Constructing Optical LIFO Buffers of Size B with $3\log_2 B$ Fiber Delay Lines

Xiaoliang Wang, Xiaohong Jiang
Graduate School of Information Sciences
Tohoku University
Sendai, Japan 980-8579
Email: {waxili,jiang}@ecei.tohoku.ac.jp

Achille Pattavina
Department of Electronics and Information
Politecnico di Milano
Milano, Italy, 20133
Email: pattavina@elet.polimi.it

***Abstract*—The lack of optical buffer is still one of the main problems that hinder the development of all optical network. The current works on this topic mainly focus on emulating optical buffers with fiber delay line (FDL). Recent advances have shown the feasibility of emulating many kinds of optical buffers, such as the First In First Out (FIFO) buffer, priority buffer, etc. The Last In First Out (LIFO) buffer is another important network components that can be used for providing QoS guarantee and congestion control. Huang et al. introduced a recursive construction of LIFO buffer with buffer size B by using no less than $9\log_2 B$ FDLs [1]. In this paper, we try to reduce the number of required delay lines there. We will show that by combining every 3 fiber delay lines into one group and applying exponential growth of FDLs among groups, one needs only $3\log_2 B$ delay lines to emulate a LIFO buffers of size B.**

I. Introduction

Optical buffering is fundamental for contention resolution in optical network. However, the lack of good buffering methods in the optical domain is a major impediment. Until now, the fiber delay lines (FDL) seems to be the only feasible way of realizing optical buffers. In the FDL, packet entering the buffer can depart only after a fixed time, i.e., the time packets need to take to traverse the fiber length [2]. As one fiber can only provide a fixed amount of delay, the length of delay lines and the control algorithm should be carefully designed so that the arrival packets can be delivered to their output link at the right time.

The common approach to buffering packets is to route arrival packets through a series of fiber delay lines connected by switch fabrics. Existing works have shown the feasibility of such constructions. A good survey of these early constructions can be found in [3]. Recent studies on the switch and FDL (SDL) based optical buffer designs have shown the possibility of exactly emulating the behaviors of various electronic buffers, see, for example, First In First Out(FIFO) buffers [1], [4]–[6], Last In First Out (LIFO) buffers [1], [7], Push In First Out (PIFO) buffers (or priority buffers) [8]–[10], shared buffers [11] and etc. A survey of the above constructions can be found in [12].

The research works on this line aim at constructing optical buffers with the least complexity and simple control. Most of the former works provide constructions with as small switch size as possible. It is notable, however, that the long lengths of fiber are also heavy cost. For example, an 1-Tb/s router, with link rate of 40Gb/s and 1μm single-wavelength fiber delay line buffers at each port, would have a total of 8,000 km of fiber delay line [13]. This trend may become worse for future higher capacity router applications. Recent researches in slow light technology show potential for optimizing the length of delay line buffers [14]. On the other hand, the systematic researches are required for optimizing the number of delay lines. Both the results in [8] and [10] indicated that to emulate a general buffer with buffer size B, the minimum number of required delay lines is $O(\log B)$.

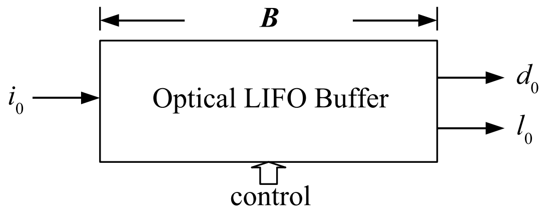

Fig. 1. An optical LIFO buffer with buffer size B

In this paper, we consider the construction of LIFO buffer, a widely used queuing structure for QoS buffer management in packet switching network [15]. The design of optical LIFO buffer can also shed light on exact emulation of the common optical priority buffer. In [1] P. K. Huang et al. introduced a recursive construction of parallel LIFO buffers. It adopts the idea of two-level cashing that directs a block of packets through different lengthed SDL-cashes under properly selected thresholds. As shown in [1], a LIFO buffer of size B actually requires no less than $9\log_2 B$ delay lines. We argue that the required number of FDLs can be reduced according to the following observations: first, the 3×3 switches adopted there only allow three connection patterns, and thus an arriving packet has to be stored in buffer space different from the one being freed by the dumped packet; Second, packets movement between two-level cashes always starts from the header of each block, but actually the movement can start from one packet at any position of one block so that buffer line is freed as soon as possible. The motivation of this paper is to reduce the number of delay lines in single LIFO buffer by replacing the multiple

3×3 switches with one switch, which can realize large scale permutation and simplify the scheduling algorithm. Our results show that it is possible to construct a single LIFO buffer with only $3\log_2 B$ delay lines which is much less than the cost of LIFO buffer in [1] (we note that the constructions in [1] are more general and can work for multiple LIFO queues).

II. The structure of LIFO buffers

In this section we introduce the definition of optical LIFO buffer, and explore its construction from using the feedback switch architectures.

A. Preliminaries

To simplify the design and operation of optical switches, we assume that the time of system is sliced and synchronized. Additionally, without loss of generality, we further assume that the packet size is fixed, a packet can be transmitted within one time slot, and the length of a delay line is equal to an integer number of time slots.

Definition 1: **(Exact emulation)** An optical buffer exactly emulates its electronic counterpart if with identical arrival packets and identical departure requests to both the optical and electronic systems, the output and the drop behaviors of the optical buffer is the same as that of the electronic one.

Definition 2: **(Optical LIFO buffer)** An optical LIFO buffer of buffer size B is a network element with one arrival link (i_0), one departure link (d_0) and one lost link (l_0) that can store up to B packets, as illustrated in Fig. 1. The optical buffer can exactly emulate an electronic LIFO buffer of the same buffer size, i.e., when a departure request comes, the departing packet is always the latest arrived packet among all the packets in the buffer; if the number of buffered packets is equal to B, the buffer is full and the newly arrived packet will be dropped before entering the buffer.

Based on these assumptions and definitions, we then explore the SDL based construction of optical LIFO buffer.

B. Architecture

Here, we take into account the commonly used feedback architecture. As illustrated in Fig. 2, this architecture consists of an $(M+1)\times(M+1)$ switch fabric, one input port, one output port, and M delay lines connecting M outputs back to M inputs of the switch fabric. The length of i^{th} delay line is denoted as r_i. An 1×2 switch is set in front of the input port for access control. If B packets have already been buffered in the system and there is no departure request, the newly arrived packet will be dropped directly to the lost link via the 1×2 switch. In the following we will focus on the length setting of M delay lines and the corresponding packet scheduling algorithm such that the structure in Fig. 2 can work as a LIFO buffer.

For the sack of presentation, the packets are assigned with descending priorities according to their arrival order. More precisely, the newly arrived packet is always assigned with the highest priority 1. If a new packet comes to the switch, the priority of all the buffered packets will be increased by one.

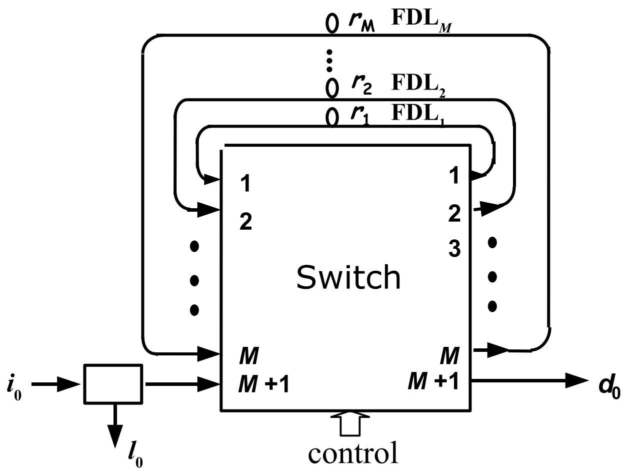

Fig. 2. A feedback structure of optical LIFO buffer

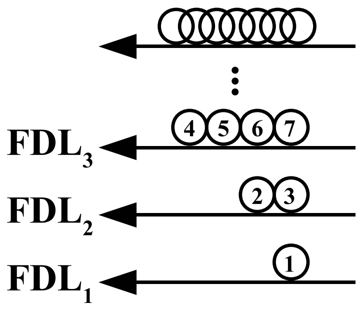

Fig. 3. The exponential growth of fiber delay lines

Similarly, after a packet departs from the system, the priority of packets buffered in FDLs will be decreased by one. To assure that the packet with priority 1 is always available at each time slot, the packet scheduling should satisfy the following rule:

(R1) the packet with priority p is always switched to a delay line with length no longer than p.

Otherwise, an extra departure delay will be introduced.

According to the result established in [8], we know that at least $M=\lceil\log_2 B\rceil$ delay lines are required to accommodate B packets in the exact emulation of a FDL based buffer. Following this result, the length of delay lines that serves as buffer are of exponential growth, i.e., the delay lengths will be 1, 2, 4, 8, ..., as illustrated in Fig. 3. The number there shows the priority of buffered packets, where the packet with priority p will be placed in the $(\lfloor\log_2 p\rfloor+1)^{th}$ delay line. Someone may wonder if it is possible to build a LIFO buffer with just $\lceil\log_2 B\rceil$ delay lines. Through an example, we will show that it is impossible to build a LIFO buffer of size B by using only $\lceil\log_2 B\rceil$ delay lines.

Example 1: Let us consider the simplest construction where $M=2$, i.e., there are only two delay lines and their lengths are 1 and 2 separately. Assume that this construction starts from an empty system and no departure request comes from time $t=0$ to $t=3$. As illustrated in Fig. 4, one packet arrives at

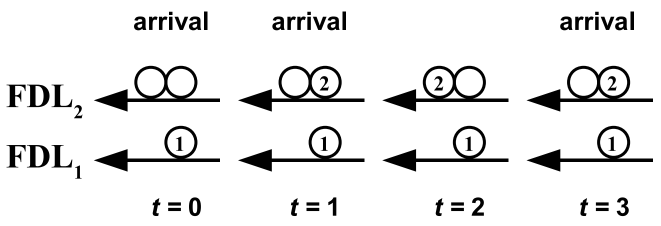

Fig. 4. A counter example

time 0, and according to (R1) it is delayed in FDL_1 whose length is one time slot. At the begining of time slot $t = 1$, another packet arrives and the former packet is also available at the output of FDL_1. After revising their priorities, the newly arrived packet with priority 1 is delivered to FDL_1, and the packet with priority 2 can be delivered to FDL_2 whose length is 2. If no packet arrives at $t = 2$, the packet with priority 1 will circulate in FDL_1 for one more time slot. At the beginning of time slot $t = 3$, these two packets with priorities 1 and 2 emerge from the outputs of delay lines. If a new packet arrives simultaneously, the problem happens as that there are only two of these three packets can be accepted in two delay lines at this time. One packet has to be dropped although the total length of delay lines is 3 there.

This observation leads to the requirement of introducing additional fiber delay lines, which can delay packets for a while to adjust the time of packets to be inserted into the fiber delay lines serving as buffer. In this paper, only $3 \log_2 B$ delay lines are applied. More explicitly, our setting of FDLs is illustrated in Fig. 5, where all the M FDLs are separated into $K = \lceil M/3 \rceil$ groups. Except the last group, each of these groups contains three FDLs. The length of FDLs in the k^{th} group G_k are all equal to 2^{k-1}, $k \in [1, K]$. In the next section, we will show that a LIFO buffer of size $B = \sum_{i=1}^{M} r_i$ can be emulated by properly scheduling packets under the setting of FDLs in Fig. 5.

III. Packet Scheduling

The main idea of our packet scheduling algorithm is based on the fact that if all the packets arrive back-to-back, the packets with successive priorities will be placed consecutively, as shown in Fig. 3. Such consecutive assignment of packet position is also applied in this paper, since by placing the emerged packet (the head-of-line packet) in the longest line with length smaller than or equal to its priority all the buffered packets can orderly depart from the output port without any void [6]. The challenge of the LIFO buffer design is that the priorities of all the packets should be changed whenever there is an arrival or a departure. Therefore, the problem we need to solve is how to schedule the head-of-line packet so that the packets with successive priorities are stored in the delay lines consecutively.

As exponential growth of fiber delay lines is adopted, the delay length of fibers in one group is always integer time than that of fibers in group with lower index. To guarantee

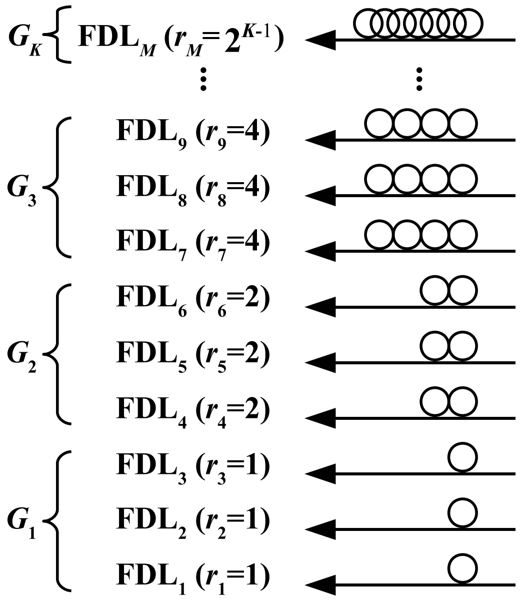
G_K
FDL_M (r_M=2^{K-1})
G_3
FDL_9 (r_9=4)
FDL_8 (r_8=4)
FDL_7 (r_7=4)
G_2
FDL_6 (r_6=2)
FDL_5 (r_5=2)
FDL_4 (r_4=2)
G_1
FDL_3 (r_3=1)
FDL_2 (r_2=1)
FDL_1 (r_1=1)

Fig. 5. The setting of delay lines in our construction

packets consecutive, we order packets movement is in block between groups. By doing so, the delay lines can be dumped or retrieved in consecutive times. Thus, the notation of *frame* is introduced here, which is the granularity for packet scheduling in different groups. The frame in G_k contains 2^{k-1} packets, that is equal to the capacity of one FDL in G_k and is also equal to the total capacities of two delay lines in G_{k-1}. Specifically, once a packet is delivered to another group, all the other packets belonging to the same frame will follow this operation too. Clearly, it requires consecutive b time slots to deliver a frame of size b.

For one group, the arrival frames are always placed in the idle FDLs until all the delay lines within this group are occupied. Then we say this group is full. On the contrary, we say a group is idle when all the FDLs in this group have no frame. If a new frame is delivered to a full group, the two frames which contain lower priority packets within this group will be combined together to form a new frame. And then this newly formed frame is immediately sent to the upper group so that the newly arrived frame can be accepted to the idle FDLs. It is notable that a frame departure from one group only happens when its lower group becomes idle. Especially, the departure frame is the frame which contains the highest priority packets within current group.

For ease of comprehension, we give an example of $M = 7$, $K = 3$ as follows,

Example 2: Assume the system works well until the end of time T, the occupation state of FDLs is shown in Fig. 6 (a). Obviously, the G_1 is full at this time. At the beginning of time slot $T + 1$, the packets with priority 1, 2, 3, 5 and 7 become head-of-line packets, and we assume a new packet arrives at this time slot. If no departure request, the priority of all the buffered packets will be increased by one, as shown in Fig. 6 (b). The newly arrived packet is going to be sent to

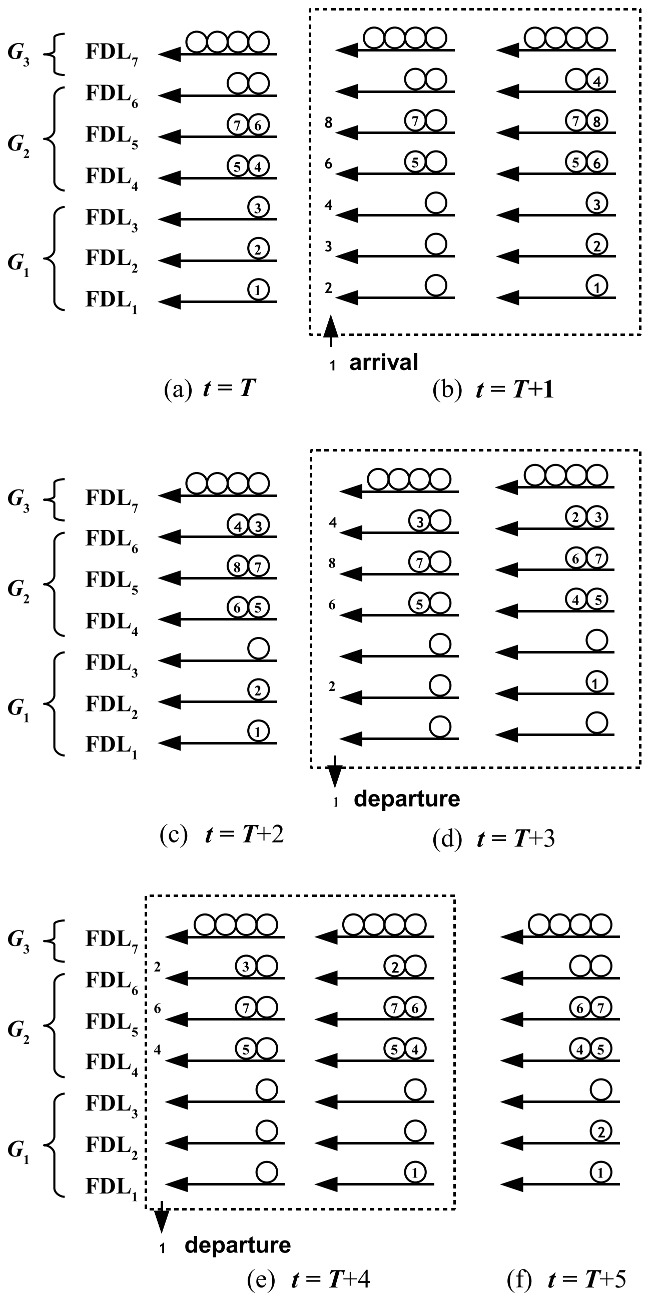

Fig. 6. An example of packet scheduling

the delay line with length 1, i.e., a new frame is delivered to a full group G_1. Thus, the packets with priority 3 and 4 are combined together to form a larger frame and will be sent to G_2. To keep packets in delay lines consecutively, the packet with priority 4 is firstly moved to the idle FDL$_6$ and packet with priority 3 will be moved to FDL$_6$ at the next time slot ($T + 2$). At the end of time slot $T + 1$, the packets with priority 1, 2, 3, 6, 8 are sent to FDL$_1$, FDL$_2$, FDL$_3$, FDL$_4$ and FDL$_5$, respectively. It is notable that the FDL$_3$ actually works as an adjuster at this time. If a departure request comes but no arrival at time slot $T + 3$, the packet with the highest priority is moved out and the priority of all the remaining packets will be reduced by one, as illustrated in Fig. 6 (d). Suppose another departure request comes at time slot $T + 4$ but no arrival, the packet with the highest priority leaves from the output port. Since G_1 becomes idle, one frame will be moved from G_2 to G_1, which is the frame containing two highest priority packets. Thus, after revising the priority of all the packets, the emerged packet with priority 1 now is sent to a FDL in G_1. At time slot $T + 5$, the packet with priority 2 is sent to another idle FDL in G_1, as that shown in Fig. 6 (f). It is notable that if one packet arrives, we still have an idle FDL to accommodate the new comer.

More formally, the scheduling algorithm is summarized as follows:

Scheduling Algorithm

```
(1) arrival
      **if** buffer is full **then**
        drop the newly arrived packet.
      **else**
        increase the priority of all buffered packets by 1,
        assign priority 1 to the newly arrived packet and
        insert it to FDL$_1$.

(2) departure
      remove the packet with priority 1, and decrease the
      priority of remaining packets by 1.

(3) scheduling of head-of-line packets
    **for** $k = 1$ **to** $K$
      **if** $G_k$ is full **then**
        **if** a new frame comes **then**
          combine the two frames which contain lower
          priorities packets in $G_k$ together to form a new
          frame, send the newly formed frame to $G_{k+1}$,
          accept the new frame to idle FDLs.
      **else if** $G_k$ is idle **then**
        **if** a departure request comes **then**
          send the frame containing the highest priority
          packet of G$_k$ to G$_{k-1}$
        **if** all the FDLs are idle **then**
          send a departure request to $G_{k+1}$.
      **else**
        **if** a new frame comes **then**
          accept the new frame to idle FDLs.
        **if** a departure request comes **then**
          send the frame containing the highest priority
          packet in G$_k$ to G$_{k-1}$
```

Now, we can establish the following theorem.

Theorem 1: For the feedback construction in Fig. 2 with above scheduling algorithm, if the length of i^{th} delay line is set as $r_i = 2^{\lfloor (i-1)/3 \rfloor}$ for $i = 1, \ldots, M$, it can exactly emulate a LIFO buffer of size $B = \sum_{i=1}^{M} r_i$.

Proof: If the scheduling rule (R1) is satisfied, a packet can never be switched to a delay line longer than its value of priority. Therefore, the packet will emerge from FDL earlier than or at the right time. If the emerged packet is the packet with the highest priority, it can be sent out from output port through switch fabric. Otherwise, following (R1), this packet will be sent back to a delay line no longer than the value of its priority until its turn to depart. Thus, the key point of our proof is to explain that the scheduling rule (R1) is guaranteed under the above scheduling algorithm.

If a frame arrives to G_k from G_{k-1}, $k \in [1, \lceil M/3 \rceil]$, we can deduce from the above algorithm that G_{k-1} is full at the beginning of this time slot and a new frame arrives to G_{k-1}. Similarly, we can also know that the G_{k-2} is full and a new frame comes as well at the beginning of the same time slot, since the newly arrived frame in G_{k-1} must come from group G_{k-2}. Following this idea, after scheduling, at least one delay line is full within each of the $k-1$ groups with lower indexes. Thus, the priority of arrival packet in G_k is larger than $\sum_{i=1}^{k-1} 2^{i-1} = 2^{k-1} - 1$. Since the length of delay lines in G_k is 2^{k-1}, the arrival packet will be sent to the delay line whose length is equal to or smaller than the its priority, which means the (R1) is satisfied. For the case that a frame departs from G_{k+1} and is buffered in G_k, we have the conclusion that none of the groups with lower indexes than k is idle, otherwise this frame can not be buffered in G_k but go to group with lower index. Thus, the priority of this dumping packet is equal or larger than $\sum_{i=1}^{k-1} 2^{i-1} + 1 = 2^{k-1}$ which is the length of FDLs in G_k.

Notice that the arrival packet is always sent to the idle delay lines G_1 first until G_1 becomes full. Similarly, the arrival frame is always sent to the idle FDLs in one group until this group becomes full. Thus, all the delay lines can served as buffer, and the construction can accommodate $\sum_{i=1}^{M} r_i$ packets. ■

IV. Conclusion and Future Works

In this paper, we focus on the feedback construction of optical LIFO buffer by using switched fiber delay lines. The considered system consists of an $(M+1) \times (M+1)$ crossbar switch and M fiber delay lines connecting the M outputs of the crossbar switch to its M inputs. We show that under the setting of delay lines in Fig. 5, and by applying the corresponding scheduling algorithm, the feedback construction can really emulate a LIFO buffer with buffer size B with only $3\log_2 B$ delay lines.

A more challenging but sensible work is the optimal construction of optical priority buffer (named Push in First Out buffer in [10]). The priority buffer is a common model which covers the FIFO buffer and LIFO buffer as special cases. The issue of exact emulation of priority buffers has been addressed in [8]–[10], [16]. Our works can also serve as a foundation for the optimal design of priority buffers.

It is notable that the studies of SDL based buffer emulation in this paper and most former papers only pay attention to the feasibility of optical buffers but did not consider the signal attenuation or spontaneous emission noise problems. This makes the SDL based optical buffer still far from practical. In the future, the practical design of SDL based buffer deserves deliberate studies.

References

[1] P. K. Huang, C. S. Chang, J. Cheng, and D. S. Lee, “Recursive constructions of parallel fifo and lifo queues with switched delay lines,” *IEEE Transactions on Information Theory*, vol. 53, pp. 1778–1789, 2007.

[2] *Optical networks : a practical perspective, 2nd Edition.* Morgan Kaufmann, 2002.

[3] D. K. Hunter, M. C. Chia, and I. Andonovic, “Buffering in optical packet switches,” *Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, December 1998.

[4] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, “Slob: a switch with large optical buffers for packet switching,” *Journal of Lightwave Technology*, vol. 16, no. 10, pp. 1725–1736, October 1998.

[5] C. S. Chang, Y. T. Chen, and D.-S. Lee, “Constructions of optical fifo queues,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2838–2843, 2006.

[6] N. Beheshti and Y. Ganjali, “Packet scheduling in optical fifo buffers,” in *High-Speed Networking Workshop*, Anchorage, Alaska, May 2007, pp. 63–66.

[7] B. A. Small, A. Shacham, and K. Bergman, “A modular, scalable, extensible, and transparent optical packet buffer,” *Journal of Lightwave Technology*, vol. 25, no. 4, pp. 978–985, April 2007.

[8] A. D. Sarwate and V. Anantharam, “Exact emulation of a priority queue with a switch and delay lines,,” *Queueing Systems: Theory and Applications*, vol. 53, pp. 115–125, July 2006.

[9] H. C. Chiu, C. S. Chang, J. Cheng, and D. S. Lee, “A simple proof for the constructions of optical priority queues,” *Queueing Systems: Theory and Applications*, vol. 56, pp. 73–77, 2007.

[10] H. Kogan and I. Keslassy, “Optimal-complexity optical router,” in *IEEE INFOCOM 2007*, Alaska AK, May 2007.

[11] X. Wang, X. Jiang, A. Pattavina, and S. Horiguchi, “A construction of 1-to-2 shared optical buffer queue with switched delay lines,” accepted by IEEE Transactions on Communicaitons. [Online]. Available: http://www.hori.ecei.tohoku.ac.jp/ waxili/hpsr08j.pdf

[12] X. Wang, X. Jiang, and S. Horiguchi, “Constructing optical buffers with switches and fiber delay lines,” in *The 15th Asia-Pacific Conference on Communications (APCC2009)*, Shanghai, China, Oct 2009.

[13] R. Tucker, “Petabit-per-second routers: optical vs. electronic implementations,” in *Optical Fiber Communication Conference, 2006 and the 2006 National Fiber Optic Engineers Conference. OFC 2006*, March 2006, p. OFJ3.

[14] P.-C. Ku, C. J. Chang-Hasnain, J. Kim, and S.-L. Chuang, “Variable optical buffer using slow light in semiconductor nanostructures,” in *Proceedings of the SPIE*, vol. 5362, March 2004, pp. 69–80.

[15] A.-M. Lin and J. Silvester, “Priority queueing strategies and buffer allocation protocols for traffic control at an atm integrated broadband switching system,” *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 9, pp. 1524–1536, Dec 1991.

[16] H. C. Chiu, C. S. Chang, J. Cheng, and D. S. Lee, “Using a single switch with $o(m)$ inputs/outputs for the construction of an optical priority queue with $o(m^3)$ buffer,” in *IEEE INFOCOM minisymposium 2007*, 2007.