Parallel and Pipeline Processing for Output-Buffer Management in Photonic Packet Switches

Hiroaki Harai[†] **and Masayuki Murata**[‡]

[†] Communications Research Laboratory, Tokyo, Japan

[‡] Osaka University, Osaka, Japan

Abstract

We investigate the mechanism of high-speed buffer management for output-buffered photonic packet switches. We propose a buffer management mechanism on parallel and pipeline processing architecture consisting of $(\log_2 N + 1)$ pipeline stages, where N is the number of ports of the packet switch. This is an expansion of a simple round-robin scheduling for determining the delays of arriving packets. The pipeline stages consist of a prefix operation part and a delay determination part. The prefix operation part determines the relative delays of arriving packets, assuming that no packet is currently buffered and that all the arriving packets will be buffered based on round-robin scheduling. We achieve speedup by a parallel-prefix operation in this part. The delay determination part gives the delays of packets on all the N ports simultaneously, by using the current status of buffer occupancy and the relative delays. Since the time complexity of each processor in the pipeline stages is $O(1)$, the throughput of the buffer management is N times larger than that of the round-robin scheduling method. We apply the processing and the architecture to buffer management for asynchronously arriving variable-length packets. We show the feasibility of a buffer manager supporting 128×128 photonic packet switches with 40Gbps ports, which provide at least ten times as much throughput as the latest electronic IP routers. The proposed mechanism for asynchronous packets overestimates the buffer occupancy to enable parallel processing. We show that the degradation in the performance of the mechanism resulting from this overestimation is quite acceptable through simulation experiments.

All correspondence should be directed to:

Dr. Hiroaki Harai	
Address:	Network Architecture Group
	Communications Research Laboratory
	Koganei-shi, Tokyo 184-8795, Japan
Tel:	+81-42-327-5418
FAX:	+81-42-327-6680
E-mail:	harai@crl.go.jp
Note:	Communications Research Laboratory will be reorganized as
	“National Institute of Information and Communications Technology”
	on April 1st, 2004.

1 Introduction

To build a core network capable of handling a tremendous amount of traffic on the Internet, it is necessary to improve the node throughput, as well as to increase the link capacity. Currently, we rely on electronic processing for packet forwarding at nodes such as routers. The node throughput can be improved through advances in LSI technology and via large-scale distributed or pipelined processing. While the link capacity can be easily increased by bundling optical fibers, the integration and pipelined processing are likely to limit increases in the node throughput. One way to increase the node throughput is to introduce a closed domain that is based on a new, lower layer photonic technology. A promising approach is to use generalized multi-protocol label switching

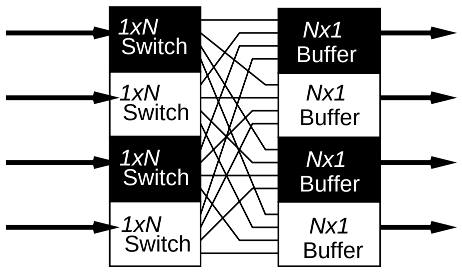

Figure 1: An $N \times N$ photonic packet switch architecture ($N = 4$)

(GMPLS) [1] over a lightpath topology [2]. However, this approach has problems with complex traffic engineering and coarse capacity granularity. We thus introduce optical technology into packet switching directly instead of using the circuit-switching based on lightpath topology.

There are two ways of achieving high-throughput photonic packet switches: increasing the number of the state-of-the-art line speed ports (i.e., interfaces), or, increasing the line speed with the same number of ports. To our knowledge, the latest electronic IP routers are capable of 48×10Gbps ports [3], which is almost as much as electronic processing can handle. In this paper, aiming at applying backbone networks, we investigate a buffer management mechanism for photonic packet switches supporting 40Gbps ports, which have already been demonstrated experimentally [4, 5], and find the number of ports supported by our buffer management mechanism. We thus prove that photonic packet switches take advantage over the electronic ones.

For that purpose, we look at the packet switching more closely. The functions of photonic packet switches are roughly divided into five groups: label lookup (i.e., forwarding), switching, buffer management (i.e., scheduling or queue management), buffering, and routing. To transfer very high-speed data at rates such as 40 and 160Gbps without O/E/O conversion, switching and buffering must be handled in the optical domain. In transferring an extremely large number of packets in a short period of time, however, the process of accessing electronic memory for label lookup can easily become a bottleneck in packet forwarding. It is therefore desirable for label lookup to be processed optically. Label lookup (multi-wavelength label analysis [6] and optical code label analysis [7]), switching [8-10], and buffering [9, 11] can be handled in the optical domain. A photonic packet switch prototype that has those optical functions is developed [4]. Packet buffering itself can be done in the optical domain using an optical fiber-delay-line (FDL) buffer (see, e.g., [4, 9]). However, managing the optical buffer still requires electronic processing, which involves calculation of the delays in the optical buffer and determination of the FDL to which the packet is directed [12]. By limiting the area in which electronic processing is used, we can achieve high-performance packet switches. It is therefore important to use a buffer management mechanism with less time complexity to avoid degradation in the performance of photonic packet switches.

In this paper, we focus on output-buffered $N \times N$ photonic packet switches. The output-buffer architecture provides better delay and/or throughput performance than the input-buffer architecture because it does not suffer from head-of-line (HOL) blocking [13]. On the other hand, its implementation is difficult because it requires N times faster switching fabric than the input-buffer architecture. Multiple-input-queue (MIQ) is a feasible solution to solving the problem of HOL blocking and it achieves the same theoretical performance as the output-buffer architecture. However, the photonic packet switch on which we focus includes N switches of size $1 \times N$, as shown in Fig. 1. It gives an N times greater number of lines to each output port, which provides functions similar to the faster switching fabric. Moreover, MIQ requires central arbitration (i.e., a buffer management method for MIQ) to avoid packet contention at the output ports [14, 15]. The arbitration mechanism assumes the existence of a RAM buffer at the input ports. The implementation of an input optical buffer is very difficult because it requires an optical RAM that is still in the process of being developed. Henceforth, we focus on a direct implementation of the output-buffer architecture rather than the MIQ architecture.

Since we use an optical FDL buffer to avoid packet contention, we must determine the delays of packets in the buffer before the packets moving through the interconnected fiber between the optical switch and the buffer actually reach the buffer. If we employ round robin scheduling, which is a conservative $O(N)$ approach, we must manage the buffer in the electronic domain N times faster than the port speed normalized by the minimum packet length. This means that we would have to face a scalability problem as the number of ports increases. In spite of the need to avoid bottlenecks resulting from electronic processing, no studies have looked at the problem of speeding up FDL buffer management from the viewpoint of algorithms.

Much effort has been done to speed up buffer management of electronic packet switches such as IP routers and asynchronous transfer mode (ATM) switches. There are various speedup mechanisms based on parallel and pipeline processing. The traditional single-processor model has been developed into a multi-processor model for high-end IP routers: label lookup is performed at the processor at each port and then buffer management is handled at another processor. A typical example is an approach by MIQ and arbitration [14, 15]. The speeding up of arbitration is also described [14]. The parallel and pipeline processing requires increasing the number of processors or the size of LSI circuitry. However, it offers the advantage of increased node throughput. Advances in LSI technology including the development of field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) enable achieving a high node throughput with parallel and pipeline processing. The problem is that MIQ is not tractable in optical FDL buffer management.

Prakash *et al.* [16] proposed an output-buffer management mechanism with a time complexity of $O((\log N)^2)$. The complexity results from the storing of packets into multiple memories with a constraint that multiple packets must not be read from one memory at the same time. To calculate the time at which the output port becomes idle, the authors used a parallel prefix-sum operation [17, 18], where the time complexity is $O(\log N)$ using N processors. We can use this parallel operation to speed up of FDL buffer management.

In this paper, we investigate a high-speed buffer management mechanism for output-buffered photonic packet switches. We propose a buffer management mechanism on parallel and pipeline processing architecture consisting of $(\log_2 N + 1)$ pipeline stages. This is an expansion of a simple round-robin scheduling for determining the delays of arriving packets. The pipeline stages consist of a prefix operation part and a delay determination part. The prefix operation part determines the relative delays of arriving packets, assuming that no packet is currently buffered and that all the arriving packets will be buffered based on round-robin scheduling. We achieve the speeding up of the prefix operation by using an $O(N \log N)$ number of processors, each of which is devoted to one part of the parallel-prefix operation [17, 18]. The delay determination part gives the delays of packets on all the N ports simultaneously, by using the current status of buffer occupancy and the relative delays sent from the prefix operation part. Information about the buffer occupancy is updated in parallel with giving the delays in this part. Since the time complexity of each processor in the pipeline stages is $O(1)$, the throughput of the buffer management is N times larger than that of the round-robin scheduling method. We first describe the multi-processing architecture and the mechanism for a buffer management for handling synchronously arriving fixed-length packets.

In the mechanism described in Ref. [16] and in the above-mentioned parallel and pipeline mechanism, the buffer manager is only capable of handling fixed-length packets. However, the length of packets is diverse on the Internet [19]. If we use the buffer management, we need additional fragmentation from variable-length packets into multiple fixed-length packets at edge nodes of a closed-domain network or at each packet switch. We also need optical synchronization [20, 21] of the fixed-length packets, which makes the optical system more complex. To alleviate the burden of such complex processing and to enable of future deployment of photonic packet switches in the Internet infrastructure, it is desirable to adopt a mechanism handling asynchronously arriving variable-length packets. A number of buffer management schemes have been developed for photonic packet switches to support asynchronously arriving variable-length packets [21-23]. However, their time complexities are equal to or greater than $O(N)$.

We also extend the parallel and pipeline processing for handling asynchronously arriving variable-length packets. We confirm the feasibility of its implementation because our mechanism uses a pipeline processing, and its space complexity is $O(N \log N)$. We expect that the FDL buffer architecture, which does not require large memory, can solve the problem of the feasibility more easily than the electronic memory buffer architecture. Through hardware simulation after place-and-route operation, we confirm the feasibility of the FPGA-based buffer manager for 8×8 photonic packet switches with 40Gbps ports, which are capable of handling variable-length packets with a minimum size of 64bytes. We also show the feasibility of a buffer manager supporting 128×128 photonic packet switches with 40Gbps ports by using the latest FPGAs. A photonic packet switch provides at least ten times as much throughput as the latest electronic IP routers. The proposed mechanism for the asynchronous packets overestimates buffer occupancy to enable parallel processing for delay determination and updates of the buffer occupancy at the delay determination part. We show that the performance degradation resulting from the overestimation is quite permissible through simulation experiments.

This paper is organized as follows. In Section 2, we briefly describe the photonic packet switch architecture. The next two sections are devoted to description of our proposed mechanisms. In Section 3, we propose a parallel and pipeline processing mechanism for output-buffer management in photonic packet switches. The packets are of fixed length and they arrive at the packet switch synchronously. In Section 4, we extend the mechanism to support asynchronously arriving variable-length packets. In Section 5, we show the feasibility of our buffer management mechanism through hardware simulation. In Section 6, we show performance of the buffer management mechanism in terms of delay and packet loss probability through network simulation. Our conclusions are presented in Section 7.

2 Overview of Photonic Packet Switch Architecture

2.1 Output Buffer Architecture

Several photonic packet switch architectures have been proposed including an input-buffer architecture [8], a recursive buffer architecture [8], and an output-buffer architecture [4, 9, 21]. We use the output-buffer architecture without wavelength conversion. Our packet switch is similar to that described in Ref. [4] although the buffer management mechanism of the former supports asynchronous, variable-length packets while that of the latter does not.

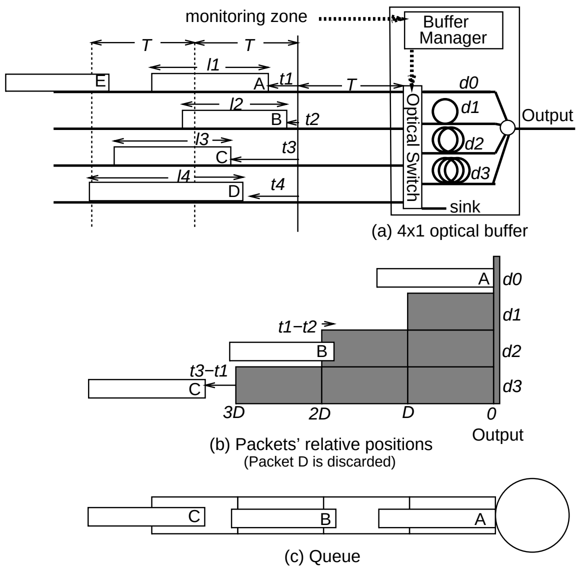

Figure 2: Fundamental behavior for packets that arrive at optical buffer. (a) Optical FDL buffer ($N = 4, B = 4$) and packet arrivals (b) Assigned FDLs and relative position (c) Logical packet position

Recall Fig. 1, which shows a photonic packet switch architecture for the use of our buffer management scheme. The output-buffered $N \times N$ packet switch consists of N bufferless packet switches of size $1 \times N$, followed by N buffers of size $N \times 1$. All of the $1 \times N$ switches and the $N \times 1$ buffers are optically interconnected in a fully meshed manner. The $1 \times N$ bufferless packet switches make the label lookup operation faster by providing photonic label lookup functions [6, 7] to the packet switch. They can handle the switching of asynchronously arriving variable-length packets with precedent activity [24]. The architecture provides ultra-high node throughput to the packet switch. The $N \times 1$ buffers are used to avoid packet collision and reduce packet loss probability.

We use FDLs to compose an $N \times 1$ optical buffer. The 4×1 optical buffer architecture is shown in Fig. 2(a). Each buffer consists of B FDLs ($B = 4$ in this example), an $N \times (B + 1)$ optical switch, a coupler, and a buffer manager. The lengths of the B FDLs, $\{d_0, d_1, \ldots, d_{B-1}\}$, are multiples of unit length D ($0, D, \ldots, (B-1)D$, respectively). The buffer provides discrete-time delays from 0 to $(B-1)D$.

2.2 Fundamental Behavior of Buffer Manager

Unlike RAM buffers for electronic node systems such as IP routers, the FDL buffer with its optical straightforwardness property does not allow for the storing of optical packets in the buffer. The FDL buffer only provides optical packets different delays to avoid packet contention. A destination FDL must be selected for each arriving packet before it arrives at the FDL buffer. In an $N \times N$ output-buffered photonic packet switch, up to N packets arrive at an $N \times 1$ optical buffer simultaneously. When we use FDLs, we must implement a buffer management system to calculate the delays of N packets within time $l_{\min}$, which corresponds to the minimum packet length.

To better understand FDL buffer management described in the following sections, we first describe the behavior of buffer manager using a sequential scheduling mechanism (i.e., round robin scheduling), which is the basis of our parallel and pipeline processing. Recall Fig. 2, which illustrates packets arriving at an optical buffer

```
for $n := 1$ to $N$ do
begin
    if ($l_n \neq 0$) then begin
        $\Delta_n := \left\lceil \frac{q-t_n}{D} \right\rceil$;
        if $\Delta_n < B$ then begin
            $q := \Delta_n D + t_n + l_n$;
            Packet $n$ is given delay $\Delta_n D$; end
        else Packet $n$ is discarded;
    end
end
$q := \max(q - T, 0)$;
```

Figure 3: Pseudo-code for sequential scheduling at an N-port packet switch

of a 4×4 photonic packet switch. The buffer manager has an internal clock (with a frequency of $1/T$) that represents the units of time for sequential scheduling. In each time cycle kT $(k = 1, 2, \ldots)$, the buffer manager receives information about the arrival of up to $N = 4$ packets that will arrive at the input ports of the optical switch in the optical buffer in the next cycle (i.e., during $[(k+1)T, (k+2)T)$). It determines the delay for each arriving packet within time T. To handle arriving packets continuously, each time T cycle must be less than $l_{\min}$ $(T \leq l_{\min})$. This constraint is necessary to prevent packets from arriving at the optical buffer before the delay for each packet has been determined.

We now describe the behavior of the buffer manager in more detail. Let l_n denote the length of a packet observed at port n, and let t_n denote the time difference between the starting time of a cycle and packet arrival. Hereafter we will call this time difference an "arrival gap" (see Fig. 2(a)). The buffer manager receives the length of each packet and the arrival gap as arriving information of a packet. It maintains variable q, which represents the time at which all packets stored in the buffer depart, and the buffer becomes idle. This time is defined relative to the starting time of the targeted cycle. Hereafter, we will use this q to denote buffer occupancy. The buffer manager calculates the delays for new packets coming from all ports during each cycle time T cycle based on round robin scheduling at ports $1, 2, \ldots, N$. Theoretically, $q - t_n$ is a sufficient delay for each packet to avoid packet collision. Unfortunately, due to the discrete-time nature of the FDL buffer, the delay given to one packet must be $\Delta_n D$, where $\Delta_n = \lceil \frac{q-t_n}{D} \rceil$. A packet enters delay line d_{Δ_n} if $\Delta_n < B$, and it is discarded if $\Delta_n \geq B$. When the packet enters the delay line, the buffer occupancy, q, changes to $q \leftarrow \Delta_n D + t_n + l_n$ to properly handle packets at the subsequent ports. After calculating the packet delays for all ports, q is changed to $q \leftarrow \max(q - T, 0)$ to provide appropriate buffer management during the next cycle. We show the pseudo-code for this scheduling in Fig. 3, where "packet n is given delay $\Delta_n D$"means that the corresponding packet is switched to delay line d_{Δ_n}. Since up to N packets arriving at N ports must be handled sequentially in a cycle, the time complexity is $O(N)$.

Assume that four packets, A through D, arrive at the buffer during the same cycle, and that one packet, E, arrives during the next cycle as shown in Fig. 2(a). The buffer manager determines that packets A, B, and C will be switched to delay lines d_0, d_2, and d_3, respectively, and that packet D will be discarded due to buffer overflow (i.e., it is switched to the sink). The buffering and discarding functions are performed by driving the optical switch. Figure 2(b) shows the assigned FDLs and the relative position of the packets from the output port just after packet A has been switched to delay line d_0. As illustrated in the figure, the three packets depart from the

```
for $n := 1$ to $N$ do
begin
   if ($l_n = 1$) then begin
      if $q < B$ then begin
         Packet $n$ is given delay $qD$;
         $q := q + 1$; end
      else Packet $n$ is discarded;
   end
end
$q := \max(q-1, 0)$;
```

Figure 4: Pseudo-code for sequential scheduling of synchronously arriving, fixed-length packets

buffer without collision. Figure 2(c) shows logical packet position in the FDL buffer when packets are assigned as shown in Fig. 2(b). As can be seen in the figure, there are void spaces between two continuous packets. Due to the discrete-time nature of the FDL buffer, there is a space of $\theta = \Delta_n D - q + t_n$ in the optical buffer between a new packet and the previously arrived, adjacent packet. We can use a "void filling" approach [22] to improve the buffer utilization. However, the time complexity of this approach limits the speed of the ports of the packet switch, because it is greater than that in the sequential approach. Moreover, void filling may cause inconsistencies in arrival/departure order of packets. In sequential scheduling, the departure order of the packets may be different from their arrival order. In the example in Fig. 2, packet B arrives at the buffer before packet A, but packet A departs first. However, focusing on each input port, the arrival order of the packets is identical to their departure order, except for the discarded packets. For example, packet E will depart the buffer after packet A. Consistency in the arrival/departure order of packets at each port gives better performance for higher layer applications than when there is inconsistency. This is because multiple packets in a flow or a connection (e.g., TCP) usually follow the same route on the Internet. To achieve better performance at the same port speed and to preserve consistency, it is important to determine unit delay D of each FDL and use sequential scheduling. In Refs. [21, 25], the authors showed that the packet loss probability can be minimized by setting D at around $0.3/\mu$ when the offered load is 0.8, where $1/\mu$ is the mean packet length.

3 High-Speed Buffer Management based on a Parallel and Pipeline Processing for Synchronous Fixed-Length Packets

In this section, we propose a high-speed buffer management mechanism based on parallel and pipeline processing for synchronously arriving, fixed-length packets. Although this mechanism requires the use of optical synchronization systems at the input ports of the packet switch [21, 20], the arrival gap and the packet length are not needed to determine packet delays and update buffer occupancy. For example, the pseudo-code for sequential scheduling shown in Fig. 3 can be described as shown in Fig. 4. Here, the packet length and cycle time T are identical to the unit length of FDLs D, and buffer occupancy q is normalized by D. The electronic operation of buffer management becomes simple.

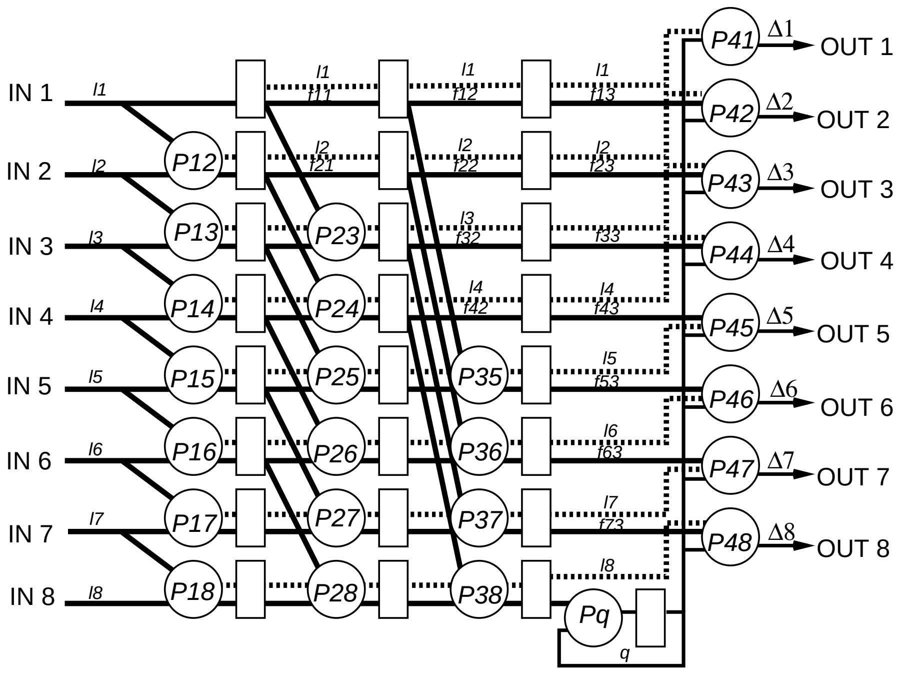

Figure 5: Parallel and pipeline processing architecture ($N = 8$)

3.1 Multi-Processor Architecture and Its Functional Partitioning

The throughput of the sequential scheduling is increased by introducing a parallel and pipeline processing architecture into the buffer manager. We show the architecture for $N = 8$ in Fig. 5. The architecture has $(\log_2 N + 1)$ pipeline stages, and it consists of processors (shown by circles) and registers (shown by rectangles). The front $(log_2 N)$ stages form a prefix operation part. $(N - 2^{k-1})$ processors, $P_{k,n}$ ($n = 2^{k-1} + 1, \ldots, N$), are allocated at the kth stage ($k = 1, 2, \ldots, \log_2 N$). The last stage is a delay determination part. $N + 1$ processors, $P_{\log_2 N+1,n}$ ($n = 1, \ldots, N$) and P_q, are allocated. The solid lines show the bus for transferring the values, $f_{k,n}$ ($1 \leq k \leq \log_2 N$, $1 \leq n \leq N$) and q, which are calculated at each processor. The dashed lines show the bus for the values, l_n ($1 \leq n \leq N$), which arrive at the input ports in the multi-processor architecture. Different from the sequential scheduling, l_n's arrive at the input $(\log_2 N + 1)$-cycle before the packets arrive at the optical switch in the FDL buffer. Arrival information to input port n (labelled "IN n" in the figure) is "1" if the packet arrives at port n, and "0" otherwise. The buffer manager generates signal OUT n, which indicates the delays of packets at port n. The delay signals activate the optical switch in the optical FDL buffer to give delays to the packets.

By using the nature of the sequential scheduling, the delay for the packet arriving at port n is determined as $\Delta_n = q_{start} + s_{n-1}$, where q_{start} is the current status of the buffer occupancy (i.e., the buffer occupancy in the beginning of the cycle) and s_{n-1} is the relative delay of arriving packet at port n, assuming that no packet is currently buffered and that all the arriving packets at ports 1 through $(n - 1)$ will be buffered according to the sequential scheduling. For the synchronous case, the relative delay at port n is determined as the number of packets arriving at ports 1 through $(n - 1)$. For example, the relative delay of an arriving packet at port 1 is always zero. The relative delay of a packet at port 2 is the packet length at port 1 if another packet arrives at port 1 in the same cycle, and zero otherwise. N processors at the $(\log_2 N + 1)$ stage, $P_{\log_2 N+1,n}$ ($n = 1, 2, \ldots, N$),

```
1st stage:                for each processor $P_{1n}$, in parallel ($n := 2$ to $N$)
                              $f_{n,1} := l_n + l_{n-1}$;

2nd stage:                for each processor $P_{2n}$, in parallel ($n := 3$ to $N$)
                              $f_{n,2} := f_{n,1} + f_{n-2,1}$;

kth stage:                for each processor $P_{kn}$, in parallel ($n := 2^{k-1} + 1$ to $N$)
                              $f_{n,k} := f_{n,k-1} + f_{n-2^{k-1},k-1}$;

($\log_2 N + 1$)th stage: for each processor $P_{(\log_2 N+1),n}$, in parallel ($n := 1$ to $N$)
                          begin
                              if ($l_n = 0$) then exit;
                              $\Delta_n := q + f_{n-1,\log_2 N}$;
                              if ($\Delta_n < B$) then Packet $n$ is given delay $\Delta_n D$;
                              else Packet $n$ is discarded;
                          end
```

Figure 6: Pseudo-code for parallel and pipeline mechanism for synchronous case

are used to determine the delay simultaneously. The buffer occupancy is updated at the remaining processor P_q.

The relative delays, s_n's, are calculated by using parallel prefix operation, in which given N elements $< l_1, l_2, \ldots, l_N >$, N prefix sums $< s_1, s_2, \ldots, s_N >$ defined as $(s_n = \sum_{i=1}^{n} l_i)$ are calculated by using N processors [17, 18]. We incorporated pipeline processing into the parallel prefix operation by using front ($\log_2 N$) pipeline stages. The kth-stage processors are devoted to the kth stage of the parallel prefix operation. The value $f_{\log_2 N,n}$ after the $\log_2 N$ stages is the nth prefix sum s_n, which is used for the delay determination, as described earlier.

We describe procedures performed in each time cycle at each stage in the following subsection. Since none of the procedures requires iteration, the time complexity in each processor becomes $O(1)$. The multi-processing architecture increases the throughput of buffer management to N times larger than that of the round-robin scheduling.

3.2 Internal Procedures

We first describe procedures in the prefix operation part, which includes front ($\log_2 N$) pipeline stages. The first-stage processors performs the first operation of the parallel prefix part. For processor P_{1n} $(2 \leq n \leq N)$, values l_n and l_{n-1} indicating whether packets arrive or not at ports n and $(n-1)$, respectively, in $(\log_2 N + 1)$ cycles, are inputted. The value is 1, if there is a packet at the corresponding port, and it is 0 otherwise. Processor P_{1n} performs the procedure in Fig. 6 and generates $f_{n,1}$, which represents the number of packets at the two ports. Value $f_{n,1}$ is stored in the corresponding register.

The second-stage processors are devoted to the second operation of the parallel prefix part. For processor P_{2n} $(3 \leq n \leq N)$, the inputs are connected to the registers of the two first-stage processors, P_{1n} and $P_{1,n-2}$. Processor $P_{2,n}$ thus receives $f_{n,1}$ and $f_{n-2,1}$ and performs the procedure in Fig. 6. It generates $f_{n,2}$, which represents the number of packets at ports $\max(n-3,1)$ through n. Value $f_{n,2}$ is stored in the corresponding register.

The kth-stage processors are devoted to the kth operation of the parallel prefix part. For processor P_{kn} $(2 \leq k \leq \log_2 N$, $2^{k-1} + 1 \leq n \leq N)$, the inputs are connected to the registers of the two $(k-1)$th-stage processors based on the following predetermined rule: the inputs of processor P_{kn} are connected to the registers of processors $P_{k-1,n}$ and $P_{k-1,n-2^{k-1}}$. Processor P_{kn} thus receives $f_{n,k-1}$ and $f_{n-2^{k-1},k-1}$; it then performs the procedure in Fig. 6 and generates $f_{n,k}$, which represents the number of packets at ports $\max(n-2^k+1,1)$ through n. Value $f_{n,k}$ is stored in the corresponding register.

By performing the above procedures at the front $\log_2 N$ stages, each output of the ($\log_2 N$)th stage generates a prefix sum from ports 1 through n ($n = 1, 2, \ldots, N$). The prefix sum is the number of packets arriving at ports 1 through n, which represents the relative delays of packets arriving at port ($n+1$).

We will now describe the delay determination part at the $(\log_2 N+1)$th stage. N processors are devoted to delay determination. For processor $P_{(\log_2 N+1),n}$ ($1 \le n \le N$), the inputs are connected to the register of the $(\log_2 N)$th stage processor, $P_{\log_2 N,n-1}$. Processor $P_{(\log_2 N+1),n}$ thus receives $f_{n-1,\log_2 N}$, which is the relative delay for packet at port n. At the same time, the processor also receives the original information about packet arrival l_n and the buffer occupancy q. The delay to a packet at port n is determined by sum of the buffer occupancy and the relative delay. When $l_n \neq 0$, processor $P_{(\log_2 N+1),n}$ performs the procedure in Fig. 6. The packet on port n is given delay $\Delta_n D$ if $\Delta_n < B$, where $\Delta_n = q + f_{n-1,\log_2 N}$. It is discarded if $\Delta_n \ge B$.

At the last stage, the buffer occupancy is updated at processor P_q. The input of the processor is connected to the register of processor $P_{\log_2 N,N}$. Processor P_q thus receives $f_{N,\log_2 N}$, and updates q as follows.

$$q := \max\left(\min(q + f_{N,\log_2 N}, B) - 1, 0\right). \tag{1}$$

Note that term $q-1$ in the last column in Fig. 4 is replaced with $\min(q + f_{N,\log_2 N}, B) - 1$. We use this term to preserve the consistency of the scheduling mechanism. In sequential scheduling, the buffer occupancy is not updated when a packet is discarded. In contrast, in the proposed mechanism, value $f_{N,\log_2 N}$ indicates the number of arriving packets, and it may also include the number of discarded ones. To avoid overestimation of buffer occupancy, we introduced the above term.

4 Extension in Support of Asynchronous Variable-Length Packets

We extend our parallel and pipeline processing mechanism described in the previous section to support asynchronously arriving variable-length packets. With this extension, optical synchronization and packet fragmentation are not needed.

4.1 Concatenated Packets

The point of allowing the asynchronous variable-length packets is how to determine the relative delays of such packets. If we use optical synchronization, we can directly calculate prefix sums by using the packet lengths as elements. However, because of the discrete-time nature of the FDL buffer and the absence of synchronization, we must take into account the arrival gap (i.e., the time difference between the start of each cycle and the packet arrival time) and the packet length. To set an appropriate interval between two packets to avoid packet contention in the optical buffer, we introduce fictitious concatenated packets that multiple packets arriving at different ports in a cycle are fictitiously connected to. We define length l' of each concatenated packet, which consists of two packets as follows.

$$l' = \left\lceil \frac{t_1 + l_1 - t_2}{D} \right\rceil D + t_2 + l_2 - t_1, \tag{2}$$

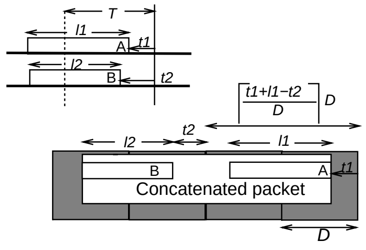

Figure 7: Fictitious concatenation of two packets

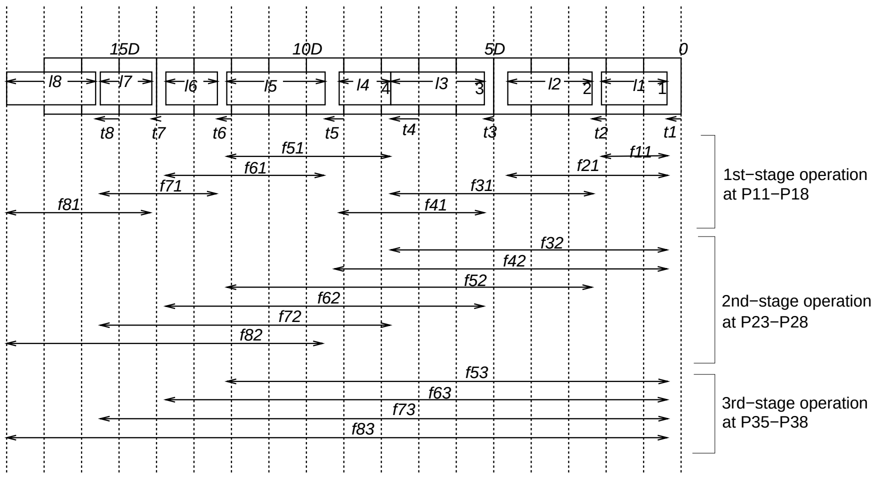

Figure 8: The lengths of concatenated packets generated at the front $\log_2 N$ stages ($N = 8$)

where t_1 and l_1 are, respectively, the arrival gap and the length of the front packet, and t_2 and l_2 are those of the following packet. We also define arrival gap t' of a concatenated packet as $t' = t_2$ if $l_1 = 0$ (i.e., no packet arrives at the front port in the cycle) and $t' = t_1$ otherwise. A concatenated packet is shown in Fig. 7. We use the above definitions in our extension. Figure 8 shows the length of concatenated packets generated at each stage in the prefix operation part. Each notation is defined in Subsection 4.2. By maintaining an appropriate interval between two packets, we avoid packet contention in the optical buffer. We thus calculate relative delays of arriving packets, assuming that no packet is currently buffered and that all the arriving packets will be buffered according to sequential scheduling.

4.2 Multi-Processor Architecture and Internal Procedures

We use a multi-processor architecture similar to that for synchronously arriving packets, except for the internal procedures and interconnected bus widths. For asynchronous case, the bus represented by the solid lines in Fig. 5 are used to transfer the lengths ($f_{n,k}$) and the arrival gaps ($t_{n,k}$) of the concatenated packets, and buffer occupancy q, which are calculated at each processor. The bus represented by the dashed lines are used to transfer the lengths (l_n) and the arrival gaps (t_n) of the packets which arrive at the optical buffer. The architecture consists of $(\log_2 N + 1)$ pipeline stages: the front $\log_2 N$ stages are performed in the prefix operation part, and the last stage is performed in the delay determination part. In this subsection, we describe the procedure at each stage. None of the procedures requires iteration although each procedure is slightly longer than that for synchronously arriving packets. We still achieve $O(1)$ time complexity for each processor even with the extended mechanism.

We first describe procedures in the prefix operation part. The first-stage processors are devoted to the first operation of the parallel prefix part. For processor P_{1n} $(2 \leq n \leq N)$, the following information is inputted about the packets that will arrive at the buffer in $(\log_2 N + 1)$ cycle time.

- l_n and t_n: The length and the arrival gap, respectively, for the packet arriving at port n.
- l_{n-1} and t_{n-1}: The length and the arrival gap, respectively, for the packet arriving at port $n-1$.

If no packet arrives at port n, $l_n = t_n = 0$ is given. Processor P_{1n} performs the procedure in Fig. 9, in which

```
1st stage:  for each processor $P_{1n}$, in parallel ($n := 2$ to $N$)
            begin
               if ($l_{n-1} = 0$) then $t_{n,1} := t_n$
               else $t_{n,1} := t_{n-1}$;
               if ($l_n = 0$ then $f_{n,1} := l_{n-1}$
               else begin
                  $\Delta_n := \left\lceil \frac{t_{n-1} + l_{n-1} - t_n}{D} \right\rceil$;
                  $f_{n,1} := \Delta_n D + t_n + l_n - t_{n-1}$;
               end ;
            end

2nd stage:  for each processor $P_{2n}$, in parallel ($n := 3$ to $N$)
            begin
               if ($f_{n-2,1} = 0$) then $t_{n,2} := t_{n,1}$
               else $t_{n,2} := t_{n-2,1}$;
               if ($f_{n,1} = 0$) then $f_{n,2} := f_{n-2,1}$;
               else begin
                  $\Delta_n := \left\lceil \frac{t_{n-2,1} + f_{n-2,1} - t_{n,1}}{D} \right\rceil$;
                  $f_{n,2} := \Delta_n D + t_{n,1} + f_{n,1} - t_{n-2,1}$;
               end ;
            end

kth stage:  for each processor $P_{kn}$, in parallel ($n := 2^{k-1} + 1$ to $N$)
            begin
               if ($f_{n-2^{k-1},k-1} = 0$) then $t_{n,k} := t_{n,k-1}$
               else $t_{n,k} := t_{n-2^{k-1},k-1}$;
               if ($f_{n,k-1} = 0$) then $f_{n,k} := f_{n-2^{k-1},k-1}$
               else begin
                  $\Delta_n := \left\lceil \frac{t_{n-2^{k-1},k-1} + f_{n-2^{k-1},k-1} - t_{n,k-1}}{D} \right\rceil$;
                  $f_{n,k} := \Delta_n D + t_{n,k-1} + f_{n,k-1} - t_{n-2^{k-1},k-1}$;
               end ;
            end
```

Figure 9: Pseudo-code for the prefix operation part of the parallel and pipeline mechanism for asynchronous case

the length and arrival gap of the concatenated packet is calculated according to the definition in the previous subsection. It generates two values, $f_{n,1}$ and $t_{n,1}$, which represent the length and arrival gap, respectively, for the concatenated packet. The values are stored in the corresponding register.

The second-stage processors are devoted to the second operation of the parallel prefix part. For processor P_{2n} ($3 \leq n \leq N$), the inputs are connected to the registers of the two first-stage processors, P_{1n} and $P_{1,n-2}$. Processor P_{2n} thus receives $f_{n,1}$, $t_{n,1}$, $f_{n-2,1}$, and $t_{n-2,1}$; it performs the procedure in Fig. 9 to generate $f_{n,2}$ and $t_{n,2}$, which represent the length and arrival gap, respectively, for the new concatenated packet.

The kth-stage processors are devoted to the kth operation of the parallel prefix part. For processor P_{kn} ($2^{k-1} + 1 \leq n \leq N$, $2^{k-1} + 1 \leq n \leq N$), the inputs are connected to the registers of the two $(k-1)$th-stage processors based on the following predetermined rule: the inputs of processor P_{kn} are connected to the registers of processors $P_{k-1,n}$ and $P_{k-1,n-2^{k-1}}$. Processor P_{kn} thus receives $f_{n,k-1}$, $t_{n,k-1}$, $f_{n-2^{k-1},k-1}$, and $t_{n-2^{k-1},k-1}$; it performs the procedure in Fig. 9 and generates $f_{n,k}$ and $t_{n,k}$, which represent the length and

```
(log_2 N + 1)th stage:   for each processor $P_{(\log_2 N+1),n}$, in parallel ($n := 1$ to $N$)
                         begin
                             if ($l_n = 0$) then exit;
                             if ($f_{n-1,k} = 0$) then $q' := q$;
                             else $q' := \left\lceil \frac{q - t_{n-1,k}}{D} \right\rceil + t_{n-1,k} + f_{n-1,k}$;
                             $\Delta_n := \left\lceil \frac{q' - t_n}{D} \right\rceil$;
                             if ($\Delta_n < B$) then Packet $n$ is given delay $\Delta_n D$;
                             else Packet $n$ is discarded;
                         end

        queue update:    if ($f_{N,k} = 0$) then $q := \max(q - T, 0)$
                         else begin
                             $\Delta := \left\lceil \frac{q - t_{N,k}}{D} \right\rceil$;
                             if ($\Delta < B$) then
                               $q := \max(t_{N,k} + f_{N,k} + \Delta D - T, 0)$;     /* replaced by Eq.(3) */
                             else $q := q - T$;
                         end
```

Figure 10: Pseudo-code for delay determination part of the parallel and pipeline mechanism for asynchronous case

arrival gap, respectively, for the new concatenated packet. Values $f_{n,k}$ and $t_{n,k}$ are stored in the corresponding register.

By performing the above procedures at the front ($\log_2 N$) stages, each output of the ($\log_2 N$)th stage generates a prefix sum from ports 1 through n ($n = 1, 2, \ldots, N$). The prefix sum is the length and arrival gap of the concatenated packets arriving at ports 1 through n, which represent the relative delays of the packets arriving at port ($n + 1$).

We will now describe the delay determination part at the $(\log_2 N + 1)$th stage. N processors $P_{(\log_2 N+1),n}$ ($1 \leq n \leq N$) are devoted to delay determination. For processor $P_{(\log_2 N+1),n}$, the inputs are connected to the register of the $\log_2 N$th stage processor $P_{\log_2 N,n-1}$. Processor $P_{(\log_2 N+1),n}$ thus receives $f_{n-1,\log_2 N}$ and $t_{n-1,\log_2 N}$, which indicate the relative delay for packet at port n. At the same time, the processor also receives original information (l_n and t_n) and buffer occupancy q. When $l_n \neq 0$, processor $P_{(\log_2 N+1),n}$ first calculates q', which represents the time at which all packets arriving at ports 1 through ($n - 1$) will depart, by using the buffer occupancy and the relative delay. It then calculates $\Delta_n D$, which is the delay for the packet at port n. Finally, the packet is given delay $\Delta_n D$ if $\Delta_n < B$, and it is discarded if $\Delta_n \geq B$. The procedure is described in Fig. 10

At the last stage, the buffer occupancy is updated at processor P_q. The input of the processor is connected to the register of processor $P_{\log_2 N,N}$. Processor P_q thus receives $f_{N,\log_2 N}$ and $t_{N,\log_2 N}$, which represent the length and arrival gap, respectively, for the concatenated packet originally consisting of up to N packets arriving at ports 1 through N. The processor updates q as in Fig. 10.

The parallel and pipeline scheduling mechanism for asynchronously arriving variable-length packets slightly overestimates the buffer occupancy, however, it provides N times faster buffer management than sequential scheduling. Even if processors $\{P_{(\log_2 N+1),n}\}$'s decide that some of the packets cannot be assigned due to buffer overflow, processor P_q simultaneously updates the buffer occupancy by assuming that all the packets can enter

Table 1: Specification

Clock speed	$(1/T)$	78.2 MHz
Port speed	(C)	40.0 Gbps
Number of inputs	(N)	8
Minimum packet length	$(l_{\min})$	64 bytes
Maximum packet length	(MTU)	2,047 bytes
Number of FDLs	(B)	31
Unit length of FDLs	(D)	3.125 m (64 bytes)

the buffer. For instance, when the front two packets in Fig. 8 are assigned to appropriate FDLs and the actual buffer occupancy reaches full, the length of the remaining six packets is also added to the buffer occupancy.

The degradation of the performance of the scheduling resulting from the above overestimation is acceptable, which will be shown in Section 6. We can minimize overestimation by taking into account the fact that the actual buffer occupancy never exceeds the sum of the buffer size ($B \times D$) and the maximum packet length. We achieve this by substituting the fifth line in processor P_q's procedure into the following equation.

$$q \quad := \quad \max\left(\min(t_{N,k} + f_{N,k} + \Delta D, BD + MTU) - T, 0\right), \tag{3}$$

where MTU is the maximum transfer unit (MTU) of packets. We also show the performance of the scheduling Section 6.

5 Hardware Feasibility

We designed hardware for buffer management based on the proposed mechanism and the architecture to verify the feasibility of hardware size and high-speed management. The buffer management scheme is capable of handling asynchronous, variable-length packets, which was described in the previous section. Instead of implementing the scheme, we simulated our buffer management scheme after performing a place-and-route operation on a 0.22μm FPGA device. We modified the mechanism to match it with a hardware description language (HDL). We show the specifications of the buffer manager in Table 1. As shown in the table, we can verify that the buffer manager for 8$\times$8 photonic packet switches works at a frequency of 78.2MHz, which is equivalent to 40Gbps ports. This equivalence is based on the minimum packet length and it is given by the following equation.

$$C = 8l_{\min} \times f_{\max} \times 10^{-3}, \tag{4}$$

where C, $f_{\max}$, and $l_{\min}$ are the port speed (Gbps), the frequency (MHz), and the minimum packet length (byte), respectively.

We next investigate the feasibility of the buffer management hardware depending on the number of ports of the packet switch. We use the latest 0.13μm FPGA devices as target hardware, in which $79,040$ logic cells are highly integrated. We estimate the number of logic cells required for the buffer management hardware by using logic synthesis software. The estimation is shown in Fig. 11. Since the space complexity of our architecture is $O(N \log N)$, we also plot function $64N \log_2 N$ for reference purposes.

We observe that the rate of increase of logic cells for N in our architecture is smaller than that in the reference function, and the number of logic cells in the reference function at $N = 128$ does not reach the number

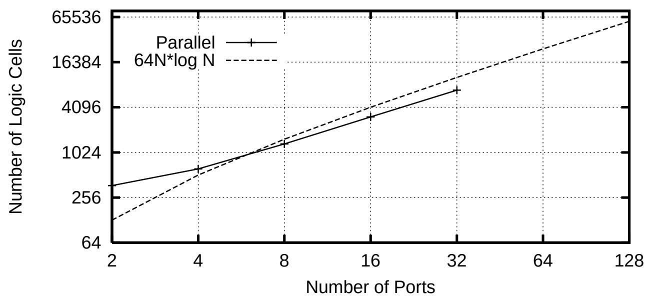

Figure 11: Hardware feasibility depending on the number of ports

of accommodated logic cells on the target FPGA. Thus, our buffer management can support 128-port packet switches. Since 0.13μm FPGA devices provide faster frequency than 0.22μm FPGAs, our buffer management scheme can support 40Gbps ports. Thus we can conclude that a 128×128 packet switch with 40Gbps ports is feasible. By using our buffer management mechanism, a photonic packet switch can easily provide at least ten times as much throughput as the latest 48×48 IP routers with 10Gbps ports. We can expect further speedup of the buffer management by critical path optimization of the hardware or by using ASICs.

6 Scheduling Performance

We show the effect on the parallel and pipelined scheduling scheme through simulation experiments. First, in Subsection 6.1, we compare our scheme with sequential scheduling for optical FDL buffers. In Subsection 6.2, we compare our scheme with a management scheme for RAM buffers. Although the void spaces in the FDL buffer make buffer utilization less efficient compared to the efficiency of RAM buffer utilization, we show that our high-speed buffer management compensates for this inefficiency.

6.1 Comparison of Parallel and Pipeline Scheduling with Sequential Scheduling

We use two sequential scheduling schemes. One is the ideal sequential scheduling, which can handle packets at the same port speed as in the parallel and pipeline scheduling scheme. The other is the $O(N)$ sequential scheduling, in which the port speed becomes slower than in the parallel and pipeline scheduling in proportion to the time complexity. In the latter case, we assume that the port speed is $\frac{1}{N+1}$ times that in the parallel and pipeline scheduling because the sequential scheduling requires $(N+1)$ steps to handle packets in one cycle (N steps for the determination of delays of up to N packets and the remaining step for subtraction). We set $N = 8$ in this evaluation.

In Subsection 6.1.1, we show the delay and packet loss probability for packets of which lengths are distributed quasi-exponentially. In Subsection 6.1.2, we evaluate the packet loss probability by using actual traced data.

6.1.1 Packet Delay and Packet Loss Probability

In this simulation, we focus on an output buffer of a photonic packet switch. We generate 10^6 packets. The packets arrive according to a Poisson process, identically set for all the input ports. The mean packet length

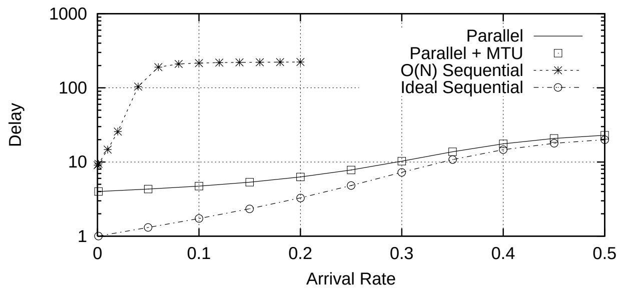

Figure 12: Packet delay ($B = 25, D = 64$ byte)

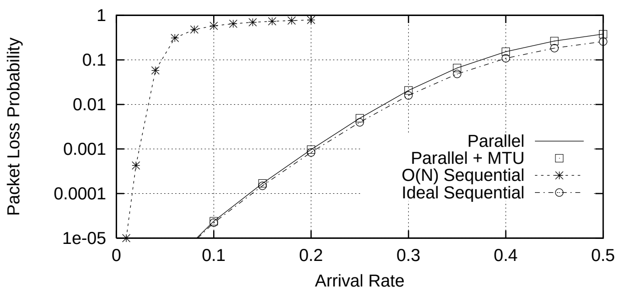

Figure 13: Packet loss probability ($B = 25, D = 64$ bytes)

of each arriving packet is 141.6bytes. The packet length is distributed quasi-exponentially, and the length of each arriving packet is determined as follows. First, we generate a random value as the packet length according to an exponential distribution with a mean of 128bytes. When the length is smaller (larger) than the minimum (maximum) packet length, it is changed into 64bytes (1,500bytes). In the proposed scheduling, the offered load reaches 1.0 at an arrival rate of around 0.45.

We show the delay performance of three scheduling methods (i.e., parallel and pipeline scheduling, $O(N)$ sequential scheduling, and ideal sequential scheduling) for a case of a finite number of FDLs, $B = 25$ in Fig. 12. We also show the performance of the parallel and pipeline scheduling scheme that minimizes the overestimation of buffer occupancy by using Eq. (3), which is labelled "Parallel + MTU" in the figure. The vertical axis is the delay time normalized by the minimum packet length (i.e., 64bytes). The unit time is 12.8nsec when the port speed is 40Gbps. The delay time is defined as the sum of the time needed for scheduling and the time in the FDL buffer. The horizontal axis is the packet arrival rate at the optical buffer, where the rate is normalized by the minimum packet length at the port speed in parallel and pipeline scheduling.

We clearly find that the delay in the parallel and pipeline scheduling is much smaller than that in the $O(N)$ sequential scheduling. This can be accounted for by the following two reasons. First, the processing delay of the parallel and pipeline scheduling scheme is $4/9$ times that of the $O(N)$ sequential scheduling. The parallel and pipeline scheduling scheme requires four processors in a sequence to determine the delay of the packets. On the other hand, it can reduce the per-processor processing time to one-ninth due to the difference in time complexity between the parallel and pipeline scheduling and the $O(N)$ sequential scheduling. Secondly, the difference in the offered load results in delays in the FDL buffer. The port speed in the parallel and pipeline scheduling is nine times faster than that in the $O(N)$ sequential scheduling.

Figure 13 shows performance of scheduling schemes in terms of packet loss probability. The vertical axis shows the packet loss probability. The horizontal axis shows the arrival rate. We find that the parallel and pipeline scheduling scheme clearly outperforms the $O(N)$ sequential scheduling scheme with respect to packet loss probability. The main reason for this result is the difference in the offered load as described above.

From Fig. 12, we can also see that the delay in the parallel and pipeline scheduling is larger than that in the ideal sequential scheduling. The difference is only time equivalent to three minimum-length packets, which is negligible for high-speed ports. The difference arises from the increase in the number of processors required for the pipeline stages. The overestimation of buffer occupancy does not significantly affect the delay performance. In Fig. 13, we can see that the packet loss probability in the parallel and pipeline scheduling is slightly larger than

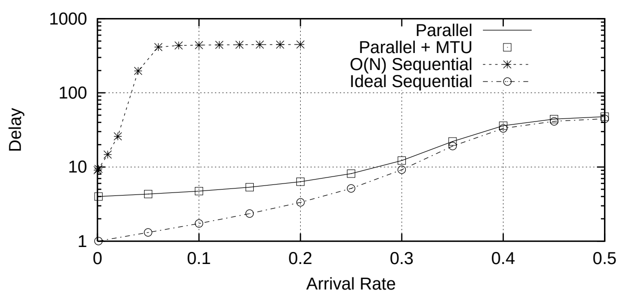

Figure 14: Packet delay ($B = 50, D = 64$ bytes)

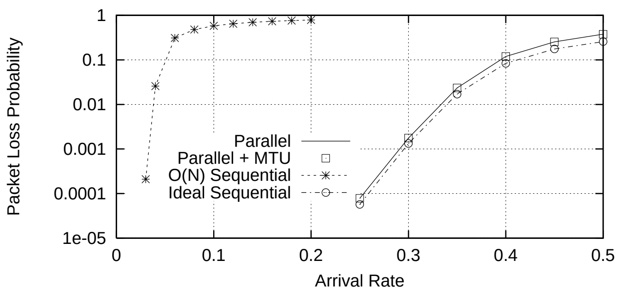

Figure 15: Packet loss probability ($B = 50, D = 64$ bytes)

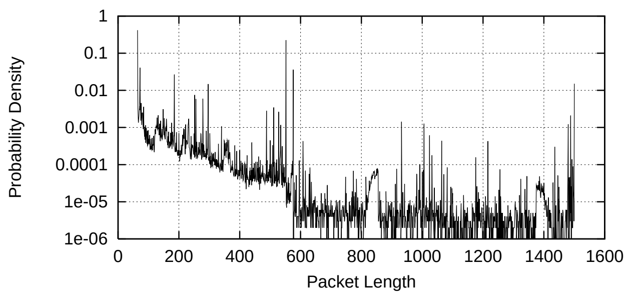

Figure 16: Packet length distribution

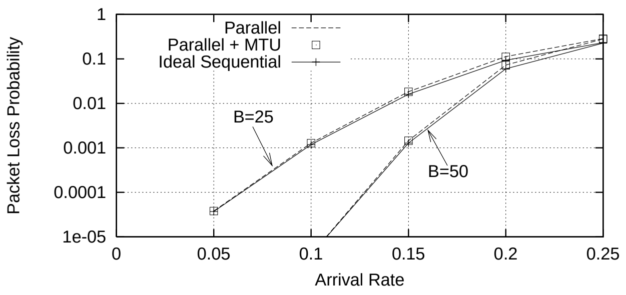

Figure 17: Packet loss probability for the case that packet size distribution follows actual data ($D = 128$ byte)

that in the ideal sequential scheduling due to the overestimation of buffer occupancy. However, the performance degradation is quite acceptable compared to the degradation of the $O(N)$ sequential scheduling.

In order to reduce the packet loss probability in parallel and pipeline scheduling, we can increase the number of FDLs. In Figs. 14 and 15, we show the delay and packet loss performance for $B = 50$. We find that the parallel and pipeline scheduling improves performance in terms of the packet loss probability while all its other advantages are preserved.

6.1.2 Influences of Packet Length Distribution

Since the Internet packet length does not follow an exponential distribution, we then investigate its influence on the scheduling performance. For this purpose, we use actual traced data [19]. We modify the data so that the length of the packets smaller (larger) than 64bytes (1,500bytes) is changed to 64bytes (1,500bytes). The mean packet length is 266.4bytes. Figure 16 show the packet length distribution. The offered load reaches 1.0 at an arrival rate of around 0.24 in the parallel and pipeline scheduling.

Figure 17 shows the packet loss probability depending on the packet arrival rate. From this figure, we find that the difference in packet loss probability between the parallel and pipeline scheduling and ideal sequential scheduling is almost the same as that in quasi-exponentially distributed packet length. We can expect that the influence of packet length distribution is not very significant and that the performance degradation due to the overestimation of buffer occupancy is quite permissible.

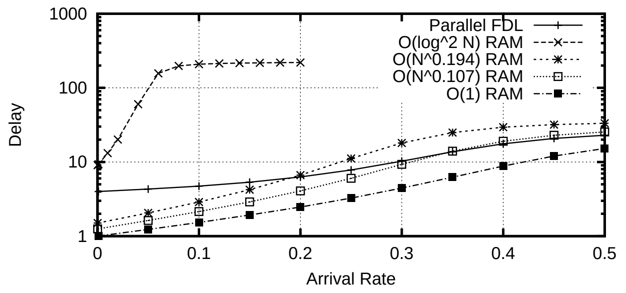

Figure 18: Packet delay in an FDL buffer and in a RAM buffer ($B = 25, D = 64$ bytes)

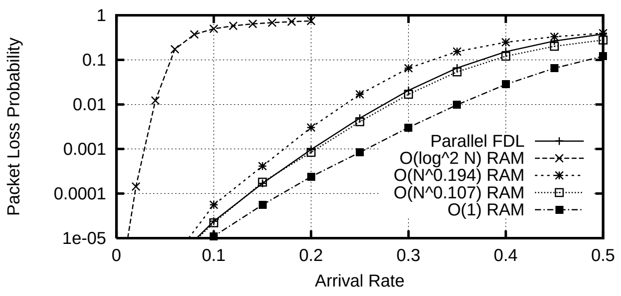

Figure 19: Packet loss probability in an FDL buffer and in a RAM buffer ($B = 25, D = 64$ bytes)

6.2 Comparison to Electronic RAM Buffer Management

Since FDL buffers inherently have void spaces between two optical packets, the utilization of such buffers is less efficient than that of RAM buffers. However, the parallel and pipeline scheduling for FDL buffers compensates for this inefficiency because it enables increasing the port speed more than the other scheduling methods. In this subsection, we compare the performance of our parallel and pipeline scheduling in an FDL buffer with that of several scheduling methods for an electronic RAM buffer. One feasible method for RAM buffers is $O((\log N)^2)$ scheduling [16], but it is much slower than our scheduling. We therefore use fictional $O(N^{0.194})$, $O(N^{0.107})$, and $O(1)$ methods for the RAM buffer. We set $N = 8$. The packet arrival process and the distribution of the packet length are the same as in Subsection 6.1.1.

Figures 18 and 19 show the performance of scheduling schemes in terms of the delay and packet loss probability, respectively. The port speed of the packet switch labelled "$O(x)$" is $1/x$ times that in the parallel and pipelined method for operating all the packets. Since the RAM buffer does not have any void spaces, the performance of the parallel and pipeline scheduling for the FDL buffer is no better than that of the $O(1)$ scheduling for the RAM buffer, and it is almost the same as that of the $O(N^{0.107})$ scheduling for the RAM buffer except for the delay performance at low arrival rates. The $O(N^{0.107})$ scheduling for RAM buffer, which is never proposed, is needed to provide the same performance as that of photonic packet switches. We conclude that photonic packet switches using the parallel and pipeline scheduling scheme compensate for the inefficiency in buffer utilization due to void spaces and provide much better performance than electronic packet switches.

7 Concluding Remarks

We have investigated a high-speed buffer management mechanism for output-buffered photonic packet switches. We have developed an mechanism based on parallel and pipeline processing and its architecture. Since the time complexity of each processor is $O(1)$, the proposed mechanism provides N times faster processing than the existing $O(N)$ mechanism, where N is the number of ports of the packet switch. We have used the mechanism of buffer management for handling asynchronously arriving variable-length packets. We needed to access its feasibility because the space complexity of the architecture is $O(N \log N)$. We have thus conducted hardware simulation experiments after performing a place-and-route operation and have verified feasibility of the FPGA-based buffer manager for 8×8 photonic packet switches with 40Gbps ports, which is capable of variable-length packets of which minimum is 64bytes. We also have found that our buffer manager can support 128×128 photonic packet switches with 40Gbps ports, which can provide ten times as much throughput as the latest electronic IP routers. We expect further speedup of the buffer management by critical path optimization of the hardware or by using ASICs. The proposed mechanism overestimates buffer occupancy to enable parallel processing for delay determination and the update of the buffer occupancy in the delay determination part. The performance degradation resulting from this overestimation is quite acceptable through simulation experiments.

References

[1] E. Mannie *et al.*, "Generalized multi-protocol label switching architecture (draft-ietf-ccamp-gmpls-architecture-05.txt)," *IETF Internet Draft (Work in Progress)*, March 2003.

[2] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Transactions on Communications*, vol. 40, pp. 1171–1182, July 1992.

[3] Procket Networks *available from* `"http://www.procket.com/"`.

[4] N. Wada, H. Harai, and F. Kubota, "40Gbit/s interface, optical code based photonic packet switch prototype," *OFC 2003 Technical Digest*, pp. 801–802, March 2003.

[5] M. Duelk, J. Gripp, J. Simsarian, A. Bhardwaj, P. Bernasconi, M. Zirngibl, and O. Laznicka, "Fast packet routing in a 2.5 Tb/s optical switch fabric with 40 Gb/s duobinary signals at 0.8 b/s/Hz spectral efficiency," *OFC 2003 Post Deadline Paper* (PD8), March 2003.

[6] N. Wada, H. Harai, W. Chujo, and F. Kubota, "Photonic packet routing based on multi-wavelength label switching using fiber Bragg gratings," *ECOC 2000 Technical Digest (26th European Conference on Optical Communication)*, vol. 4 (No. 10.4.6), pp. 71–72, September 2000.

[7] K. Kitayama and N. Wada, "Photonic IP routing," *IEEE Photonic Technology Letters*, vol. 11, pp. 1689–1691, December 1999.

[8] D. K. Hunter and I. Andonovic, "Approaches to optical Internet packet switching," *IEEE Communications Magazine*, vol. 38, pp. 116–122, September 2000.

[9] K. Habara, H. Sanjo, H. Nishizawa, Y. Yamada, S. Hino, I. Ogawa, and Y. Suzaki, "Large-capacity photonic packet switch prototype using wavelength routing techniques," *IEICE Transactions on Communications*, vol. E83-B, pp. 2304–2311, October 2000.

[10] S. Araki, S. Takahashi, Y. Maeno, Y. Suemura, A. Tajima, H. Takahashi, K. Matsuda, T. Tamanuki, S. Dohmae, and N. Henmi, "A 2.56 Tb/s throughput packet/cell-based optical switch-fabric demonstrator," *ECOC '98*, pp. 127–128, 1998.

[11] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, pp. 2081–2094, December 1998.

[12] H. Harai and M. Murata, "Performance analysis of prioritized buffer management in photonic packet switches for DiffServ Assured Forwarding," *Proceedings of SPIE vol. 4874 (OptiComm 2002)*, pp. 298–309, July 2002.

[13] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space division switch," *IEEE Transactions on Communications*, vol. COM-35, pp. 1347–1356, December 1987.

[14] N. McKeown, "The *i*SLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 188–201, April 1999.

[15] R. Sivaram, C. B. Stunkel, and D. K. Panda, "HIPIQS: A high-performance switch architecture using input queueing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 275–289, March 2002.

[16] A. Prakash, S. Sharif, and A. Aziz, "An $O(\log^2 N)$ parallel algorithm for output queueing," *Proceedings of IEEE INFOCOM 2002*, pp. 1623–1629, June 2002.

[17] T. H. Cormen, C. E. Leiserson, and R. H. Rivest, "Algorithms for parallel computers," *Introduction to Algorithms*, ch. 30, MIT Press, 1989.

[18] J. Jájá, *An Introduction to Parallel Algorithms*. Addison Wesley, 1992.

[19] "WAN packet size distribution," *available from "http://www.nlanr.net/NA/Learn/packetsizes.html"*.

[20] T. Sakamoto, A. Okada, M. Hirayama, Y. Sakai, O. Moriwaki, I. Ogawa, R. Sato, K. Noguchi, and M. Matsuoka, "Demonstration of an optical packet synchronizer for an optical packet switch," *OFC 2002 Technical Digest*, pp. 762–763, March 2002.

[21] M. Murata and K. Kitayama, "Ultrafast photonic label switch for asynchronous packets of variable length," *Proceedings of IEEE INFOCOM 2002*, pp. 371–380, June 2002.

[22] L. Tancevski, S. Yegnanarayanan, G. Castanon, L. Tamil, F. Masetti, and T. McDermott, "Optical routing of asynchronous, variable length packets," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2084–2093, October 2000.

[23] A. Ge, L. Tancevski, G. Castanon, and L. S. Tamil, "WDM fiber delay line buffer control for optical packet switching," *Proceedings of SPIE Vol. 4233 (OptiComm 2000)*, pp. 247–256, October 2000.

[24] N. Wada, H. Harai, W. Chujo, and F. Kubota, "Photonic variable length packet routing based on multi-wavelength label switch using multi-section fiber Bragg gratings and supercontinuum light source," *Proceedings of OAA/BGPP 2001 (Optical Amplifiers and Their Applications/ Bragg Graring, Photosensitivity, and Poling in Grass Waveguides 2001)*, vol. JW4, July 2001.

[25] F. Callegati, G. Corazza, and C. Raffaelli, "Exploitation of DWDM for optical packet switching with quality of service guarantees," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 190–201, January 2002.