Optimal Packet Scheduling in Output-Buffered Optical Switches with Limited-Range Wavelength Conversion

Lin Liu and Yuanyuan Yang
Department of Electrical and Computer Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794, USA
{linliu,yang}@ece.sunysb.edu

ABSTRACT

All-optical packet switching is a promising candidate for future high-speed switching. However, due to the absence of optical Random Access Memory, the traditional Virtual Output Queue (VOQ) based input-queued switches are difficult to implement in optical domain. In this paper we consider output-buffered optical packet switches. We focus on packet scheduling in an output-buffered optical packet switch with limited-range wavelength conversion, aiming at maximizing throughput and minimizing average queuing delay simultaneously. We show that it can be converted to a minimum cost maximum network flow problem. To cope with the high complexity of general network flow algorithms, we further present a new algorithm that can determine an optimal scheduling in $O(\min\{W^2, BW\})$ time, where W is the number of wavelength channels in each fiber and B is the length of the output buffer. We also conduct simulations to test the performance of the proposed scheduling algorithm under different traffic models.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet-switching networks

General Terms

Algorithms, Design

Keywords

WDM optical switches, packet scheduling, output-queued (OQ), wavelength conversion, minimum cost maximum flow

1. INTRODUCTION

Advanced optical switching technologies, such as optical packet switching, are required for ultra high speed communications. The recent introduction and rapid growth of the wavelength-division-multiplexing (WDM) technology [1] provides a platform to exploit the huge capacity of optical fiber. Optical switches that combine the advantages of WDM with packet switching capability are strong candidates for future ultra high speed switches with throughput in the range of hundreds of terabits/sec. In a WDM switch, the multiplexing of multiple optical signals on a single fiber is achieved by carrying each signal on a separate wavelength. In an optical packet switched network, data is encapsulated into packets at the source node, and then sent to wavelength channels. At each intermediate node before the destination is reached, packets need to be switched by a crossconnect to their desired output port. Contention of channels arises when more than one packets are destined for the same output wavelength channel. In traditional electronic networks, buffering is a primary method to resolve contention. However, inexpensive, large random access optical buffers are not readily available in current optical technology. The most common implementation of optical buffers is to use fiber delay lines (FDL). The buffering effect is achieved by sending the optical data to be buffered to the FDL. Assume the length of the FDL is L and the optical signal transmits in the FDL at a speed v, then the FDL can provide a fixed delay of L/v to any incoming signal. While fiber delay lines are of relatively low cost and handily available, they are bulky and can only provide discrete storage durations. Recently, slow light technique [10] was proposed, which can generate continuous buffering time by slowing down the optical signal using an external control light source to change the dispersion characteristic of the medium. Buffering architectures based on slow light have been proposed in [6] and [11]. However, slow light may not be as promising as it seems to be. As pointed out in [12], slow-light optical buffers are constrained by some fundamental physical limitations. At least for the near future, efficient implementation of all-optical buffers in optical switches will remain a challenging issue.

Although the design and deployment of optical networks suffer from the lack of a device comparable to random access memory (RAM) in electronic networks, there is yet a unique dimension to resolve contentions in WDM optical networks, which is the conversion in wavelength domain. If wavelength conversion is available, when two packets compete for one output wavelength channel, one of them can be converted to another wavelength on which the output channel is idle. Buffering is not necessary as long as such an alternative wavelength can be found. To maximize throughput while keeping queuing delay under control, a well-designed switch needs to function in both time domain and wavelength domain.

Several optical switch architectures, along with their buffering schemes, have been proposed in recent years [4] [5] [15] [6] [7] [9]. Packet scheduling in bufferless and buffered optical switches has been studied in the literature, see, for example, [13] and [14]. It is usually modeled as a maximum (weighted) matching problem in bipartite graphs. However, directly adopting general maximum matching algorithms for packet scheduling in optical switches may

ANCS'07, December 3–4, 2007, Orlando, Florida, USA.

not be practical due to their relatively high time complexity, as all-optical switches must be operated at a very high speed. A similar problem was encountered in electronic switches and three approaches have been proposed. The first approach is to adopt parallel iterative matching based approximation, of which *i*SLIP [16] is the most well-known algorithm and has been implemented in commercial products. The second approach is to use frame based methods [17] which schedule packets every frame (consisting of a few time slots) instead of every time slot. The third approach is to employ distributed queuing [18] which allocates bandwidth based on the lengths of input and output queues rather than schedules packets one by one. However, *i*SLIP cannot be directly applied to optical switches with wavelength conversion capability, where each packet has multiple possible output wavelength channels to choose. Frame-based scheduling and distributed queuing tend to put additional requirements on buffers, which is currently the weakness of optics. Moreover, all of the three approaches were designed with an objective of optimizing the performance of input-buffered electronic switches with virtual output queuing (VOQ), which may not be suitable to optical switches. Due to the absence of optical RAM, VOQ is difficult to implement in optical domain, yet it is known that input-buffered switches without VOQ suffer from the Head-of-Line blocking and can achieve at most 58.6% throughput. For these reasons, in this paper we consider output-buffered optical switches and design an optimal scheduling algorithm for this type of switch. We formalize packet scheduling in an output-buffered WDM optical switch into a network flow problem and propose a fast algorithm for finding an optimal scheduling in such switches.

The rest of the paper is organized as follows. Section 2 introduces the switch model to be considered. Section 3 describes how to formalize optimal packet scheduling in the switch into a network flow problem. Section 4 presents the new algorithm for finding an optimal scheduling. Section 5 gives the simulation results. Finally, Section 6 concludes the paper.

2. WDM OPTICAL SWITCH MODEL

In this section we introduce the WDM optical packet switch model considered in this paper, including the wavelength conversion model and the switch architecture.

2.1 Wavelength Conversion Model

As mentioned earlier, wavelength conversion is a unique dimension to resolve contentions in WDM switches. Packet scheduling in WDM switches needs to take into consideration the wavelength conversion capability of the switch, since each packet has more than one possible output wavelength channels to choose. In electronic networks, little scheduling needs to be done in an output-queued switch unless extra requirements such as Quality of Service (QoS) must be met, which is also the case in a WDM switch with no wavelength conversion. If full-range wavelength conversion is available in a WDM switch, with which any wavelength can be converted to any other wavelength in the optical system, the scheduling is also trivial since no matter which wavelength a packet comes from, all the output wavelength channels are available for it. Thus, two packets coming from different wavelengths are indistinguishable with regard to scheduling, unless some extra requirements such as QoS are imposed. However, WDM switches without wavelength conversion ability cannot exploit the wavelength domain when there is contention for channels, while the technology to implement full-range wavelength conversion is not yet mature [2]. Therefore, we adopt limited-range wavelength conversion in the switch we consider. With limited-range wavelength conversion, a packet arriving on an input wavelength cannot be converted to any output wavelength, instead, it can only be converted to a fix set of adjacent output wavelengths. Previous studies have shown that with careful designs, limited-range wavelength conversion can achieve performance comparable to that with full-range conversion [2] [3]. In this paper, we assume that the convertible range of a wavelength is symmetric, in other words, a wavelength can be converted to the same number of wavelengths longer and shorter than itself, and this number is defined as the conversion degree. An example of this wavelength conversion model for $W = 6$ and $d = 2$ is shown in Fig. 1, where W represents the total number of wavelengths in the optical system and d is the conversion degree. Note that some wavelengths at the top and the bottom of the figure have a conversion range less than $2d + 1$.

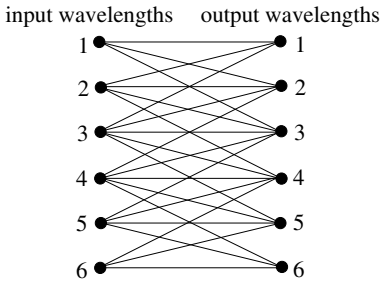


Figure 1: The wavelength conversion model for a switch with $W = 6$ and $d = 2$.

2.2 WDM Switch Architecture

The WDM optical packet switch architecture considered in this paper is shown in Fig. 2. The switch works in a time-slotted manner, and packets of a fixed length arrive at the inputs of the switch at the boundary of a time slot. The length of a time slot is equal to the duration of a packet. The switch has N fibers at the input side and N fibers at the output side, with each fiber containing W wavelength channels. Since the scheduling of packets destined to different output fibers is mutually independent, we only consider the scheduling of packets heading to one of the output fibers.

In this switch architecture, the demultiplexer at each input fiber demultiplexes the incoming optical signal to W signals, one on each wavelength. A limited-range wavelength converter is assigned to each incoming wavelength. An important feature of this switch architecture is that it does not require any speed-up as output-queued switches usually do. This is achieved by implementing the output buffer at each output fiber with $B + 1$ fiber delay lines of lengths from 0 to B. Packets on the same wavelength and destined for the same output fiber can be sent to different delay lines of that fiber in the same time slot without any speed-up requirement, through the internal switching fabric that has more ports than the inputs and outputs of the switch. It can be considered as trading-off the complexity in space domain to release the constraint in time domain. Packets scheduled to the delay line of length 0 will leave the switch immediately. At first glance, the delay lines seem to be able to accept up to $B + 1$ packets on each wavelength in each time slot, and store up to $B(B + 1)/2$ packets on each wavelength. However, as pointed out in [14], this is not true, since on each wavelength there should be no more than one packet coming out of these FDLs at the same time slot. Consider the output buffer on a single wavelength. If we divide an FDL of length l into l units, each of which can hold up to one packet, then at any instant, among all the buffer units of the $B + 1$ FDLs that introduce the same queuing delay, at most one can be occupied. Thus there are a total of $B+1$ *logic buffer cells* on

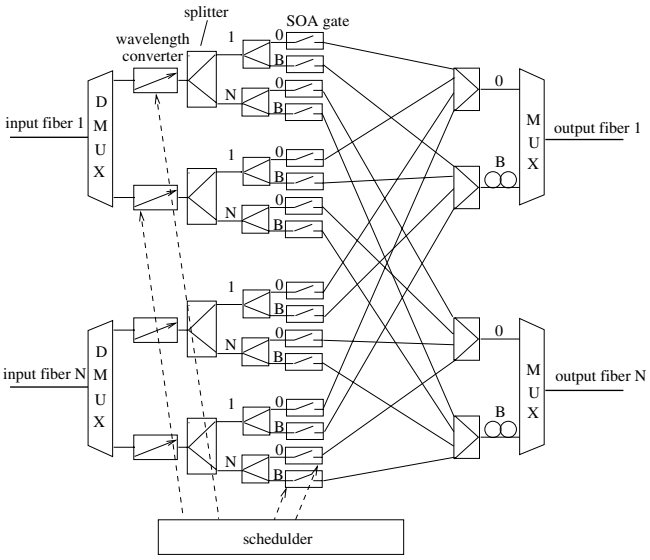


Figure 2: The output-buffered WDM optical switch. The output buffer at each output port is implemented by $B + 1$ fiber delay lines.

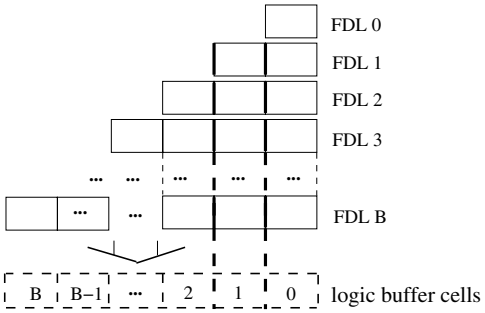


Figure 3: The physical and logic buffer on an output wavelength channel. At any time physical buffer cells in the same column can store at most one packet in total, thus can be viewed as a single logic buffer cell.

each wavelength. Logic buffer cells are labeled by the buffer delay they introduce. That is, cell i of the logic buffer is i time slots away from the output fiber, as shown in Fig. 3. Packets scheduled to logic buffer cell i will be multiplexed and sent to the output fiber in i time slots. Note that Fig. 3 illustrates only one wavelength. Since there are W wavelengths on each fiber, the output buffer at each output fiber can be considered as a $W \times (B+1)$ two-dimensional array of logic buffer cells. All the $B + 1$ logic buffer cells on an output wavelength are accessible to packets at the input side on the same wavelength or wavelengths that can be converted to this wavelength. In the rest of this paper, we will mainly consider logic buffer cells and simply call them buffer cells.

We will see later in the paper that such a buffer scheme can be simply modeled as a first-in-first-out queuing buffer for each output wavelength channel, which greatly simplifies our scheduling algorithm.

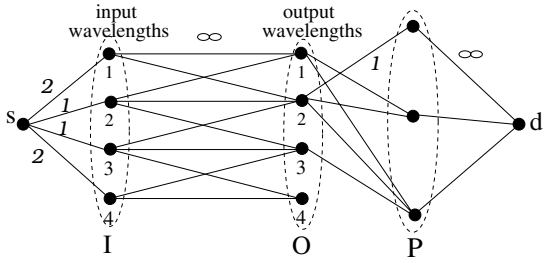


Figure 4: An example of the flow graph for $d = 1, W = 4$ and $B = 2$, where the number of packets arriving in the current time slot on each wavelength is $2, 1, 1$ and 2, respectively. The number above an edge between s and I represents the capacity of the edge. The capacity of edges between I and O and between P and d is infinite, while the capacity of edges between O and P is 1. In the graph, all buffer cells are available on wavelength 2, while no buffer cell is available on wavelength 4.

3. NETWORK FLOW APPROACH FOR FINDING OPTIMAL SCHEDULING

In this section, we consider optimal packet scheduling in the optical switch described in the last section. One observation we can draw for the switch is that packets arriving on the same wavelength and destined for the same output fiber can be treated indistinguishably. Thus instead of matching each of them to the output side, we can schedule as many of them as possible in a bulk at a time, which leads to our idea of adopting the network flow approach. Next we show how to formulate optimal scheduling in the WDM switch into a minimum cost maximum flow problem and solve it by network flow algorithms.

We introduce three sets of nodes, I, O and P. I and O each has W nodes, with each node representing an input or output wavelength, respectively. P has $B + 1$ nodes, with the i_{th} node representing output buffer cell i, $0 \leq i \leq B$. There are also two dummy nodes s and d. There is an edge between s and a node in I, with capacity equal to the number of packets arriving at the input in the current time slot on the wavelength represented by this node. A node in I is connected to a node in O if the wavelengths they represent are mutually convertible. Similarly, there is an edge between a node in O and the i_{th} node in P if buffer cell i is available on the wavelength represented by that node, and the edge has unit capacity and cost i. Finally, all nodes in P are directly connected to d. All other edges without specified capacity and cost are assumed to have an infinity capacity and zero cost. We refer to the subgraph of the flow graph containing nodes in sets I and O, and edges between nodes in these two sets as a *request graph*. A request graph also includes information on the number of packets on each input wavelength channel and the buffer usage status on each output wavelength channel. An example of the flow graph and its corresponding request graph is shown in Fig. 4 and Fig. 5, respectively.

With the flow graph, we can convert the optimal scheduling problem into a network flow problem. Then finding the maximum number of packets that can be scheduled to the output in the current time slot is equivalent to finding a maximum flow from s to d. In the meanwhile, we also need to minimize the total queuing delay, which implies that buffer cells with smaller labels are preferred. Combining these two aspects, it is clear that an optimal scheduling corresponds to a maximum flow with minimum cost in the flow graph. Of course, we can use any existing network flow algorithm to find such a flow. However, although the network flow approach

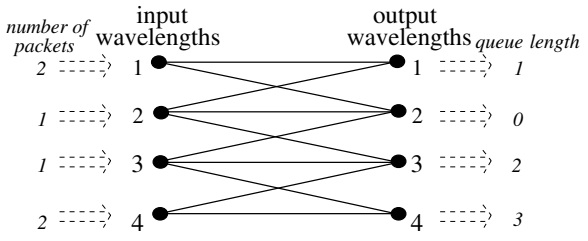


Figure 5: The request graph corresponding to the flow graph in Fig. 4 (we will soon explain why the buffer status on each output channel can be represented by a single number).

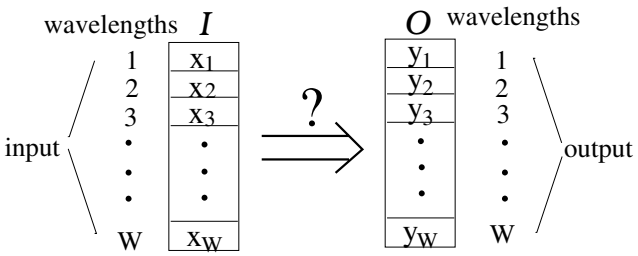


Figure 6: Sets $\mathcal{I}$ and $\mathcal{O}$ of a scheduling. In our case a scheduling can be constructed once its $\mathcal{I}$ and $\mathcal{O}$ are known.

is relatively straightforward, its time complexity may be too high for an optical switch which requires very high-speed switching. In fact, the best known algorithm to find a minimum cost maximum flow in an arbitrary graph requires $O(nC(m + n \lg n))$ time [8], where n is the total number of nodes in the flow graph, m is the number of edges, and C is the largest cost of all edges. In our case, $n = 3 + 2W + B$, $m = O(WB)$ and $C = B$, thus the time complexity of the network flow algorithm is $O(WB^2(W + B))$. Next we will design a new scheduling algorithm with lower time complexity.

4. NEW SCHEDULING ALGORITHM

We first notice that when Quality-of-Service (QoS) is not considered, packets at each input wavelength channel destined for the same output wavelength channel can be treated indistinctively, which is the case in this paper. In such a switch, in each time slot the scheduler must determine how many packets on each wavelength should be transmitted from the input side of the switch, and how these packets should be distributed to the output buffer of the switch. In general, to construct a scheduling, one must know the number of packets that should be scheduled from each input to each output in a switch. However, in our case, as will be proved later in this section, if we know the number of packets on each input wavelength that are to be transmitted in the current time slot, and the number of output buffer cells on each wavelength these packets will use, we can construct an optimal scheduling.

Let x_i denote the number of packets on input wavelength i to be transmitted in the current time slot, and y_i denote the number of buffer cells on output wavelength i to be used by these packets, for $1 \leq i \leq W$. Let $\mathcal{I} = \{x_i | i = 1, 2, \ldots, W\}$ and $\mathcal{O} = \{y_i | i = 1, 2, \ldots, W\}$. Then the main concern of a scheduling algorithm is how to map $\mathcal{I}$ to $\mathcal{O}$ as shown in Fig. 6. Once sets $\mathcal{I}$ and $\mathcal{O}$ are given, a scheduling with minimum queuing delay among all schedulings based on the $\mathcal{I}$ and $\mathcal{O}$ can be constructed as follows. In such scheduling, since $\mathcal{O}$ is known, the buffer cells to be used on each wavelength are known: they must be the ones with labels as small as possible. Starting from wavelength 1, we schedule as many as possible packets on wavelength 1 at the input to wavelength 1 at the output. That is, we schedule $\max\{x_1, y_1\}$ packets from input wavelength 1 to output wavelength 1. This is equivalent to augmenting $\max\{x_1, y_1\}$ along edge (i_1, o_1) in the request graph. If $x_1 \geq y_1$, we schedule as many of the remaining $x_1 - y_1$ packets as possible to output wavelength 2. If $x_1 < y_1$, we schedule $\max\{x_2, y_1 - x_1\}$ packets from input wavelength 2 to the remaining buffer cells on wavelength 1. This process is carried on until all the $\sum_{i=1}^{W} y_i$ cells are occupied. We will give the formal description of this process and the proof in Section 4.4.

Therefore, our main focus next is to find sets $\mathcal{I}$ and $\mathcal{O}$ of an optimal scheduling. Based on the above method and the properties of the output FDL buffer to be discussed next, we will design a new algorithm that is able to efficiently determine $\mathcal{I}$ and $\mathcal{O}$ for an optimal scheduling.

4.1 Properties of Output FDL Buffer

The output FDL buffer in the switch has some nice properties that make it possible to consider the output FDL buffer on each wavelength as a first-in-first-out (FIFO) queuing buffer, which can greatly simplify our optimal scheduling algorithm.

Since an optimal scheduling must schedule the maximum number of packets while keeping the average delay minimized, it will use buffer cells that are as close to the head of the buffer as possible. This leads to the first property of the output buffer that if a buffer cell on a wavelength is not used in an optimal scheduling, then any buffer cells on the same wavelength with a larger label cannot be used by this scheduling. Conversely, if a buffer cell is used in an optimal scheduling, then none of the buffer cells on the same wavelength with a smaller label can be idle after this scheduling. The correctness of this property can be easily seen by contradiction. Suppose that in an optimal scheduling, buffer cell k on wavelength w is not used but there is an $i > k$ such that buffer cell i on this wavelength is used by some packet. By scheduling this packet to buffer cell k and keeping all other packets at the same cells as in the original scheduling, we actually find a new scheduling with the same number of scheduled packets yet shorter total queuing delay, which contradicts the assumption that the original scheduling is optimal. Thus the first half of the property is true. The second half of the property can be justified in a similar manner.

The second property can be derived directly from the first property. Under the assumption that an optimal scheduling was performed in each of the previous time slots, available buffer cells on each output wavelength are consecutive at the beginning of any time slot. In other words, when a time slot starts, if a buffer cell is not available, then any buffer cell on the same wavelength with a smaller label must be unavailable. Conversely, if a buffer cell is available, then all buffer cells on the same wavelength with a larger label must be available.

Combining these two properties we have the following observation that, as long as an optimal scheduling has been performed in each time slot, the output FDL buffer on each wavelength can be considered as a first-in-first-out (FIFO) queuing buffer. As a result, all the available buffer cells on an output wavelength can be recorded by a single variable: the current length of the packet queue on this wavelength. This is also the label at which a buffer cell is first available on this wavelength. It will be seen later that making use of this fact can greatly simplify our optimal scheduling algorithm. It is also worth pointing out that our algorithm works not only with the switch architecture introduced in Section 2, but also with all output-buffered WDM optical switches whose buffer can be modeled as an FIFO queuing buffer.

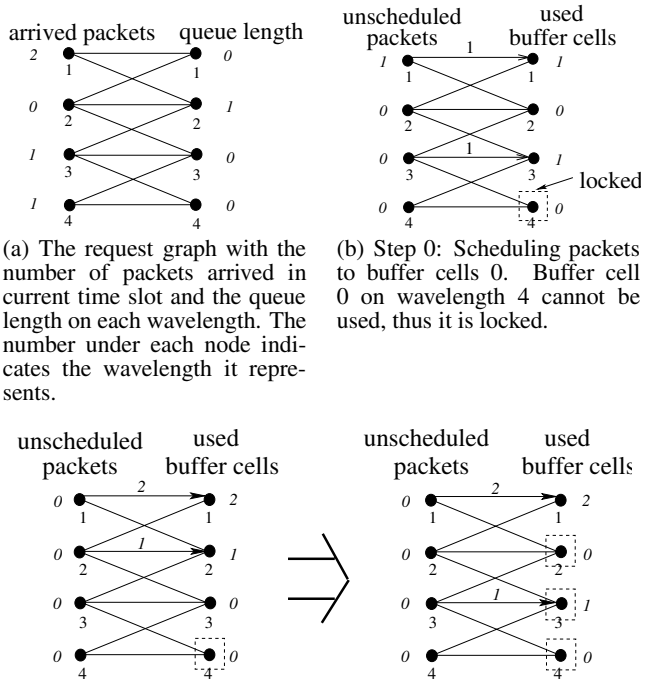
4.2 Augment to Full Algorithm

In this subsection, we give an algorithm called *Augment to Full Algorithm* that can efficiently determine the number of packets on each input wavelength to be transmitted in the current time slot, $\mathcal{I}$, and the number of buffer cells on each output wavelength to be used by these packets, $\mathcal{O}$, of an optimal scheduling from the request graph, based on above properties of the output buffer. The basic idea is similar to that of constructing a scheduling from $\mathcal{I}$ and $\mathcal{O}$ described earlier in this section. Starting from output wavelength 1, we schedule as many as possible packets from input wavelength 1 to the available buffer cells on output wavelength 1. If all packets from input 1 have been scheduled, we say input wavelength 1 is full, or "filled" by output wavelength 1, then we continue to send as many packets as possible from input wavelength 2 to output wavelength 1. If the buffer on wavelength 1 has no more idle cells, we will turn to output wavelengths 2, 3, ..., until input wavelength 2 is finally "filled" by some output wavelength, or the largest wavelength that wavelength 2 can be converted to is reached. Then input wavelength 3 is to be filled. The process continues until there are no more available packets or buffer cells. During this process we fill input wavelengths one by one, thus the process is called *filling process*," which leads to the name of our algorithm "Augment to Full."

To ensure the resulting scheduling is optimal, buffer cells that introduce shorter queuing delay should have a higher priority to be used. The priority is guaranteed in the algorithm by splitting the filling process into $B+1$ steps. In step i, only cells labeled smaller or equal to i will be used to fill the inputs. In this step, if all cells labeled smaller or equal to i are used, no additional work needs to be done. However, it is possible that due to the participation of buffer cell i of each wavelength, some of the buffer cells labeled $i-1$ or smaller that were used in step $i-1$ now cannot be used. In this situation, we need to *lock* certain output wavelengths. Locking an output wavelength in step i means that buffer cells labeled greater or equal to i on this wavelength will not be considered in the following steps, i.e., the "augmenting" of the queue length of this wavelength stops. For instance, suppose in step j when we try to fill the inputs with output wavelength w, k cells on w are available in this step but cannot be used. There are two different cases depending on whether wavelength w is currently locked or not. If w is so far not locked, then $k' = k-1$ of the k cells must have been used in the previous step. The only exception is buffer cell j on wavelength w. If w is already locked in previous steps, this cell is not considered in the current step, thus all the $k' = k$ cells must have been used in step $j-1$. Therefore if $k = 1$ and w is not locked, then simply lock w, since the only cell that cannot be scheduled is buffer cell j on wavelength w; otherwise, we need to ensure the usage of these k' cells by removing k' cells labeled j that are used in this step. This can be done as follows. Starting from w towards w_1, search for an output wavelength whose buffer cell j has been used in this step. Find the k'_{th} such wavelength, remove the k' cells labeled j from the set of used buffer cells, and lock all unlocked output wavelengths between that output wavelength and w.

One property of the Augment to Full Algorithm is that no cross edges are generated in each step of the algorithm. By cross edges we mean two edges (i_1, o_1) and (i_2, o_2) in the request graph, such that $i_1 < i_2$, $o_1 > o_2$, and i_1 has some packets scheduled to o_1, while i_2 also has some packets scheduled to o_2. This property can be seen directly from the way the Augment to Full Algorithm works.

The detailed algorithm is given in Table 1. At the end of the algorithm, the number of packets on each input wavelength to be transmitted and the number of buffer cells on each output wavelength to be used are stored in sets $\mathcal{I}$ and $\mathcal{O}$, respectively.



(a) The request graph with the number of packets arrived in current time slot and the queue length on each wavelength. The number under each node indicates the wavelength it represents.

(b) Step 0: Scheduling packets to buffer cells 0. Buffer cell 0 on wavelength 4 cannot be used, thus it is locked.

(c) Step 1: Due to the use of buffer cell 1 on wavelengths 2 and 3, buffer cell 0 on wavelength 3 that was used in step 0 now cannot be used. As a result, wavelengths 2 and 3 are locked.

Figure 7: Applying the Augment to Full Algorithm to an example request graph.

Fig. 7 gives an example to show how the algorithm works. The request graph of this example was given in Fig. 7(a). In step 0, packets are scheduled to buffer cells with label 0. No packets can be scheduled to buffer cell 0 on wavelength 4. Consequently, no packets will be scheduled to any buffer cell with a larger label than wavelength 4 in the following steps, thus this wavelength is locked. In step 1, due to the use of buffer cell 1 on wavelengths 1 and 2, buffer cell 0 on wavelength 3 that was used in step 0 now cannot be used. As a result, we know that buffer cell 1 on wavelength 2 should not be used, therefore wavelengths 2 and 3 are locked in this step. From the buffer length of an output wavelength at the beginning of the current time slot and the step at which this output wavelength is locked, the number of buffer cells to be used in this time slot can be easily calculated. For instance, the number of buffer cells to be used is $1-1=0$ for wavelength 2 and $1-0=1$ for wavelength 3 in this example.

4.3 Correctness of the Augment to Full Algorithm

To prove the correctness of the Augment to Full Algorithm, we will show that sets $\mathcal{I}$ and $\mathcal{O}$ found by the Augment to Full Algorithm are equal to the $\mathcal{I}$ and $\mathcal{O}$ of an optimal scheduling. As mentioned earlier in this section, from $\mathcal{I}$ and $\mathcal{O}$, we can obtain the number of buffer cells on each wavelength to be used by the scheduling that introduces minimum total queuing delay among all schedulings with the given $\mathcal{I}$ and $\mathcal{O}$, since such scheduling uses buffer cells with labels as small as possible on each wavelength. Consequently, the number of buffer cells on each wavelength to be used by that scheduling can be obtained. Let ${c_i}^S$ denote the number

Table 1: The Augment to Full Algorithm

Input: request graph
Output: sets $\mathcal{I}$ and $\mathcal{O}$ for constructing an optimal scheduling
Notations:
x_j: number of packets on input wavelength j to be transmitted
y_j: number of buffer cells on output wavelength j to be used
q_j: length of packet queue at output wavelength j at the beginning of current time slot
c_j: number of buffer cells that are currently available on wavelength j
p_j: number of packets on input wavelength j that are not scheduled yet
c'_i: number of buffer cells i that are used in step i
in, out: point to input and output wavelength channels currently filled or being filling, respectively.
s_i: the step at which output channel i is locked.
f: number of buffer cells that were used in the previous step but cannot be used in this step due to the use of cells with larger labels.

Algorithm:
Initialize all output wavelength channels to "not locked"
for $i = 0$ to B
  for $j = 1$ to W
    $p_j \leftarrow$ number of packets arrived on input wavelength j in current time slot
    $in \leftarrow 1$; $out \leftarrow$ the first unlocked output wavelength
    $x_j \leftarrow 0$
    if output wavelength j is not locked
      $c_j \leftarrow \max\{0, i + 1 - q_j\}$
    else
      $c_j \leftarrow \max\{0, s_j - q_j\}$
  end for
  while $out \leq W$
    if $c_{out} \leq p_{in}$
      send c_{out} packets from in to out
      $x_{in} \leftarrow x_{in} + c_{out}, p_{in} \leftarrow p_{in} - c_{out}, out \leftarrow out + 1$
      if out is not locked
        $c'_i \leftarrow c'_i + 1$
      end if
    else
      send p_{in} packets from in to out
      $x_{in} \leftarrow x_{in} + p_{in}, c_{out} \leftarrow c_{out} - p_{in}, in \leftarrow in + 1$
    if $out > \min\{W, in + d\}$
      $in \leftarrow in + 1$
    end if
    if $in > \min\{W, out + d\}$
      if out is already locked
        $f = c_{out}$
      else $f = c_{out} - 1$
      if $f = 0$ and out is not locked
        lock out, $s_{out} \leftarrow i$
      else from output wavelength $out - 1$ towards wavelength 1, find the f_{th} output wavelength whose buffer cell j is used, lock each output wavelength w' between the wavelength and wavelength out, $s_{w'} \leftarrow i, y_{w'} \leftarrow \max\{0, i - q_{w'}\}, c'_i \leftarrow c'_i - l$
    end if
  end while
  if $i = B$
end for
$\mathcal{I} = \{x_i | i = 1, 2, \ldots, W\}; \mathcal{O} = \{y_i | i = 1, 2, \ldots, W\}$

of buffer cells with label i to be used in scheduling S, $0 \leq i \leq B$, and S_{af} denote the scheduling with minimum total queuing delay among all schedulings whose $\mathcal{I}$ and $\mathcal{O}$ are equal to the output of the Augment to Full Algorithm in a certain time slot. Then we have the following lemma.

LEMMA 1. *The number of buffer cells with label i to be used by scheduling S_{af}, ${c_i}^{S_{af}}$, satisfies the following recursive property: ${c_i}^{S_{af}}$ is the maximum number of buffer cells with label i that can be used under the precondition that ${c_j}^{S_{af}}$ buffer cells with label j were used for $0 \leq j \leq i - 1$.*

PROOF. Suppose a scheduling S uses buffer cells in such a way that ${c_i}^{S}$ satisfies the above recursive property. We prove by induction that S and S_{af} use the same number of buffer cells for each label, namely, ${c_i}^{S_{af}} = {c_i}^{S}$ for $0 \leq i \leq B$. It is clearly true when $i = 0$, since in this case only cells labeled 0 are considered in the Augment to Full Algorithm, and we use as many as possible of these cells to fill the input channels. Now suppose it is true for $0 \leq i \leq k - 1$. We will prove that it is also true for $i = k$ by contradiction. Assume ${c_k}^{S_{af}} \neq {c_k}^{S}$. The only possibility is ${c_k}^{S_{af}} < {c_k}^{S}$, otherwise, ${c_k}^{S}$ cannot satisfy the recursive property. Then there is a buffer cell labeled k on some wavelength w, denoted as C, and a packet p on input wavelength w', such that C is not in the set of scheduled buffer cells and p is not in the set of scheduled packets after the k_{th} step of the Augment to Full Algorithm, but can be added to the sets without removing any elements already in the sets. This means that there exists an alternating list from p to C among all packets and buffer cells scheduled in S_{af} as shown below:

$$p \longrightarrow C_1 \longrightarrow p_1 \longrightarrow C_2 \longrightarrow p_2 \longrightarrow C_3 \longrightarrow \cdots \longrightarrow p_n \longrightarrow C$$

where C_i denotes a buffer cell and p_i denotes a packet, two adjacent nodes in the list are on wavelengths mutually convertible, and p_i is assigned to C_i by the Augment to Full Algorithm for $1 \leq i \leq n$. By rescheduling packet p to C_1, p_i to C_{i+1} and p_n to C, p and C can be added to S_{af} without removing any packets or buffer cells that are already scheduled in S_{af}. We assume that no more than two consecutive nodes in the list are on the same wavelength. This can be achieved by removing $C_i \rightarrow p_i$ segments in the list as long as p_{i-1} and C_{i+1} are on mutually convertible wavelengths. Since there is no cross edge generated by the Augment to Full Algorithm as discussed in Section 4.2, the wavelengths that the nodes in the list belong to either increase or decrease monotonously. First assume $w' < w$, if such an alternating list could be found, p_n should have been assigned to C instead of C_n based on the Augment to Full Algorithm, which is a contradiction. Then assume $w' > w$. Thus the wavelengths decrease monotonously along the list. As a result, no such C_1 and p_1 can be found. The reason is that, since input wavelength w' is not filled yet, if such C_1 and p_1 existed, p should have been assigned to C_1 instead of p_1 based on the Augment to Full Algorithm. Thus no such alternating list can be found, hence ${c_k}^{S_{af}} = {c_k}^{S}$ must hold. Therefore, ${c_i}^{S_{af}} = {c_i}^{S}$ holds for $0 \leq i \leq B$. □

Now by the recursive property of ${c_i}^{S_{af}}$, we can obtain the following theorem.

THEOREM 1. *Scheduling S_{af} is an optimal scheduling.*

PROOF. We prove it by contradiction. Assume S_{af} is not optimal, which indicates that there exists an optimal scheduling S_{opt} and some l, $0 \leq l \leq B$, such that ${c_j}^{S_{opt}} = {c_j}^{S_{af}}$ for $0 \leq j < l$ but ${c_l}^{S_{opt}} \neq {c_l}^{S_{af}}$. Clearly, ${c_l}^{S_{opt}} < {c_l}^{S_{af}}$, otherwise, we should have a larger ${c_l}^{S_{af}} = {c_l}^{S_{opt}}$. This implies that at least one of the

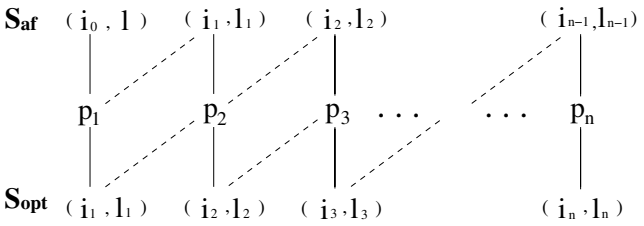


Figure 8: A list of packets that are scheduled in both S_{af} and S_{opt}. (i, l) denotes buffer cell l on wavelength i. p_j is scheduled to cell (i_{j-1}, l_{j-1}) in S_{af} and (i_j, l_j) in S_{opt}. When the list ends, $l_n = l$ must hold. Otherwise, by rescheduling p_j to (i_j, l_j) in S_{af} for $0 \leq j \leq n$, a better scheduling can be obtained, which contradicts the definition of S_{af}.

buffer cells with label l is used in S_{af} but not in S_{opt}. Suppose this cell is on wavelength i_0, and the packets scheduled to this cell in S_{af} is p_1. Then p_1 must have been scheduled in S_{opt} to another buffer cell on some wavelength i_1 with label l_1, where $l_1 \leq l$ must hold, otherwise, we can reschedule p_1 to buffer cell l on wavelength i_0 in S_{opt} and obtain a better scheduling, i.e., a scheduling with the same number of packets to be transmitted but shorter total queuing delay, which contradicts the assumption that S_{opt} is optimal. If in S_{af} some packet p_2 is scheduled to cell l_1 on wavelength i_1, then p_2 must also have been scheduled in S_{opt}, otherwise, in S_{opt} we can reschedule p_2 to buffer cell l_1 on wavelength i_1, and p_1 to buffer cell l on wavelength i_0 and obtain a better scheduling. Suppose the buffer cell occupied by p_2 in S_{opt} is the cell on wavelength i_2 with label l_2, then $l_2 \leq l$ must hold, otherwise, by rescheduling c_2 to cell l_1 on wavelength i_1, and p_1 to cell l on wavelength i_0 in S_{opt} we can obtain a better scheduling. Repeat this process and we will obtain a list of packets as shown in Fig. 8.

Note that for any $k > 0$, $l_k \leq l$ must hold, otherwise, for $1 \leq j \leq k$, rescheduling packet p_j to cell l_{j-1} on wavelength i_{j-1} in S_{opt} will lead to a better scheduling, which contradicts that S_{opt} is optimal. Since the number of packets is finite, the list will end eventually. Suppose the list ends after packet p_n is added to it. That means either (1) p_n is not scheduled in S_{opt}, or (2) the buffer cell used to store p_n in S_{opt}, cell l_n on wavelength i_n, is not used in S_{af}. If it is the first case, a new scheduling better than S_{opt} can be obtained by rescheduling p_j to cell l_{j-1} on wavelength i_{j-1} in S_{opt} for $1 \leq j \leq n$ and keeping the cells for other packets unchanged. Hence when the list ends, it must be the second case. Then $l_n = l$ must hold. Otherwise, since $l_n \leq l$ as proved above, $l_n < l$. If this is true, in S_{af} we can reschedule p_j to buffer cell l_j on wavelength i_j for all $1 \leq j \leq n$. The outcome is that, cell l on wavelength i_0 is replaced by cell l_n on wavelength i_n in S_{af}. Thus the new scheduling must have scheduled more buffer cells than S_{af} for some label smaller than l, which is a contradiction. Thus $l_n = l$.

Then we remove all the packets and buffer cells from S_{af} and S_{opt} that have appeared in the list, and update ${c_j}^{S_{af}}$ and ${c_j}^{S_{opt}}$ accordingly for $0 \leq j \leq l$. All but one of the buffer cells removed in S_{af} and S_{opt} are the same. The only different one is cell l on wavelength w_{i_0} in S_{af} and cell l_n on wavelength i_n in S_{opt}. Since $l_n = l$, the number of removed buffer cells for each label is the same for S_{opt} and S_{af}. Thus after the removal, ${c_j}^{S_{opt}} = {c_j}^{S_{af}}$ still holds for $0 \leq j \leq l-1$, and ${c_l}^{S_{opt}} < {c_l}^{S_{af}}$. Then it means that at least one of the buffer cells with label l is still used in S_{af} but not in S_{opt}. Following all the steps described above, we obtain another list of packets. Again remove all packets and buffer cells in the list from S_{af} and S_{opt} . Repeat this process until no more packets can be removed. At this point ${c_j}^{S_{opt}} = {c_j}^{S_{af}}$ still holds for all $0 \leq j \leq l-1$, and ${c_l}^{S_{opt}} < {c_l}^{S_{af}}$. Then there is a cell with label l used in S_{af} by some packet p, but the cell and the packet are not scheduled in S_{opt}. Then S_{opt} cannot be optimal, which contradicts our initial assumption. Therefore there is no such an optimal scheduling S_{opt}, and S_{af} is optimal. □

From Theorem 1, we can draw the conclusion that the scheduling with $\mathcal{I}$ and $\mathcal{O}$ equal to the $\mathcal{I}$ and $\mathcal{O}$ given by the Augment to Full Algorithm and minimum total queuing delay is an optimal scheduling, that is, a scheduling which can transfer maximum number of packets and achieve minimum total delay. Next we show that such a scheduling can indeed be constructed from sets $\mathcal{I}$ and $\mathcal{O}$.

4.4 Scheduling Construction Algorithm

At the beginning of Section 4, we briefly described how to construct a scheduling from sets $\mathcal{I}$ and $\mathcal{O}$. The formal scheduling construction algorithm is given in Table 2. We now prove the correctness of the scheduling construction algorithm, which consists of two parts: (1) $\mathcal{I}$ and $\mathcal{O}$ of the constructed scheduling are exactly the ones given by the Augment to Full algorithm; (2) the constructed scheduling has minimum total queuing delay among all schedulings that satisfy (1).

The two conditions are unrelated thus can be proved independently. Condition 2 can be satisfied directly by using buffer cells with labels as small as possible on each wavelength, while condition 1 can be proved by contradiction. Suppose we fail to construct a scheduling that has $\mathcal{I}$ and $\mathcal{O}$ as given above, which means that we end up with some unscheduled packets and buffer cells that should have been scheduled, i.e., there exist some wavelengths i and j such that $x'_i < x_i$ and $y'_j < y_j$, where x'_i (y'_j) is the number of packets to be transmitted (buffer cells to be used) on wavelength i (j) in the constructed scheduling. We select the first such unscheduled packet or buffer cell, whichever occurs first, i.e., on a smaller wavelength. If it is an unscheduled buffer cell on wavelength i, then based on the construction algorithm, any packet on a wavelength between wavelengths 1 and $\min\{i+d, W\}$ (which is the largest wavelength i can be converted to), cannot be scheduled to buffer cells on a wavelength larger than i. In other words, all of these packets are scheduled to buffer cells on wavelength i or smaller, but there is still a buffer cell on wavelength i that cannot be used. That is to say, among all the packets and buffer cells that are used in the scheduling with the given $\mathcal{I}$ and $\mathcal{O}$, the maximum number of packets that can be scheduled to output wavelengths in the range of $[1, i]$ is smaller than the number of buffer cells used by this scheduling on these wavelengths, which is impossible. Similarly, if the first unscheduled packet is a packet on wavelength i, then all the buffer cells on wavelengths in the range of $[1, \min\{i+d, W\}]$ are occupied by packets on wavelengths smaller or equal to i. As a result, the maximum number of buffer cells that can be occupied by packets on wavelengths in the range of $[1, i]$ is smaller than the number of packets successfully scheduled to these wavelengths, which is also impossible. Thus, combining all these arguments, we can draw the conclusion that the algorithm in Table 2 can construct a scheduling with the given $\mathcal{I}$ and $\mathcal{O}$, which must be an optimal scheduling by Theorem 1.

4.5 Time Complexity Analysis

We now analyze the time complexity of the new scheduling algorithm. The Augment to Full Algorithm has $B+1$ iterations. In each iteration, two major operations are involved: filling and locking. We first examine the filling part. Define b_w as the largest

Table 2: Scheduling construction from sets $\mathcal{I}$ and $\mathcal{O}$

Input: sets $\mathcal{I}$ and $\mathcal{O}$
Output: a scheduling with the given $\mathcal{I}$ and $\mathcal{O}$ and minimum total queuing delay
Notations:
x_i: number of packets on wavelength i to be transmitted in a scheduling.
y_i: number of buffer cells on wavelength i to be used in a scheduling.
$\mathcal{I} = \{x_i | i = 1, 2, \ldots, W\}$; $\mathcal{O} = \{y_i | i = 1, 2, \ldots, W\}$
in, out: point to input and output wavelength channels currently filled or being filling, respectively.
$p_i(c_i)$: number of remaining packets (buffer cells) that should be scheduled (used) on wavelength i

Algorithm:
$in \leftarrow 1, out \leftarrow 1$
$p_{in} \leftarrow x_{in}, c_{out} \leftarrow y_{out}$
while $in \leq W$ and $out \leq W$
  if $c_{out} \leq p_{in}$
    schedule c_{out} packets from in to buffer cells on out with the smallest labels
    $p_{in} \leftarrow p_{in} - c_{out}, out \leftarrow out + 1$
    if $out \leq W$
      $c_{out} \leftarrow y_{out}$
    end if
  else
    schedule p_{in} packets from in to buffer cells on out with the smallest labels
    $c_{out} \leftarrow c_{out} - p_{in}, in \leftarrow in + 1$
    if $in \leq W$
      $p_{in} \leftarrow x_{in}$
    end if
end while

input wavelength that is "filled" by output wavelength w. Since the number of available buffer cells on each wavelength is nondecreasing as the iteration goes on, b_w is also nondecreasing. Once an input wavelength i is filled by an output wavelength j in some iteration, we only need to consider using wavelength j to fill larger input wavelengths in the following iterations, where input wavelength i will surely be filled by wavelength j or a smaller output wavelength. In other words, during the whole process, the filling of an input wavelength with an output wavelength needs to be done at most once. Since the largest possible number of wavelengths any output wavelength channel can fill is W, the filling operations require at most $O(W^2)$ time during all of the $B + 1$ iterations.

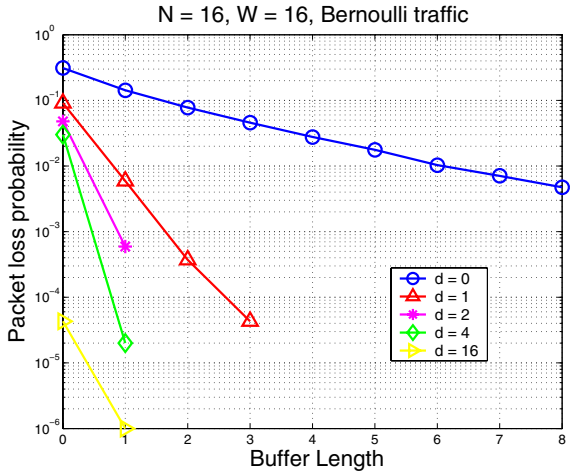
For the locking operations, since the algorithm at most needs to find the wavelength from which locking of output wavelengths starts in $\min\{W, B + 1\}$ of all $B + 1$ iterations, each of which takes at most $O(W)$ time to look up the output wavelength, then all the locking operations in the Augment to Full Algorithm requires $O(W \cdot \min\{W, B + 1\})$ time. Thus the time complexity of the Augment to Full Algorithm is $O(\min\{W^2, BW\})$, where B is the length of the output FDL buffer and W is the number of wavelengths per fiber. We still need to construct the optimal scheduling from the output of the Augment to Full Algorithm. This process has the same time complexity as the filling process in a single iteration, which is $O(W)$. Hence using the new algorithm to find an optimal scheduling has an overall time complexity $O(\min\{W^2, BW\})$, which is lower than directly adopting a general flow algorithm. The low time complexity of our algorithm is mainly due to the two properties discussed in Section 4.1: the output buffer can be treated as an FIFO queue on each channel. As a result, available output buffer cells on a wavelength are consecutive, hence do not need to be considered individually.

5. PERFORMANCE EVALUATIONS

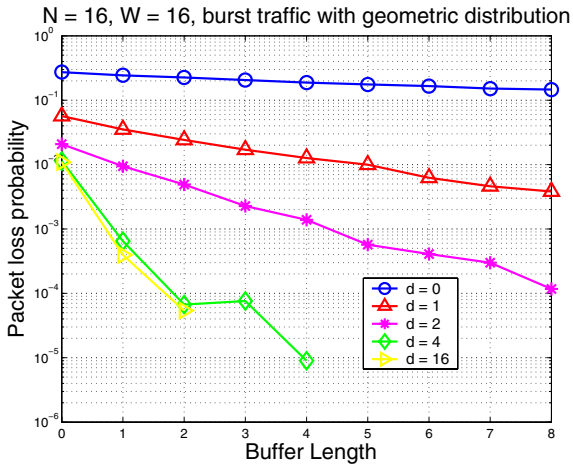
We have conducted simulations to verify the performance of the proposed scheduling algorithm. The main performance criteria considered are packet loss probability and average queuing delay. The simulated switch has 16 input fibers and 16 output fibers, with 16 wavelengths on each fiber. We conducted simulations under three different traffic models, namely, Bernoulli arrival, burst arrival with geometric distribution and burst arrival with Pareto distribution. Bernoulli arrival assumes that at the beginning of each time slot the probability that there is a packet arriving on any input wavelength is solely determined by the traffic intensity ρ. Under the burst arrival model the status of an input wavelength channel alternates between "on" and "off." At the beginning of each time slot there is always a packet arriving on input channels which are "on," and no packet will arrive if during an "off" period. The length of a state may follow different distributions, for example, geometric and Pareto. The former is the ideal on-off traffic model [19], while the latter better simulates the Internet self-similar traffic [20]. The probability density function of a random variable X with Pareto distribution is given by kx_m^k/x^{k+1}, where x_m is the minimum possible value of X, and k is a positive parameter. Its mean is given by $kx_m/(k-1)$ for $k > 1$. In the simulation x_m is set to 1 so that the length of an "on" or "off" state can vary from 1 to infinity time slots. Uniform traffic is assumed for all models in our simulations, i.e., a packet arriving at the input has the same probability to be destined to any of the output fibers. The simulation data were obtained by running the simulation for 10^6 time slots under each traffic model, and the results are plotted in Fig. 9.

It can be seen from Fig. 9 that the packet loss probability is largely affected by the burstness of the traffic. While the packet loss probability is almost the same for all three traffic models when neither buffer nor wavelength conversion are available (i.e., $d = 0$ and $B = 0$), the more bursty the traffic is, the less impact buffering and wavelength conversion can make on packet loss probability. As shown in Fig. 9(a), under Bernoulli traffic, FDLs that can store a few packets will suffice to bring the loss probability to an acceptable level, even when only very limited wavelength conversion is available, for example, $d = 1$ and $B = 3$ produces a loss probability smaller than 10^{-4} in our simulation, while under bursty traffic, the packet loss probability drops rather slowly with the increase of the buffer length. However, under all three traffic models, the improvement in packet loss probability is significant when wavelength conversion becomes available. Especially under burst traffic with Pareto distribution where a small buffer can hardly help reduce the packet loss probability, the increase in wavelength conversion degree from 0 to 1 is crucial as can be seen in Fig. 9(c).

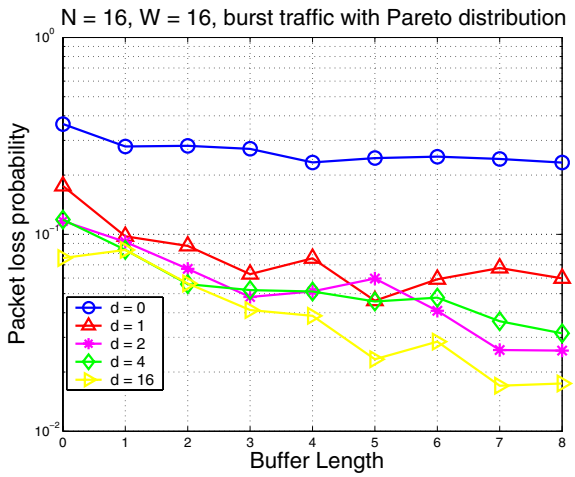
Fig. 10 illustrates the average packet queuing delay under each traffic model. Under Bernoulli traffic and burst traffic with geometric distribution, the average queuing delay converges very fast with the increase of the buffer length as long as $d > 0$. In Fig. 10(a) and Fig. 10(b), when $d > 0$, the average queuing delay becomes stable after B increases from 0 to 1. However, under burst traffic with Pareto distribution (Fig. 10(c)), such nice convergence cannot be observed even at $B = 8$. On the other hand, the average queuing delay with full-range wavelength conversion ($d = 16$) is very close to that with $d = 4$ or even $d = 2$, which implies that for an optical packet switch the ability of wavelength conversion is



(a)



(b)



(c)

Figure 9: Packet loss probability versus buffer length. (a) Bernoulli traffic with $\rho = 0.8$. (b) Burst traffic with geometric distribution and $\rho = 0.8$. (c) Burst traffic with Pareto distribution and $\rho = 0.8$.

critical, while it is not necessarily to be full-range conversion. A similar observation can be drawn for packet loss probability.

In summary, wavelength conversion is a unique method for optical networks to resolve channel contention. A scheduler can efficiently regulate the traffic at the switch input by making use of wavelength conversion. As a result, system performance can greatly benefit from the reduction of traffic burstness.

6. CONCLUSIONS

In this paper we have studied packet scheduling in WDM optical packet switches with output buffer and limited-range wavelength conversion. We have shown that the output buffer can be viewed as a separate FIFO queuing buffer on each output wavelength channel. We have formalized the problem of finding an optimal scheduling in such a switch into a minimum cost maximum flow problem, and presented a new scheduling algorithm, which consists of the Augment to Full Algorithm and the scheduling construction algorithm, to find an optimal scheduling. The new algorithm is able to find an optimal scheduling in $O(\min\{W^2, BW\})$ time, which is faster than directly adopting a general flow algorithm, where W is the number of wavelengths per fiber and B is the length of the longest FDL at the output of the switch. This optimal scheduling algorithm can be applied to any output-buffered WDM optical switches whose output buffer can be modeled as an FIFO queuing buffer on each wavelength. Simulations were conducted to test the performance of the proposed scheduling algorithm under different traffic models.

Acknowledgments

This research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0207999 and CCF-0744234.

7. REFERENCES

[1] L. Xu, H.G. Perros and G. Rouskas, "Techniques for optical packet switching and optical burst switching," *IEEE Communications Magazine*, pp. 136-142, Jan. 2001.

[2] T. Tripathi and K.N. Sivarajan, "Computing approximate blocking probabilities in wavelength routed all-optical networks with limited-range wavelength conversion," *IEEE Journal of Selected Areas in Comm.*, vol. 18, pp. 2123-2129, 2000.

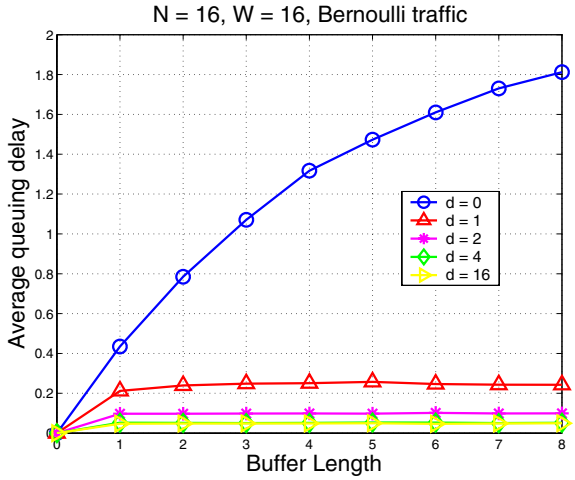
[3] X. Qin and Y. Yang, "Nonblocking WDM switching networks with full and limited wavelength conversion," *IEEE Trans. Communications*, vol. 50, no. 12, pp. 2032-2041, 2002.

[4] Y. Yang and J. Wang, "Designing WDM optical interconnects with full connectivity by using limited wavelength conversion," *IEEE Trans. Computers*, vol. 53, no. 12, pp. 1547-1556, Dec. 2004.

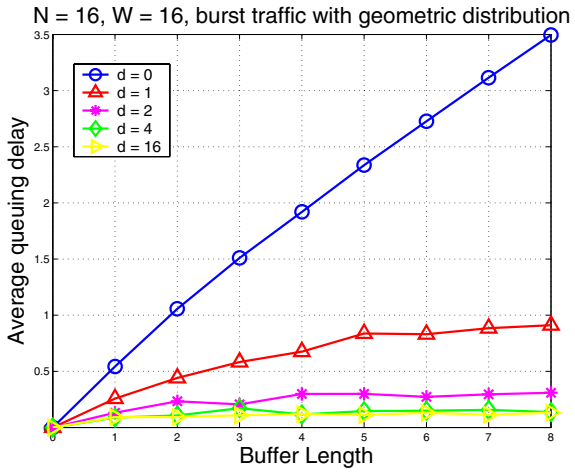
[5] S. Danielsen, C. Joergensen, B. Mikkelsen and K. Stubkjaer, "Analysis of a WDM packet switch with improved performance under bursty traffic conditions due to tunable wavelength converters," *Journal of Lightwave Technology*, vol. 16, pp. 729-735, May 1998.

[6] H. Yang and S.J.B. Yoo,"All-optical variable buffering strategies and switch fabric architectures for future all-optical data routers," *Journal of Lightwave Technology*, vol. 23, pp. 3321-3330, Oct. 2005.

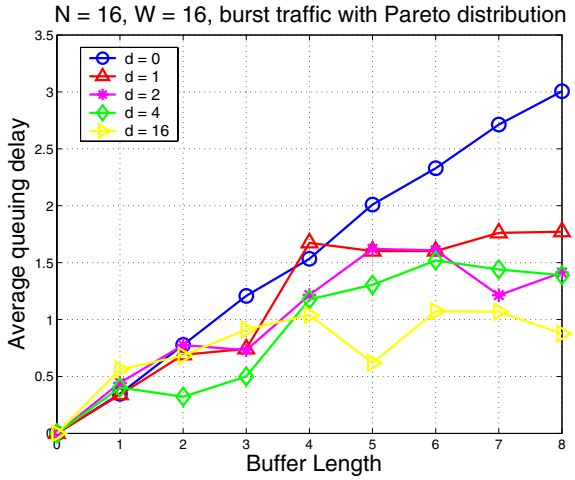
[7] W.D. Zhong and R.S. Tucker, "A new wavelength-routed photonic packet buffer combining traveling delay lines with



(a)



(b)



(c)

Figure 10: Average packet queuing delay versus buffer length. (a) Bernoulli traffic with $\rho = 0.8$. (b) Burst traffic with geometric distribution and $\rho = 0.8$. (c) Burst traffic with Pareto distribution and $\rho = 0.8$.

delay-line loops," *Journal of Lightwave Technology*, vol. 19, No. 8, Aug. 2001.

[8] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall, 1993

[9] L. Li, S.D. Scott and J.S. Deogun, "A novel fiber delay line buffering architecture for optical packet switching," *Proc. GLOBECOM 2003*, vol. 5, pp. 2809-2813, 2003.

[10] P.C. Ku, P. Palinginis, T. Li, F. Sedgwick, S. Chang, H. Wang, C. Chang-Hasnain and S.L. Chuang, "Variable optical buffer using slow light in semiconductor nanostructures," *Proceedings of the IEEE*, vol. 91, pp. 1884-1897, Nov. 2003.

[11] L. Wosinska, J. Haralson, L. Thylen, J. Oberg and B. Hessmo,"Benefit of implementing novel optical buffers in an asynchronous photonic packet switch," *Eur. Conf. Optical Communications (ECOC)*, Stockholm, Sweden, 2004.

[12] R.S. Tucker, P.-C Ku and C.J. Chang-Hasnain, "Fundamental limitations of slow-light optical buffers," *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, vol. 3, Mar. 2005.

[13] Z. Zhang and Y. Yang, "Optimal scheduling algorithms in WDM optical interconnects with limited range wavelength conversion capability," *IEEE Trans. Parallel and Distributed Systems*, vol. 15, pp. 1012-1026, Nov. 2004.

[14] Z. Zhang and Y. Yang,"Optimal scheduling in buffered WDM interconnects with limited range wavelength conversion capability," *IEEE Trans. Computers*, vol. 55, no. 1, pp. 71-82, Jan. 2006.

[15] S. Danielsen, B. Mikkelsen, C. Joergensen, T. Durhuus, and K. Stubkjaer, "WDM packet switch architecture and analysis of the influence of tuneable wavelength converters on the performance," *Journal of Lightwave Technology*, vol. 15, pp. 219-227, Feb. 1997.

[16] N. Mckeown, "The iSlip scheduling algorithm for input-queued switches,"*IEEE/ACM Trans. Networking*, vol. 7, no. 2, Apr. 1999.

[17] A. Bianco, M. Franceschinis, S. Ghisolfi, A. M. Hill, E. Leonardi, F. Neri and R. Webb,"Frame-based matching algorithms for input-queued switches," *High Performance Switching and Routing, 2002. Merging Optical and IP Technologies. Workshop on*, pp. 69-76, 2002.

[18] P. Pappu, J. Parwatikar, J. Turner and K. Wong, "Distributed queueing in scalable high performance routers," *INFOCOM 2003*, vol. 3, pp. 1633-1642, Apr. 2003.

[19] T. Hou and A. Wong, "Queueing analysis for ATM switching of mixed continuous-bit-rate and bursty traffic," *Proc. INFOCOM'90*, pp. 660-667.

[20] M. Garrett and W. Willinger, "Analysis, modeling, and generation of self-similar VBR video traffic," *Proc. SIGCOMM'94*, pp. 269-280.