Therefore, the additional introduced delay is $d_2 = \frac{B}{2}$.
- $d_3 = \frac{3B}{2}$: the inherent delay of a TSI [16].

The total resulting delay is $D = B + \frac{B}{2} + \frac{3B}{2} = 3B$.

E. Proof of Lemma 3

During time interval $[kB+1, (k+1)B]$, at each time-slot at most one packet enters the queue and at most one packet departs from the queue. Therefore, the number of packets stored in P_1 might increase by one, decrease by one, or remain the same. Also, at the edges of the time interval, no more than B packets are stored in the queue.

Therefore, the maximum number of packets stored in the queue at the middle of the time interval is $\frac{3B}{2}$.

F. Proof of Theorem 4

The construction in Figure 5 operates in a way similar to that of the construction in Figure 4. Consider the operation of the central 2×2 switch. If it is in a cross state, a packet from P_1 is sent to P_2, and the corresponding packet from P_2 is sent to P_3. If it is in a bar state, a packet from P_1 is sent to P_3, and the corresponding packet in P_2 remains in P_2. The first time interchange synchronizes the packets in P_1 such that no collision occurs:

- A packet in P_1 that should enter P_2 reaches the switch simultaneously with a packet in P_2 that should enter P_3.
- A packet in P_2 that should remain in P_2 reaches the switch simultaneously with a packet in P_1 that should enter P_3.

The delay introduced by this construction is composed of the sum of the delays of the two time-slot interchangers. Each time-slot interchange has delay $D = \frac{3B}{2}$, and the total delay of the construction results in $3B$. The delay with this construction is fixed, therefore it strongly emulates an optical buffer.

G. Proof of Lemma 4

We will show that the construction in Figure 6(a) is an exact emulation of an OQ-PIFO switch. The arriving packets are stored in a column of memory cells, where a memory cell is constructed as in [18] by combining a 2×2 switch and an FDL with length 1. The packets arriving on the input links are directed to empty memory cells through an $N \times NB$ switch. The packets that are scheduled to depart are read from the memory cells and placed on the appropriate output links through an $NB \times N$ switch. The input and output switches can be implemented using $O(NB\log(NB))$ 2×2 switches [9], [12]. Therefore, a total number of

$$C = O(NB\log(NB))$$

2×2 switches is necessary.

H. Proof of Lemma 5

We will show that the construction in Figure 6(b) is an exact emulation of an OQ-PIFO switch. There are N^2 parallel priority queues, one for each input-output pair. The arriving packets are sent to the appropriate priority queue according to the input link and the desired departure link. A central controller sends a departure request in each time-slot to the elements storing the packets about to depart in the current time-slot. Each construction of priority queue is emulated as described in [20]. It requires a $\sqrt{B} \times \sqrt{B}$ switch and $\sqrt{B}$ feedback FDLs. Therefore, each priority queue requires $O(\sqrt{B}\log B)$ 2×2 switches to implement. Therefore, a total number of

$$C = O(N^2\sqrt{B}\log(B))$$

2×2 switches is necessary.

I. Proof of Theorem 6

We will prove that the construction presented in Figure 7 is a strong emulation of an optical OQ-PIFO switch with $O(N\log NB)$ 2×2 switches and delay $D = 3B + N$. The OQ-PIFO switch is implemented using the pigeonhole principle with a modified departure process as explained in section VI-B. The constraint set is composed of the following rules:

1) Only one packet can enter a buffer at each time.
2) A packet will not enter a buffer on which there are $N-1$ previous packets or $N-1$ consequent packets destined to the same output link.
3) A packet will not enter a buffer which already contains B packets.

The first rule implies that a packet might collide with $N-1$ other arriving packets. The second rule implies that a packet must not enter at most $2N-2$ buffers which store packets that are potential to collide at the departure. The third rule implies that a packet must not be assigned to at most N buffers which already have B stored packets. Therefore, a minimal number of $4N-2$ buffers in parallel is necessary. We will emulate an optical buffer with the construction presented in Figure 5. A central scheduler controls the assignment of arriving packets into optical buffers and the departure of packets from optical buffers. The central scheduler's policy is as follows:

- *Lost packets:* If $B-x$ packets destined to output link j are stored in the OQ switch, and y packets arrive to destination j such that $y > x$, the $y-x$ packets with the lowest priority labels amongst the stored packets and arriving packets are lost. Lost packets that have just arrived to the OQ switch do not enter the OQ switch at all. For packets that are stored in the middle loop and should be lost, their position is marked as "empty". When a new packet enters the middle loop instead of the lost packet, the lost packet is simultaneously sent to the lost output link.
- *Assignment policy:* For each arriving packet with destination j that should enter the OQ switch, the

scheduler chooses a buffer such that the $N-1$ previous packets or $N-1$ consequent packets to the same destination are not stored in this buffer. According to the pigeonhole principle it is possible to find such an assignment.

- *Departure requests:* At each time-slot, the scheduler chooses one of the N destinations, in a round robin fashion. The scheduler sends departure requests to the buffers holding the N packets with the highest priority waiting to depart to the chosen destination. The N packets can depart simultaneously since they are stored on N different buffers due to the assignment policy.

The N departing packets are then concatenated in time according to their priorities. In addition, in order to have a strong emulation of an OQ-PIFO switch, i.e., to have a fixed delay for all the destinations, additional fiber delay lines are added at the output. The destination that is chosen first in the round robin encounters an additional delay of $N-1$ time-slots, whereas the destination that is chosen last doesn't encounter any additional delay.

The number of 2×2 switches in the construction is:

$$\begin{aligned} C(PIFO) &= 2C(Switch_{4N-2}) + 2(4N-2)C(\pi_B) \\ &+ 2C(Switch_N) + (4N-2) \\ &\le 16(N\log N + N\log B) \\ &= O(N\log(NB)) \end{aligned}$$

The delay of the construction is given by the sum of the delay of the optical buffer and the delay of the modified departure process, $D = 3B + N$.

J. *Proof of Theorem 7*

By Lemma 2, the practical complexity of a frames switch with any bounded delay is

$$C_F^* = \Theta(N\log(NB)).$$

From Lemma 1 it holds that:

$$C_P^* = \Omega(C_F^*) = \Omega(N\log(NB)).$$

The construction presented in Figure 7 switch introduces an upper bound on the practical complexity of an OQ-PIFO:

$$C_P^* = O(N\log(NB)).$$

Combining the upper bound and the lower bound, the practical complexity of an OQ-PIFO switch with delay $D = 3B + N$ is given by

$$C_P^* = \Theta(N\log(NB)).$$

K. *Proof of Theorem 8*

We will prove that the construction presented in Figure 7 is an emulation of a shared memory with capacity NB and PIFO departure policy. The difference between an OQ-PIFO switch with N output links and queues with capacity B and a PIFO shared memory with capacity NB is the lost mechanism. With a PIFO shared memory, arriving packets are lost only when there are already NB packets stored in the buffer. There is no limitation on the number of packets with the same destination that are stored in the buffer. The shared buffer is implemented with the same modified scheduling policy described in section VI-B, and $4N-2$ parallel optical buffer suffice with the same constraint set.