Emulation of Optical PIFO Buffers

Houman Rastegarfar
Department of Electrical and Computer Engineering
University of Toronto, Canada
houman.rastegarfar@utoronto.ca

Monia Ghobadi
Department of Computer Science
University of Toronto, Canada
monia@cs.toronto.edu

Yashar Ganjali
Department of Computer Science
University of Toronto, Canada
yganjali@cs.toronto.edu

***Abstract*—With recent advances in optical technology, we are closer to building all-optical routers than ever before. A major problem in this area, however, is the lack of all-optical memories similar to what we have in electronics. To overcome this problem, recently, there have been several proposals that show how we can emulate First-In First-Out (FIFO) queues using a combination of fiber delay lines and switches. Unfortunately, FIFO queues cannot be used for implementing many link scheduling policies including weighted fair queuing, weighted round-robin, or strict priority, which are essential components of any modern router today.**

In this paper, we introduce an architecture based on fiber delay lines and optical switches that can be used for emulating Push-In First-Out (PIFO) queues. In a PIFO queue, an incoming packet can be pushed anywhere in the queue, and therefore it can be used for the implementation of various link scheduling policies. We describe a scheduling algorithm for this architecture and show that with a small speedup, we can build a PIFO queue of size $N-1$ using only $O(\log^2 N)$ 3×3 optical switches. The resulting system has a minimum reliability of 99.5%, and even for the small portion of departure requests that cannot be fulfilled immediately, the requested packet is ready to depart within approximately five time slots from the request time.

I. Introduction

Optical transmission and switching technologies based on Wavelength Division Multiplexing (WDM) have been increasingly deployed in the Internet infrastructure in order to meet the ever-increasing demand for bandwidth. As a result, several efforts have been made to improve the performance of the optical networks so as to eliminate the existing electronic bottlenecks by utilizing all-optical switches. There are several functions of critical importance for the realization of all-optical routers among which optical buffering is of great importance [11]. The lack of a straightforward technique for storing information in the optical domain remains a major roadblock for building all-optical routers. In all-optical packet switch designs, one way for storing optical packets is to use a combination of Fiber Delay Lines (FDLs) and switches to delay packets for a certain period of time [3], [4], [5], [7].

Recently, there have been several proposals for emulating First-In First-Out (FIFO) queues with the objective of minimizing the size of the switch, as well as the length and the number of the fiber delay lines [10], [4], [3]. Unfortunately, FIFO queues cannot be used for implementing many link scheduling policies including weighted fair queuing [12], weighted round-robin [8], and strict priority. These scheduling policies are essential components of any modern router today, as they are the basis for providing Quality-of-Service in routers. Unlike a FIFO queue in which the incoming packets can only join the tail of the queue, in Push-In First-Out (PIFO) queues new packets can be pushed anywhere in the queue. Given the freedom of choosing where to insert the new packet, PIFO queues are significantly more flexible than FIFO queues and can be used for implementing various scheduling policies such as weighted fair queuing, weighted round-robin, and strict priority as mentioned above.

This paper is mainly motivated by Beheshti *et al.*'s work in which an optical FIFO buffer architecture is proposed [3]. Their proposed scheme achieves a buffering capacity of $N-1$ packets by using only $O(\log N)$ 2×2 optical switches, which has been shown to be the lower bound on the size of the switches required [10]. In this paper, we extend their approach and introduce an architecture based on fiber delay lines and optical switches that can be used for emulating PIFO queues. As in input-queued switches, we assume that the departure time of a packet is not known in advance. At the same time, since the incoming packets can be placed at any location in the queue, coming up with a scheduling algorithm is more challenging. We show that with a small speedup, we can build a PIFO queue of size $N-1$ using only $O(\log^2 N)$ 3×3 switches. We also show that the resulting system is extremely reliable: The system can handle more than 99.5% of the departure requests immediately, and the maximum delay to handle a request which is not fulfilled immediately is within approximately five time slots from the request time.

The rest of this paper is organized as follows. A short review of the previous works is presented in Section II. Our optical PIFO buffer emulation architecture and the scheduling algorithm are described in Section III and the simulation results studying the performance of our proposed design are presented in Section IV. Finally, Section V concludes the paper.

II. Related Work

The problem of realizing all-optical buffers using a combination of fiber delay lines and optical switches has been addressed in [3], [4], [5], [7], [10], and [6]. Sarwate *et al.* showed that for emulating any priority queue, the minimum number of required delay lines is $O(\log N)$. They proposed an architecture to emulate a FIFO queue of size N with $O(\sqrt{N})$ delay lines [10]. A recursive approach for constructing optical FIFO multiplexers with $O(\log N)$ delay lines has been proposed by Chang *et al.* [5]. However, their proposed

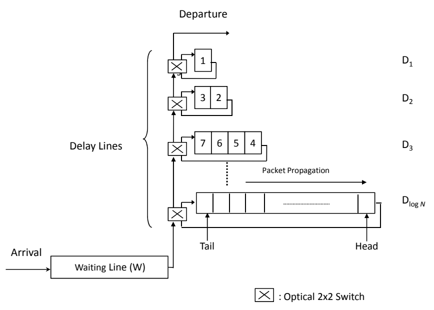

Fig. 1. Emulation of a FIFO buffer using fiber delay lines.

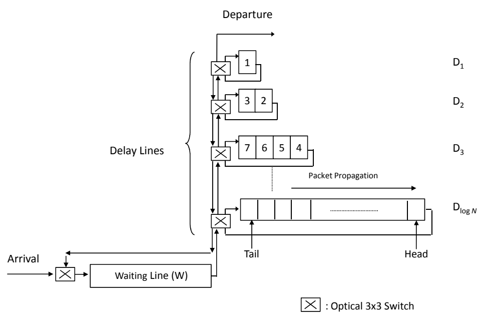

Fig. 2. Emulation of a PIFO buffer using fiber delay lines.

construction needs to keep track of the shortest and the longest queues in each recursion step.

Beheshti *et al.* introduced a buffering architecture which consists of $2\log N - 1$ delay lines of fixed length, and is capable of exactly emulating a FIFO queue of size $N - 1$ [3]. Their proposed architecture is shown in Fig. 1, in which the delay loops are of exponentially growing lengths. In each delay loop, a delay line is coupled with a 2×2 optical switch which connects the delay line to the main path between the arrival and departure ports. In this architecture, the incoming packets are buffered by going through a subset of delay lines. Upon the arrival of a packet, the scheduler decides whether the incoming packet needs to go through the waiting line or it can be directly forwarded to one of the delay lines. As time progresses, optical packets in each delay line move in the direction shown in Fig. 1 towards the head of that delay line [3]. The waiting line, W, is a recursively emulated FIFO buffer, and is used to prevent void places between packets with successive departure orders. In other words, since the interarrival intervals are not known in advance, the proposed system uses this waiting line to adjust the location of packets in the delay lines, and thus packets with successive departure orders will be placed back to back in the delay lines.

This work is inspired by the architecture presented in [3] to emulate PIFO buffers using a recursive architecture. The architecture of the new system, and the packet scheduling algorithm are explained in the following section.

III. Optical PIFO Buffer Emulation

We propose the architecture in Fig. 2 to emulate a PIFO buffer using fiber delay lines. The architecture includes a group of delay lines $D_i, i = 1, 2, \cdots, \log N$, where each delay line consists of L_i delay units $d_{head,i}, d_{head+1,i}, \cdots, d_{tail,i}$. The length of each delay line grows exponentially as $1, 2, 4, \cdots, 2^{(\log N)-1}$, generating an overall queue length of $N-1$. Without loss of generality, we assume that N is a power of 2. Note that the waiting line, W, is also a PIFO buffer built recursively using the same infrastructure as of the main PIFO buffer.

In a PIFO queue, the departure order of a packet determines its position in the buffer; e.g., a packet with departure order 1 has the highest departure priority and should be pushed to the head of the buffer. In our proposed structure, packets can be read only from the head of a delay line and can be written only to the tail of a delay line. During each time slot, our emulation algorithm performs the following tasks in order: (1) serving a possible arrival event, (2) serving a possible departure event, and (3) performing the scheduling task. These tasks are explained in more detail below.

A. Arrival Event

An incoming packet with departure order x, $1 \leq x \leq N-1$, should be pushed to its corresponding delay unit in one of the delay lines. For example, according to Fig. 2, an incoming packet with departure order $x = 3$ should be pushed to the tail of delay line D_2. Note that an incoming packet will be accepted only if the current occupancy of the PIFO buffer is less than $N - 1$ (maximum size of the buffer). Otherwise, the system drops the packet. The accepted arrival request is managed based on the following two steps:

1) In the first step, our algorithm increments the departure order of all packets with departure order greater than or equal to x by one unit. This affects the packets in the delay lines and the waiting line. This step is carried out in order to be able to place the new packet at the proper position in the buffer and to preserve the packets' departure order. Furthermore, our algorithm determines D_{prop}, the proper delay line that the incoming packet should be inserted to. More precisely, if the incoming packet's departure order is x, then $prop = \lfloor \log x \rfloor + 1$ and the packet should be placed at delay line $D_{\lfloor \log x \rfloor + 1}$.
2) In the second step, our algorithm checks the occupancy status of D_{prop} and performs one of the following cases:
   a) If D_{prop} is completely empty, the packet will be marked to be placed at the tail of D_{prop} at the end of the current time slot.
   b) If there is a packet at the tail position of D_{prop} and its departure order is $x - 1$, the incoming packet will be marked to be inserted into D_{prop} at the end of the current time slot.
   c) In any other case, the system holds the incoming packet in the waiting line for a number of time slots until the scheduler places this packet at its proper position. This process will be explained in more

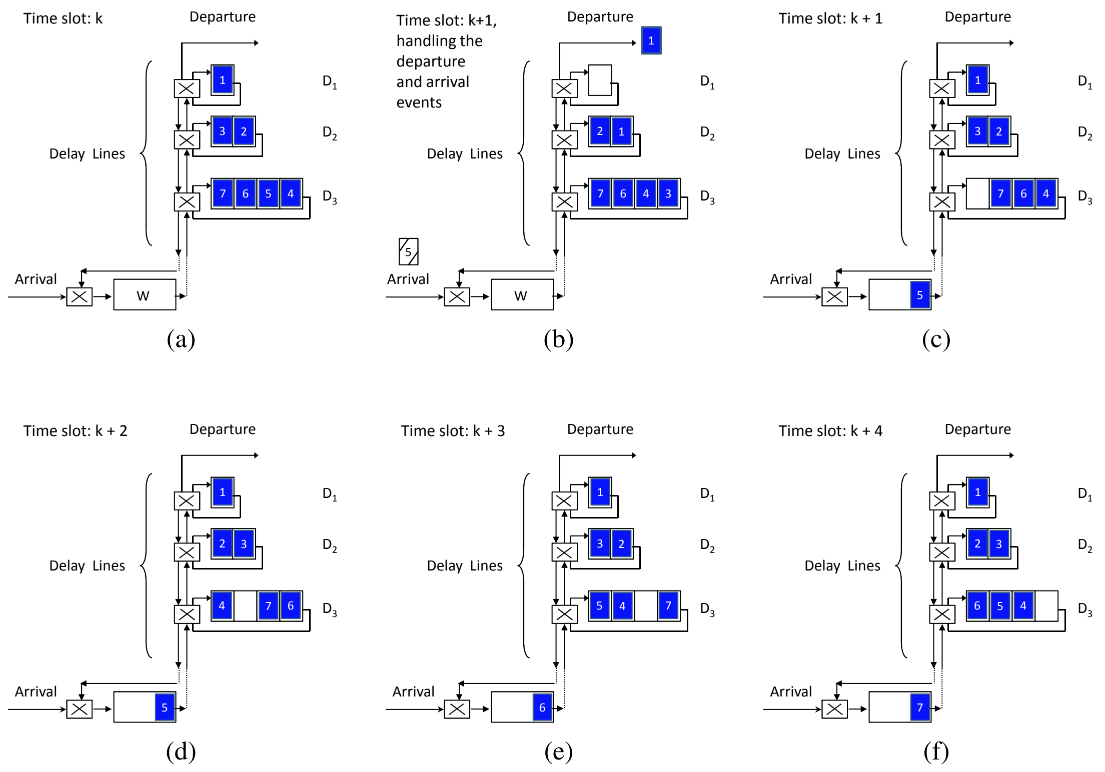

Fig. 3. Example of an emulated PIFO buffer.

detail in Section III-C, handling the scheduling task.

B. Departure Event

During each time slot, a departure request for the packet with the highest priority, i.e., with departure order equal to 1 may occur. In this case, if this packet is at the head position of one of the delay lines, it will successfully depart from the system. Moreover, our algorithm decrements the departure order of all packets in the system including the ones in the waiting line by one unit. However, there is the possibility that the packet with departure order 1 is not at the head of any delay line yet, rather in the middle of a delay line. In this case, our system fires an error indicating that it cannot address the departure request, immediately. The departure request will eventually be addressed during a future time slot in which the packet reaches the head position of the delay line it is currently in. In our simulations, we have studied the error rate and the delay in serving departure requests, and have observed that the error rate is always less than 0.5% and that the delay in serving departure requests is negligible. Our simulation results are explained in more detail in section IV.

C. Scheduling

The scheduling task is performed at the end of each time slot. The main idea of the scheduling task is to keep packets in the delay lines according to proper departure orders; i.e., packets with successive departure orders should be placed back to back in the delay lines. In our proposed recursive design, during each time slot at most one push to and at most one pop from the waiting line can take place. According to this constraint, the scheduling task is based on the following three steps, respectively.

1) The scheduler iterates over the delay lines $D_i, 1 \leq i \leq logN$ starting from D_1 and checks the head and the tail positions in each of them to find delay lines such as D_j for which the tail position is nonempty and either one of the following two conditions is satisfied:
   a) The head position in D_j is empty, or
   b) the head position in D_j contains a packet with departure order, x_{head}, greater than that of its tail position, x_{tail}, by more than one unit, i.e., $x_{head} > x_{tail} + 1$.

   If the second condition is satisfied and no other packet has been previously selected to be inserted into the waiting line during the current time slot, the scheduler marks the head-of-line packet of D_j to be transferred to the waiting line. Then it checks to see whether the head-of-line packet of the waiting line, p_c, can be transferred to D_j. If so, p_c will also be marked to be transferred to the tail of D_j at the end of the current time slot.
2) To prevent the accumulation of void places in the delay lines after each successful departure event, the scheduler tries to fill the empty delay units by marking packets at the head of a delay line ($d_{head,i}$) to be transferred to the tail of the proper delay line (D_{prop}). Of course, this can happen only if the candidate delay unit is empty.

3) The scheduler performs all packet transfers in this step, simultaneously. In other words, optical packets in each delay line move by one delay unit in the direction shown in Fig. 2 towards the head of the corresponding delay line. At the same time, the marked packets in the previous steps will move towards their determined positions.

Fig. 3 illustrates an example of an arrival event, a departure event, and the scheduling task in our PIFO emulation algorithm. Let us assume that the system is in the state shown in Fig. 3(a) at time slot k with 7 packets $p_1, p_2, \cdots, p_7$ having departure orders equal to $1, 2, 3, \cdots, 7$, respectively. Furthermore, let us assume that at time slot $k+1$ a packet p with departure order $x_p = 6$ enters the system, and that there is also a departure request for packet p_1. As described above, the system first serves the arrival event, then the departure event, and finally performs the scheduling task. The system starts with step 1 in the arrival event by incrementing the departure order of packets p_6 and p_7 by one unit. Then, our algorithm determines that p should be pushed to $D_{prop} = D_3$ (since $\lfloor \log x_p \rfloor + 1 = 3$). Based on case 2c handling the arrival event, p will be marked to be inserted into the waiting line at the end of this time slot. Next, the system serves the departure request by sending packet p_1 out of the system and decrementing the departure order of all packets in the system by one unit. The updated status of the system is illustrated in Fig. 3(b), where the packet on the arrival line indicates the marked packet p. Further, during the scheduling phase none of the conditions in scheduling step 1 are satisfied, and therefore the system only follows scheduling steps 2 and 3 in which the marked packets are placed at their determined positions while packets are circulated in the delay lines (Fig. 3(c)). In the next time slot, assuming there are no arrival and no departure events, the system only performs the scheduling task. Neither of scheduling steps 1 or 2 applies and hence the system follows step 3 and circulates packets in the delay lines (Fig. 3(d)). During time slot $k+3$, condition (b) in step 1 of the scheduling task is satisfied because the difference between the head and the tail packet's departure order in delay line D_3 is 2. As a result, the scheduler marks the packet with departure order 6 to be transferred to the waiting line. It also marks the packet with departure order 5 to be transferred to the tail of D_3, and proceeds to the next steps of the scheduling task (Fig. 3(e)). The same event as in time slot $k+3$ is repeated in the next time slot, $k+4$, leading the packet with departure order 6 to be transferred from the waiting line to the tail of D_3, and the packet with departure order 7 to be transferred to the waiting line (Fig. 3(f)). Finally at time slot $k+5$, condition (a) of step 1 in the scheduling task is satisfied, causing the only packet in the waiting line to be inserted into delay line D_3, and hence at the end of this time slot, the status of the system will be the same as in Fig. 3(a).

Assuming that the occupancy of the waiting line is always smaller than $N/2$, the theorem below shows that our structure needs $O(\log^2 N)$ delay lines to emulate a PIFO buffer of size N. We explore the feasibility of this assumption in Section IV.

Theorem 1: In our PIFO structure, the number of delay lines required is $O(\log^2 N)$: As we recursively build a PIFO buffer of size N, the total number of delay lines required, $T(N)$, is equal to $\log N + T(N/2)$. The first term is the total number of delay lines D_i in Fig. 2, i.e., the main PIFO buffer, and the second term is the number of delay lines required for the waiting line. Thus:

$$\begin{cases} T(N) = \log N + T(N/2), & N > 1 \\ T(1) = 1, & N = 1 \end{cases} \tag{1}$$

Assuming $N = 2^k$, one can easily solve the above recursive formula as $T(N) = O(log^2 N)$.

IV. Simulation Results

TABLE I
Simulation parameters

Parameter	Value(s)
Simulation time slots	1000000
Optical PIFO queue length	127, 255, 511, 1023
Packet arrival rate	0.25+0.025i, i=0,1,...,9
Packet departure rate	0.50

We simulate our proposed optical PIFO buffer architecture to study several performance parameters including:

- Reliability of the system as the percentage of the departure requests that can be served immediately.
- The distribution of the delays incurred in serving the departure requests.
- Relative occupancy of the waiting line as the ratio of its maximum length to the actual buffer length $(N-1)$.

We run several simulations with different system loads and PIFO queue lengths of 127, 255, 511, and 1023. Our simulation parameters are summarized in Table I. We use Bernoulli IID random variables to generate packet arrival and departure requests.

The pseudocode illustrated in Fig. 4 summarizes our simulation program in which variable *count* is the main loop variable and varies between 1 and ST, where ST denotes the number of simulation time slots. Arrival and departure requests are issued using the "Mersenne twister" pseudorandom number generator implemented in [1] according to the arrival and the departure rates, respectively.

In this paper, each simulation result is the average of 200 rounds of simulation run. Since the distribution of the variables under consideration is unknown, we compute confidence intervals using the method of batch means with 10 batches of 20 samples as explained in [9]. Here, we find the 95% confidence intervals for the mean of the random variables as $[\overline{X}_{10} - t_{0.025,9}\frac{\hat{\sigma}_{10}}{\sqrt{10}}, \overline{X}_{10} + t_{0.025,9}\frac{\hat{\sigma}_{10}}{\sqrt{10}}]$ where $\overline{X}_{10}$ is the sample mean of the batch sample means, $\hat{\sigma}_{10}$ is the sample standard deviation of the batch sample means, and $t_{0.025,9}$ is

```
Optical PIFO Buffer Simulation Pseudocode

set the simluation parameters (rates, buffer length)
ST ← number of simulation time slots
count ← 1
while count ≤ ST
    if an arrival request exists and the buffer is not full then
        generate the order of the incoming packet (x) at random
        insert packet with order x into the queue
    end if
    if a departure request exists and the queue is nonempty then
        if the packet with order 1 is at the head of a
        delay line then
            remove the packet from the system
        else
            log the departure error
        end if
    end if
    schedule the whole system
    count ← count+1
end while
```

Fig. 4. Optical PIFO buffer simulation pseudocode.

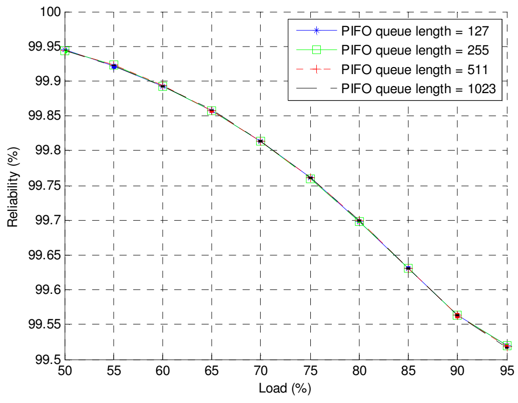

Fig. 5. Reliability of the optical PIFO buffer versus load.

equal to 2.262 according to Table A.5 (Student's t-distribution table) in [2].

Fig. 5 illustrates the reliability of the emulated optical PIFO buffer versus load for different queue lengths. As mentioned earlier, reliability is defined as the percentage of departure requests that can be served immediately. As can be observed in Fig. 5, in all of the simulation scenarios, the reliability is greater than 99.5% which is a very promising value. The reliability of the optical PIFO buffer tends to fall slightly when the load increases. Another interesting observation is that the size of the optical PIFO buffer does not have any effect on system reliability in the studied cases.

In order to obtain a more detailed view of the performance of the optical PIFO buffer, we study the distribution of the delays incurred in serving the departure requests under various circumstances. As described in section III-B, when a departure request is issued, if the packet with departure order 1 is at the head of a delay line, then the request will be served without delay. However, in some cases the packet with departure order 1 might not be at the head of any delay line as the departure request is issued. This corresponds to a departure error, and the requested packet needs a certain number of time slots to reach the head of the delay line in which it is located. Figures 6(a), 6(b), 6(c), and 6(d) show the Cumulative Distribution Function (CDF) of the departure delays for queue lengths of 127, 255, 511, and 1023, respectively. In all of the figures, we can observe an increasing trend in the experienced delay as the system load increases. However, the maximum departure delay is almost limited to five time slots in all cases. In other words, when a departure error happens, the requested packet is expected to be able to leave the system within five time slots. This is important in the sense that if the five time slot delay constraint is tolerable, then we can almost serve any departure request. Finally, note that the queue length does not have any distinguishable effect on the CDF of the departure delays for the system loads under consideration.

In the last part of this section, we study the feasibility of our proposed structure by measuring the size of the recursively built waiting line. The realization of the optical PIFO buffer depends on the assumption that the size of the waiting line, W, does not grow faster than N. In our simulations, we keep track of the waiting line's occupancy during each time slot. Fig. 7 illustrates the maximum occupancy of the waiting line versus load for different buffer sizes. The vertical axis represents the ratio of the waiting line's maximum length to the actual buffer length during the simulation time, ST, and the horizontal axis indicates the load. From the figure, it is obvious that the occupancy of the waiting line is always less than that of the emulated buffer (as it never gets to 100%). This implies that we are able to use the concept of recursion in realizing the proposed optical PIFO buffer under certain circumstances. Moreover, in all of the simulation scenarios, the relative occupancy of the waiting line is less than or equal to 50% for loads below 92.5%. Hence we can consider its size to be half of the size of the emulated buffer. By applying Theorem 1 we establish the conclusion that using a negligible speedup (e.g., 1.08 for buffer size equal to 127), we can construct a PIFO queue of size N by using $O(\log^2 N)$ 3×3 switches. The resulting queue has a minimum reliability of 99.5%, and can address the departure requests within approximately five time slots.

Realization of the proposed optical PIFO buffer seems to be challenging in some cases when the system load exceeds 90%. In other words, several recursions may be necessary to build such a buffer, since the occupancy of the waiting line is found to be comparable to that of the emulated buffer. However, according to Fig. 7, the interesting fact is that the relative occupancy of the waiting line falls sharply as the actual buffer size increases. In other words, a much smaller speedup is required to build the optical PIFO buffer when its size increases.

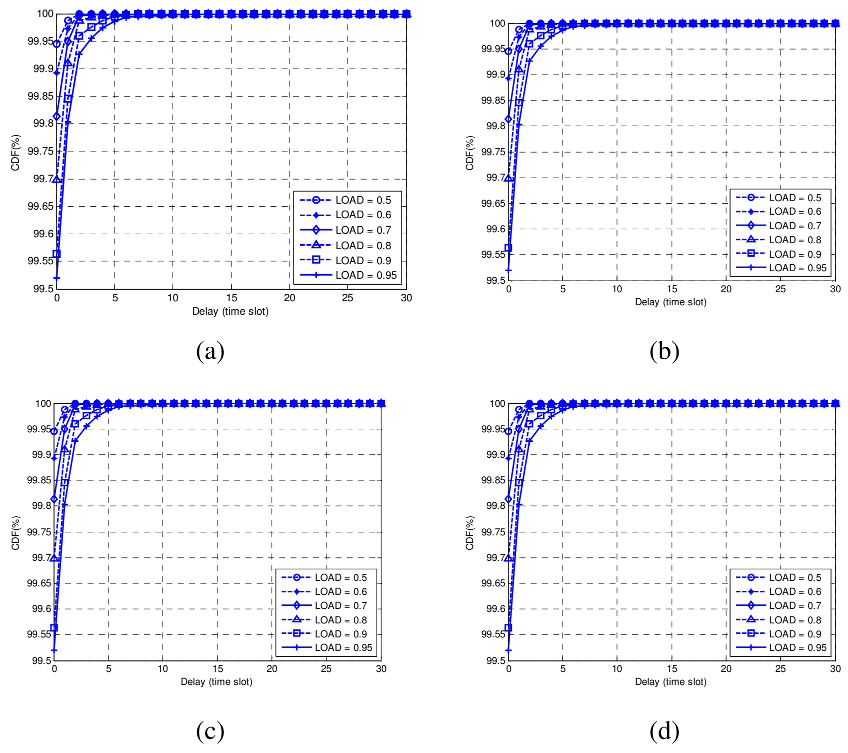

Fig. 6. Cumulative distribution function of departure delays (a) queue length = 127, (b) queue length = 255, (c) queue length = 511, (d) queue length = 1023.

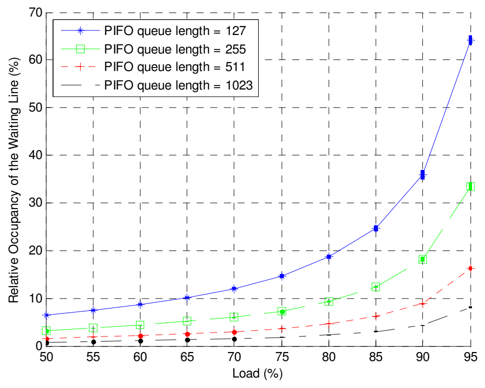

Fig. 7. Relative occupancy of the waiting line versus load.

V. Conclusions

In this paper, we introduced an architecture based on fiber delay lines and optical switches that can be used for emulating Push-In First-Out (PIFO) queues. We proposed a recursive architecture to minimize the number of delay lines and switches. We also described a scheduling algorithm for this architecture, and showed that with a negligible speedup, we can build a PIFO queue of size $N-1$ using only $O(\log^2 N)$ 3×3 optical switches. The resulting system has a minimum reliability of 99.5%, and can address departure requests within approximately five time slots. It is also necessary to note that the performance of the optical PIFO buffer in terms of reliability and delay is independent of queue length for the system loads under consideration.

References

[1] Pseudo Random Number Generators: Uniform and Non-Uniform Distributions. http://www.agner.org/random/, November 2008.

[2] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice Hall, fourth edition, 2005.

[3] N. Beheshti and Y. Ganjali. Packet scheduling in optical FIFO buffers. In *Proceedings of IEEE INFOCOM High-Speed Networks Workshop*, 2007.

[4] C. S. Chang, Y. T. Chen, and D. S. Lee. Constructions of optical FIFO queues. *IEEE/ACM Trans. Netw.*, 14(SI):2838–2843, 2006.

[5] 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*, 50(12):3221–3233, 2004.

[6] R. L. Cruz and J. T. Tsai. COD: alternative architectures for high speed packet switching. *IEEE/ACM Trans. Netw.*, 4(1):11–21, 1996.

[7] D. K. Hunter, M. C. Chia, I. Andonovic, and S. Member. Buffering in optical packet switches. *IEEE Journal of Lightwave Technology*, 16:2081–2094, 1998.

[8] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communications*, 9(8):1265–1279, 1991.

[9] A. Leon-Garcia. *Probability, Statistics, and Random Processes for Electrical Engineering*. Prentice Hall, Chapter 8, third edition, 2008.

[10] A. Sarwate and V. Anantharam. Exact emulation of a priority queue with a switch and delay lines. *Queueing Systems: Theory and Applications*, 53(3):115–125, July 2006.

[11] K. M. Sivalingam and S. Subramaniam. *Emerging Optical Network Technologies: Architectures, Protocols and Performance*. Springer, Chapter 5, 2004.

[12] D. Stiliadis and A. Varma. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. Technical report, Santa Cruz, CA, USA, 1995.