Packet Scheduling in Optical FIFO Buffers

Neda Beheshti, Yashar Ganjali
Computer Systems Laboratory
Department of Electrical Engineering, Stanford University
{nbehesht, yganjali}@stanford.edu

Abstract— **Recent advances in optical technology show the possibility of building all-optical buffers in the near future. These buffers are usually composed of a number of fiber delay lines (FDLs) and optical switches. Incoming packets are stored for a limited time by going through optical delay lines. Optical switches transfer these packets among different delay lines, or send them towards the output line if a packet is to leave the system. As a direct consequence of using optical technology, one of the major constraints in this setting is that the size of switch needs to be small.**

In this paper, we show the feasibility of constructing a FIFO queue of size N by using only $O(\log N)$ 2x2 switches. A simple scheduling algorithm that achieves this bound is developed. The proposed structure provides an efficient way of storing optical packets using a minimal number of delay lines and switches.

I. INTRODUCTION

Packet-switched communication networks need buffers for resolving contention among packets that compete for the same link. In all-optical packet switch designs, fiber delay lines (FDL) are commonly proposed as a means of delaying optical packets, and hence storing them during times of contention.

In this paper we propose an optical FIFO architecture for buffers in which the departure time of packets is not known in advance – as in input queued switches. The presented scheme achieves a buffering capacity of N packets by using only $O(\log N)$ 2x2 optical switches. This has been shown to be the lower bound on the size of required switches [8].

The work is mainly motivated by the recent results that show the ability to make small onchip optical packet buffers [6], and suggest that this small onchip buffering suffices for resolving packet contention in the core routers [4], [7]. These results raise the question of finding the maximum buffering capacity that can be achieved by using only a small number of integrated optical switches.

The framework we consider in this paper is for exact emulation of a FIFO queue, i.e., for any arbitrary sequence of arrivals and departures, the scheduling algorithm guarantees that the system has exactly the same departure and drop sequences as its equivalent FIFO queue. What makes the delay-line based design of optical FIFOs challenging is that the arrival and departure time of packets are not known in advance.

There have been many schemes proposed for buffering optical packet buffering . Almost all of the schemes involve using a system comprised of an optical switch and one or more fiber delay lines to delay the packets for a certain amount of time. Fig. 1 shows a schematic of this system where an optical
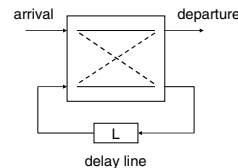

Fig. 1. An optical buffer composed of a delay line and a 2x2 switch.

delay line of a fixed length L is coupled with a 2x2 optical switch. An arriving packet can either be transferred to the departure port directly, or it can be switched into the delay line and recirculate in the loop for a number of times before departing the system.

Clearly, a FIFO queue of size N can be built by concatenating N delay loops, each capable of holding one packet. A consequence of using optical technology, however, is the constraint it imposes on the size and the number of optical switches used in this setting. The challenge is to find a structure that works with a smaller number of switches, and yet has the same buffering capacity. More specifically, the question we address in this work is the following: Given that a fixed-length delay line can generate only a certain amount of delay, what is the minimum number of delay lines required for emulating a FIFO structure, where the buffering duration of packets is not known in advance?

Our results show that the buffering capacity of an optical FIFO architecture can grow exponentially with its size. In other words, the proposed scheduling algorithm requires the number of delay lines, or equivalently the number of optical 2x2 switches, to grow only as the logarithm of the number of packets in the buffer.

A. Related Work

The problem of constructing optical queues and multiplexers with minimum number of optical delay lines and optical switches has been addressed in [5], [1], [8], [2], [3]. In a FIFO multiplexer the departure time of packets is known upon their arrivals, whereas in a FIFO queue, the departure time can not be determined in advance. In [8] it is shown that for emulating any priority queue of length N, the minimum number of required delay lines is $O(\log N)$. A construction which achieves this emulation by $O(\sqrt{N})$ delay lines is presented.

Recently, C.S. Chang et al. proposed a recursive approach for constructing optical FIFO buffers with $O(\log N)$ delay

lines [1]. The proposed construction, however, needs to keep track of the longest and the shortest queues in each step of the recursion.

II. Buffering Architecture

In this section we present a buffering architecture which consists of $2logN - 1$ delay lines of fixed lengths, and is capable of exactly emulating a FIFO queue of size $N - 1$ in the following sense:

Exact emulation: system A is said to exactly emulate system B if by applying the same sequence of arrival packets and departure requests to both systems, the output and drop sequences in system A will be indistinguishable from the corresponding sequences in system B.

Without loss of generality, In all our analysis and proofs we assume N is a power of 2, unless otherwise stated.

As depicted in Fig. 2, the delay loops are of exponentially growing lengths. In this figure, the delay loops are shown as straight lines for the sake of presentation. Each Delay line comes with a 2x2 switch which connects it to the main path between the arrival and the departure ports.

Incoming packets are buffered by going through a subset of the optical delay lines. Upon an arrival request, the scheduler decides if the arriving packet needs to go through the waiting line first, or if it can be directly delivered to one of the delay lines. As time progresses, optical packets in each delay line move in the direction shown in Fig. 2 towards the head of delay lines. At the end of each time slot, a scheduler determines the next position of the head-of-line packets. The system directs these head-of-line packets towards their scheduled positions by configuring the 2x2 switches corresponding to each delay line.

The waiting line W operates as a time regulator of the arriving packets. Since the time interval between successive arriving packets is not known in advance, the system uses this waiting line to adjust the location of packets in delay lines. When the system decides to direct an arriving packet to the waiting line, it knows how long exactly the packet should be kept there, before sending it out to one of the delay lines. Packets leave W in a FIFO order. In what follows we first show that this structure along with our proposed algorithm can emulate a FIFO queue of length $N - 1$. We then explain how the waiting line W can itself be constructed by $\log N - 1$ delay lines.

A. Packet Scheduling

The main idea of the scheduling algorithm is to place the packets in delay lines consecutively, i.e., packets with successive departing orders are placed back to back in delay lines. This is in contrast with what is proposed in [8], where arriving packets are allowed to fill out any available positions at the tail of delay lines. In order to follow this rule, an arriving packet can only be placed in a delay line if its preceding packet has been in a tail position in the previous time-slot. Otherwise, the system holds the arrived packet in W for a proper number of time slots. The waiting time is equal to the time it takes for

the preceding packet to traverse the delay line it is currently in, and gets switched to a tail position. During the waiting time, all the arrived packets are kept in W in a FIFO order.

At a departure request incident, if the system is non-empty, the packet with departing order 1 is delivered to the output link. The algorithm is shown to guarantee that this packet is always at the head of some delay line. The departing order of every remaining packet in the system is then reduced by one.
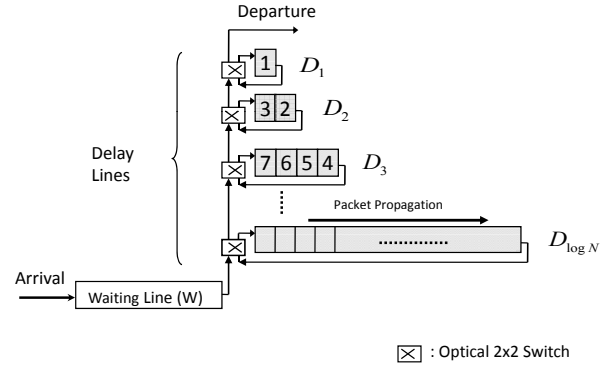

Fig. 2. Emulating a FIFO queue using delay lines. The system regulates the position of the arriving packets by passing them through a waiting line.

At the end of each time slot, the next positions of the head-of-line packets are determined by the scheduler. The scheduler decides whether to recirculate a packet in the same delay line it is currently in, or to send it to a shorter delay line. This decision is made independently for each head-of-line packet, based only on the packet's departing order as follows: in the next time slot, the packet will be transferred to the tail position of the longest delay line whose length is not greater than the order of the packet. More precisely, if the head-of-line packet has departing order k, then it will be placed in delay line $D_{\lfloor \log k \rfloor + 1}$. As will be shown later, this ensures that packets are at the head of some delay line when their departing order goes down to 1.

The same scheduling policy applies when the waiting time of the head of line packet in W is decremented to zero: the packet will be transferred to the tail position of the longest delay line whose length is not greater than the departing order of the packet.

The main result of the paper is stated in the following theorem.

Theorem 1. Under scheduling algorithm A, the set of delay lines D_i and waiting line W can exactly emulate a FIFO queue of size $N - 1$.

In the appendix we prove theorem 1 by showing that Algorithm A satisfies the following three properties.

 (i) Occupancy of W is always smaller than $N/2$.
 (ii) There is no contention among head of delay line packets.
(iii) Packet with departure order 1 is always at the head of a delay line.

Algorithm A- Packet Scheduling

1) arrival event

  let k be the total number of packets in the system.

  if $k = N - 1$ **then**

    drop the arrived packet

  else

    denote by p_{k+1} the arrived packet and by p_k its prior packet in the system.

    if p_k is in W **then**

      place p_{k+1} in W

    else

      $waiting\_time \leftarrow (h+1) \mod d$, where d is the length of the delay line which contains p_k, and h is the distance of p_k from the head of the line.

      if $waiting\_time > 0$, place the arrived packet in W. Otherwise, place it in the longest queue with length smaller than or equal to the order of the packet.

2) departure event

  remove the packet with order 1 from the system, and decrease the order of all packets in the system by 1.

3) scheduling the head of line packets

  for $i = 1, 2, ..., \log N$, move the packet at the head of D_i to the longest delay line with length smaller than or equal to the order of the packet.

  if $(waiting\_time > 0)$ **then**

    $waiting - time \leftarrow waiting\_time - 1$

  if $(waiting\_time = 0)$ & (W is nonempty) **then**

    remove the head of line packet from W. Place the packet in the longest delay line with length smaller than or equal to the order of the packet.

The first property is required for constructing the waiting line W with $log N - 1$ delay lines. This construction is explained in the next sub-section.

The scheduling algorithm must relocate head of line packets in such a way that there is no contention for a given location, i.e., at any time slot, at most one packet should be scheduled to be switched into each delay line. Therefore, the second property must be hold to avoid dropping packets after they are admitted to the buffering system.

Finally, the last property indicates that the departure sequence will exactly follow that of a FIFO, without any delay.

The above properties guarantee that an arriving packet will be buffered in the system as long as the total number of packets in the system remains less than N. Moreover, the packets depart in the same order they arrive at the system each packet is delivered to the output link at the same time as in the emulated FIFO system.

B. Constructing Waiting Line

To avoid future contention among head of line packets, the scheduling algorithm does not allow any void places between packets which have successive departing order. To achieve this, an arriving packet is kept in the waiting line until its preceding packet – in arriving order, or equally in departing order –

passes the tail of a delay line. We show that the waiting line W can be constructed by $\log N - 1$ delay lines, resulting in a total of $2 \log N - 1$ delay lines.

In this section we explain how the waiting line W can be constructed by $\log N$ fixed-length delay lines. As shown in previous sections, W should be able to buffer at most $N/2$ packets, as our algorithm always keeps the number of packets in the waiting line less than $N/2$.

Consider a group of delay lines $D_i', i = 1, 2, ..., \log N - 1$, where the length of the delay lines grows as $1, 2, 4, ...2^{(log N)-1}$, generating an overall delay length of $N - 1$. Upon the arrival of each packet to the system, the scheduler knows how long the packet should be kept in W before being transferred to one of the delay lines. Based on the availability of this information we develop algorithm B in the following way. When a packet p arrives to the system, and is scheduled by algorithm A to be placed in W then

- Calculate the binary expansion of the waiting time of p, i.e., the duration that p needs to wait in W before moving to one of the delay lines $\{D_i\}$. This determines which delay lines from the set $\{D_i'\}$ the packet should traverse before leaving W; if the ith bit in the binary representation is non-zero, then the packet needs to traverse delay line D_i'.
- Starting from the shortest line, when the packet reaches the end of a delay line, place it in the next delay line corresponding to the next non-zero bit.

Packet p leaves W when it traverses all the delay lines corresponding to the non-zero bits in its binary expansion. A *waiting* packet traverses any delay line of the set $\{D_i'\}$ at most once, and never recirculates in the same delay line.

III. CONCLUSIONS

The main result of this work is introducing an architecture for building optical FIFO buffers. The proposed structure uses a simple mechanism to control the location of the arriving optical packets by by using only $O(\log N)$ 2x2 optical switches which has been shown to be the lower bound on the size of required switches. The buffering capacity of the presented architecture grows exponentially with the number of optical switches.

IV. ACKNOWLEDGEMENTS

REFERENCES

[1] C. S. Chang, Y. T. Chen, and D. S. Lee. Constructions of optical fifo queues. *IEEE Transactions on Information Theory*, 52:2838–2843, June 2006.
[2] 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*, 50:3221–3233, 2004.

[3] R. L. Cruz and J. T. Tsai. Cod: alternative architectures for high speed packet switching. *IEEE/ACM Transactions on Networking*, 4:11–20, February 1996.

[4] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part iii: Routers with very small buffers. *ACM/SIGCOMM Computer Communication Review*, 35(3):83–90, July 2005.

[5] D. K. Hunter, M. C. Chia, and I. Andonovic. Buffering in optical packet switches. *Journal of Lightwave Technol*, 16:2081–2094, December 1998.

[6] H. Park, E. F. Burmeister, S. Bjorlin, and J. E. Bowers. 40-gb/s optical buffer design and simulations. In *Numerical Simulation of Optoelectronic Devices (NUSOD)*, August 2004.

[7] G. Raina, D. Towsley, and D. Wischik. Part ii: Control theory for buffer sizing. *ACM/SIGCOMM Computer Communication Review*, 35(3):79–82, July 2005.

[8] A. D. 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.

APPENDIX

We show that under scheduling policy A, the three following properties hold.

(i) Occupancy of W is always smaller than $N/2$.

An arriving packet p must be kept in W until its preceding packet -in departing order- is transferred to the tail of some delay line. Packet p is then scheduled to depart W in the following time-slot. According to this policy, if W is empty when packet p arrives, i.e., the preceding packet is in one of the delay lines, then the maximum waiting time would be $N/2-1$, which corresponds to longest delay line. During this waiting time, all the arriving packets which are admitted to the system according to step 1 of Algorithm A, will be kept in W. But the policy also implies that when the waiting time goes to 0, packet p and all other packets in W will be sent to the delay lines consecutively, one in each time slot, until the waiting line gets empty. Therefore, the occupancy of W never exceeds $N/2-1$.

(ii) There is no contention among head of delay line packets.

Assume that there is no packet in the system at time 0, and that at each time slot $0 \leq \tau \leq t$ at most one packet is scheduled to be switched into delay line D_i, $i=1,...,\log N$. We prove that the same holds at time $t+1$.

To prove this, we first show that if we assume that lemma 2 holds up to time t, then for any two p and p' with successive order, where p' immediately follows p in departing order, the followings also hold for any any time-slot $\tau \leq t$:

(†) p' is either in the same line that contains p or is in a longer line, and

(‡)
$$l'(\tau) - l(\tau) \equiv 1 \mod d(\tau) \tag{1}$$

where $l(\tau)$ and $l'(\tau)$ are the locations of packets p and p' respectively, and $d(\tau)$ is the length of the delay line containing packet p.

The assumption that lemma 2 is valid until time t ensures that scheduling one packet has no effect on other packets, i.e., when a packet is to be transferred to the tail of a delay line, that location is guaranteed to be empty, and hence no packet is dropped, or is prevented from going to its destined location because of another competing packet. According to the scheduling policy of algorithm A, a packet changes line only when its departing order gets smaller than the length of the current line. Therefore, a particular packet only moves upward, from longer delay lines to shorter ones, and hence,

$$
\begin{aligned}
l(\tau) - l(\tau+1) &\equiv 1 \mod d(\tau+1) \\
l'(\tau) - l'(\tau+1) &\equiv 1 \mod d'(\tau+1)
\end{aligned} \tag{2}
$$

When packet p' is transferred to one of the delay lines $\{D_i\}$ from the waiting line, step 1 of algorithm A ensures that both † and ‡ are valid. Now assume that † and ‡ are valid for some $\tau < t$. In the following we show that the same holds at $\tau+1$. If at time τ packet p' is not at the head of some delay line, then clearly † holds at $\tau+1$ too. If at time τ packet p' is at the head of a delay line, then ‡ implies that at τ packet p is located at the tail of some delay line, or equivalently, at $\tau-1$ packet p has been at the head of a delay line. But there can be at most one departure from the system at time τ. As a result, the departing order of packet p' at time τ will be greater than or equal to that of packet p at time $\tau-1$. Therefore, at time $\tau+1$ packet p will be located either in the same line where p is, or in a longer line, and hence, † holds at time $\tau+1$ too. To show that ‡ also holds at $\tau+1$, we need to keep in mind that the length of a delay line is divisible by the length of any shorter delay line. More specifically, $d'(\tau+1)$ is a multiple of $d(\tau+1)$ in Equation 2, and therefore, $l'(\tau+1) - l(\tau+1) \equiv l'(\tau) - l(\tau) \mod d(\tau+1)$. Validity of ‡ at time $\tau+1$ immediately follows from this equation, and the fact that $d(\tau+1)$ is a multiple of $d(\tau)$.

To show that lemma 2 holds at time $t+1$, consider two head-of-line packets a and b at time t, located at the head of two delay lines with length d_a and d_b, where $d_a < d_b$. Denote by π_a and π_b the departing order of packets a and b respectively. Since † holds at time t, we know that $\pi_a < \pi_b$. Now let the location of these packets (enumerated as shown in Fig. 2) be l_a and l_b respectively. By successive application of ‡ we have

$$l_b - l_a \equiv \pi_b - \pi_a \mod d_a \tag{3}$$

But $l_b - l_a \equiv 0 \mod d_a$, and hence, $\pi_b - \pi_a = kd_a$ for some integer $k \geq 1$. The scheduling policy of algorithm A is based on placing the head-of-line packets in the longest line whose length is not greater than the departing order of the packets. Assume packet a is scheduled to be at the tail of a delay line with length d_0 at the next time slot. This implies that $\pi_a \geq d_0$. On the other hand, $\pi_b = \pi_a + kd_a > (k+1)d_0 \geq 2d_0$. Therefore, there is at least one longer delay line that can accommodate packet b without violating the scheduling policy of algorithm A. This shows that none of the head-of-line packets will compete for the same location at time $t+1$.

(iii) Packet with departure order 1 is always at the head of a delay line.

When a packet p with departure order k gets to the head of some delay line, it will be scheduled to be placed in a delay line with length $l \leq k$ at the next time slot. In other words, there will be at most $k-1$ departures from the system before packet p reaches the head of a line again, and hence its departure order can not be 1 unless it is at the head.