Segmentation-Based Non-Preemptive Scheduling Algorithms for Optical Burst-Switched Networks

Vinod M. Vokkarane and Jason P. Jue
Department of Computer Science,
The University of Texas at Dallas, Richardson, TX 75083
{vinod, jjue}@utdallas.edu

ABSTRACT

One of the key components in the design of optical burst-switched nodes is the development of channel scheduling algorithms that can efficiently handle data burst contentions. Traditional scheduling techniques use approaches such as wavelength conversion and buffering to resolve burst contention. In this paper, we propose non-preemptive scheduling algorithms that use burst segmentation to resolve contention. We further reduce packet loss by combining burst segmentation and fiber delay lines (FDLs) to resolve contentions during channel scheduling. We propose two types of scheduling algorithms that are classified based on the placement of the FDL buffers in the optical burst-switched node. These algorithms are referred to as *delay-first* or *segment-first* algorithms. The scheduling algorithms with burst segmentation and FDLs are investigated through simulation. The simulation results show that the proposed algorithms can effectively reduce the packet loss probability compared to existing scheduling techniques. The delay-first algorithms are suitable for applications which have higher delay tolerance and strict loss constraints, while the segment-first algorithms are suitable for applications with higher loss tolerance and strict delay constraints.

1. INTRODUCTION

The rapid growth of the Internet will result in an increased demand for higher transmission rates and faster switching technologies. In order to efficiently utilize the amount of raw bandwidth in WDM networks, an all-optical transport method, which avoids electronic buffering while handling bursty traffic, must be developed. Optical burst-switching (OBS) is one such method for transporting traffic directly over a bufferless WDM network.[1]

In OBS networks, bursts of data consisting of multiple packets are switched through the network all-optically. A burst header packet (BHP) is transmitted ahead of the burst in order to configure the switches along the burst's route. The BHP and the data burst are separated at the source, as well as subsequent intermediate nodes, by an offset time, as shown in Fig. 1. The offset time allows for the BHP to be processed at each node before the data burst arrives at the intermediate node; thus, no fiber delay lines are necessary at the intermediate nodes to delay the burst while the BHP is being processed. The BHP may also specify the duration of the burst in order to let a node know when it may reconfigure its switch for the next burst. This signaling technique is known as *just enough time* (JET).[1] In this paper, we will consider an OBS network which uses the JET technique. Each WDM link consists of *control channels* used to transmit BHPs, and *data channels* used to transmit data bursts. In this paper, we assume that every channel consists of a wavelength and that each OBS core router has wavelength conversion capability.

One of the primary OBS core network issues is contention resolution. When two or more bursts are destined for the same output port at the same time, contention occurs. There are many contention resolution schemes[2] which may be used to resolve the contention. The primary contention resolution schemes are optical buffering, wavelength conversion, deflection routing, and burst segmentation. In optical buffering, fiber delay lines (FDLs) are used to delay the burst for a specified amount of time, proportional to the length of the delay line, in order to avoid the contention.[3] In wavelength conversion, if two bursts on the same wavelength are destined to go out of the same port at the same time, then one burst can be shifted to a different wavelength.[4] In deflection routing, one of the two bursts will be routed to the correct output port (primary)

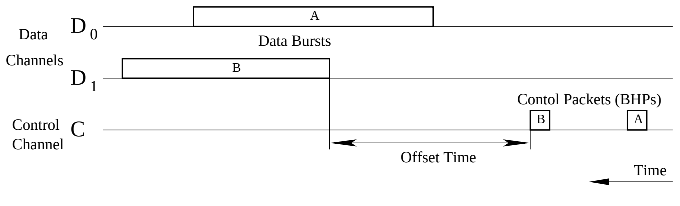


Figure 1. Data and control channels in a WDM Link.

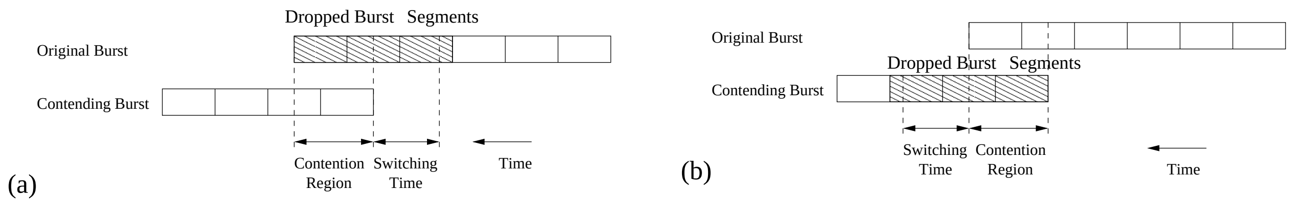


Figure 2. Selective segment dropping for two contending bursts (a) tail dropping policy (b) head dropping policy.

and the other to any available alternate output port (secondary). The deflected packets may end up following a longer path to the destination, leading to higher end-to-end delay, and packets may also arrive at the destination out-of-order.[5–7] A combination of contention resolution techniques may be used to provide high throughput, low delay, and low packet loss probability. In burst segmentation,[8] the burst is divided into basic transport units called *segments*. Each of these segments may consist of a single IP packet or multiple IP packets, with each segment defining the possible partitioning points of a burst when the burst experiences contention in the optical network. All segments in a burst are initially transmitted as a single burst unit. However, when contention occurs, only those segments of a given burst that overlap with segments of another burst will be dropped, as shown in Fig. 2. If switching time is not negligible, then additional segments may be lost when the output port is switched from one burst to another.

There are two approaches for dropping burst segments when contention occurs between bursts. The first approach, tail dropping, is to drop the tail of the original burst (Fig. 2(a)), and the second approach, head dropping, is to drop the head of the contending burst (Fig. 2(b)).[8]

Another important issue at every OBS core router is the scheduling of data bursts onto outgoing data channels. The scheduling algorithm must find an available data channel for each incoming burst in a manner which is quick and efficient, and which minimizes data loss. In order to minimize data loss, the scheduling algorithm may use one or more contention resolution techniques. Data channel scheduling algorithms that use wavelength conversion and FDLs include latest available unscheduled channel (LAUC), and latest available unscheduled channel with void filling (LAUC-VF).[9] However, these techniques drop the burst completely if all of the data channels are occupied at the arrival time of the burst. Instead of dropping the burst in its entirety, it is possible to drop only the overlapping parts of a burst using the burst segmentation technique.

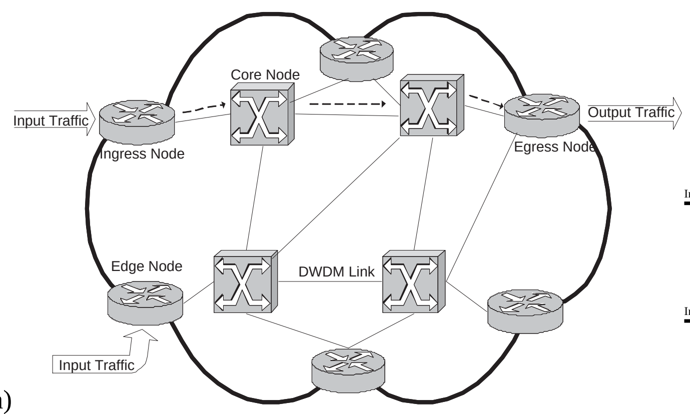
In,[10] burst segmentation and FDLs are combined to resolve contention during channel scheduling. The tail-dropping and head-dropping approaches of segmentation are adopted. Randomly choosing either tail-dropping or head-dropping may result in a change in the burst length of already scheduled bursts. Hence, additional signaling needs to be sent to the downstream nodes in order to notify the nodes of the change in length and to avoid unnecessary contention due to outdated length information. In this paper, we consider segmentation and buffering for resolving contentions without affecting already scheduled bursts.

Due to the inherent property of segmentation, the segmentation-based channel scheduling algorithms can be either *non-preemptive* or *preemptive*. In the non-preemptive approach, existing channel assignments are not altered, while in preemptive scheduling algorithms, an arriving unscheduled burst[*] may preempt existing data channel assignments, and the preempted bursts (or burst segments) may be rescheduled or dropped.

The advantage of a non-preemptive approach is that the BHP of the segmented unscheduled burst can be immediately updated with the corresponding change in the burst length and arrival time (offset time). Also, in non-preemptive channel scheduling algorithms, once a burst is scheduled on the output port, it is guaranteed to be transmitted without being further segmented. The advantage of the preemptive approach can be observed while incorporating QoS into channel scheduling. In this case, a higher priority unscheduled burst can preempt an already scheduled lower priority data burst.[11]

In order to implement non-preemptive schemes, we need to use head dropping on the unscheduled burst for non-void filling based scheduling algorithms. While, we may need to drop both the head and tail of the unscheduled burst for void filling based scheduling algorithms. In order to implement preemptive schemes, we need to use a tail dropping on the scheduled burst for non-void filling based scheduling algorithms. While, we may have to drop both the head and tail of the

[*]Bursts which have been assigned a data channel are referred as the *scheduled bursts*, and the burst which arrives to the node waiting to be scheduled as the *unscheduled burst*.

(a)


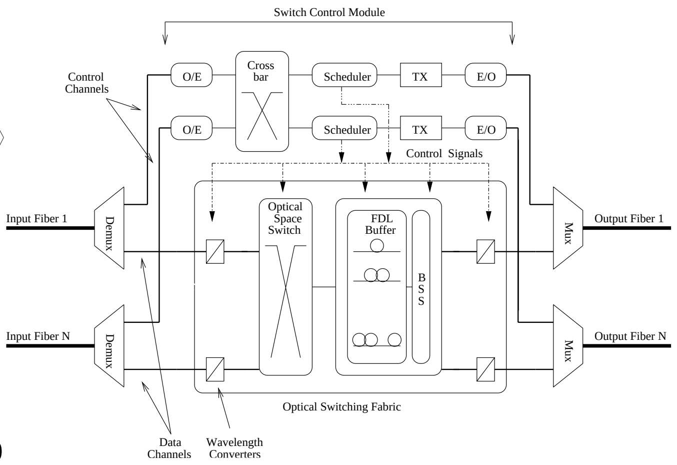
(b)


Figure 3. (a) OBS transport network architecture. (b) Architecture of Core Router.

overlapping scheduled burst for void filling based scheduling algorithms. In the void filling case, if the unscheduled burst overlaps more than two bursts then we have to execute the above procedure on a per burst basis.

In this paper, we propose new segmentation-based non-preemptive scheduling algorithms. These non-preemptive scheduling algorithms perform significantly better in terms of packet loss probability compared to existing scheduling algorithms. The paper is organized as follows. In Section 2, we discuss the OBS network architecture and describe two core node architectures with FDLs. Section 3 describes the existing and proposed data channel scheduling algorithms with and without void filling. Section 4 discusses the new segmentation-based scheduling algorithms with FDLs. Section 5 provides numerical results for the different scheduling algorithms. Section 6 concludes the paper and proposes directions for future research.

2. OBS NETWORK ARCHITECTURE

An OBS network consists of a collection of *edge* and *core* routers (Fig. 3(a)). The edge routers assemble the electronic input packets into an optical burst, which is sent over the OBS core. The ingress edge node assembles incoming packets from the client terminals, into bursts. The bursts are transmitted all-optically over OBS core routers without any storage at intermediate nodes within the core. The egress edge node, upon receiving the burst, disassembles the bursts into packets and provides the packets to the destination client terminals. Basic architectures for core and edge routers in an OBS network have been studied in.[9]

Figure 3(b) shows a typical architecture of an optical burst-switched node, where optical data bursts are received and sent to the neighboring nodes through physical fiber links. The architecture consists primarily of wavelength converters, variable FDLs, an optical space switch, and a switch control module. We assume that all the header packets incur a fixed processing time at every intermediate node. The switch control module processes the BHPs and sends the control information to the switching fabric to configure the wavelength converters, space switch, and broadcast and select switch for the associated data burst. It is important to note that the arrangement of the key components depends on the architecture of OBS node considered. A number of different OBS node architectures are possible using FDLs as optical buffers.

We consider two OBS node architectures with FDLs for realizing the proposed scheduling algorithms. The architecture in Fig. 4(a) shows an input-buffered FDL OBS node with FDLs dedicated to each input port, while Fig. 4(b) shows an output buffered FDL OBS node with FDLs dedicated to each output port.

In the input-buffered OBS node architecture shown in Fig. 4(a), each input port is equipped with an FDL buffer containing N delay lines. The input-buffered architecture supports the delay-first scheduling algorithms. The n data channels are de-multiplexed from each input fiber link and are passed through wavelength converters whose function is to convert the input wavelengths to wavelengths that are used within the FDL buffers. The use of different wavelengths in the FDL buffers and on the output links helps to resolve contentions among multiple incoming data bursts competing for the same FDL and the same output link. In the design of FDL buffers, we can have fixed delay FDL buffers, variable delay FDL buffers, or a mixture of both. In this paper, we follow the architecture of variable delay FDL buffers.

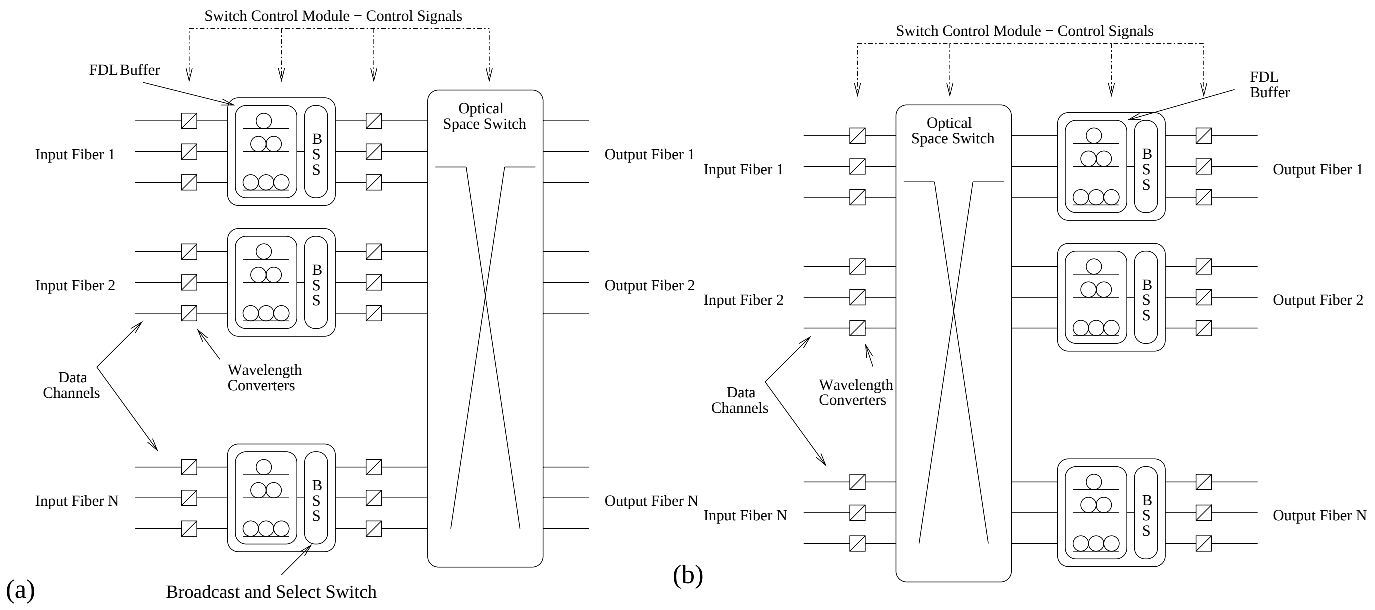


Figure 4. (a) Input-Buffer and (b) Output-Buffer FDL Architecture.

In the output-buffered OBS node architecture, shown in Fig. 4(b), the FDL buffers are placed after the switch fabric. The output-buffered architecture supports the segment-first scheduling algorithms. The input wavelength converters are used to convert the input wavelengths to the wavelengths that are used within the switching fabric. The functions of the output wavelength converters are the same as described in the input-buffer FDL architecture.

In this paper, we only considered the above described *per-port* FDL architectures. In order to minimize switch cost, a *per-node* FDL architecture can be adopted, in which a single set of FDLs can be used for all the ports in a node. This lowering of switch cost results in lower performance with respect to packet loss.

3. NON-PREEMPTIVE CHANNEL SCHEDULING ALGORITHMS

Data channel scheduling in OBS networks is the process of assigning an outgoing data channel for the unscheduled arriving burst. Data channel scheduling in OBS networks is different from traditional IP scheduling. In IP, each core node stores the packets in electronic buffers and schedules them on the desired output port. In OBS, once a burst arrives at a node, it must be sent to the next node without storing the burst in electronic buffers. We assume that each OBS node supports full-optical wavelength conversion.

When a BHP arrives at a node, a channel scheduling algorithm is invoked to assign the unscheduled burst to a data channel on the outgoing link. The channel scheduler obtains the burst arrival time and duration of the unscheduled burst from the BHP. The algorithm may need to maintain the latest available unscheduled time (LAUT), gaps, and voids on every outgoing data channel. Traditionally, the LAUT of a data channel is the earliest time at which the data channel is available for an unscheduled data burst to be scheduled. A gap is the time difference between the arrival of the unscheduled burst and ending time of the previously scheduled burst. A void is the unscheduled duration between two scheduled bursts on a data channel. For void filling algorithms, the starting and the ending time for each burst on every data channel must also be maintained.

The following information is used by the scheduler for all the scheduling algorithms:

- L_b: Unscheduled burst length duration.
- t_{ub}: Unscheduled burst arrival time.
- W: Maximum number of outgoing data channels.
- N_b: Maximum number of data bursts scheduled on a data channel.
- D_i: i^{th} outgoing data channel.

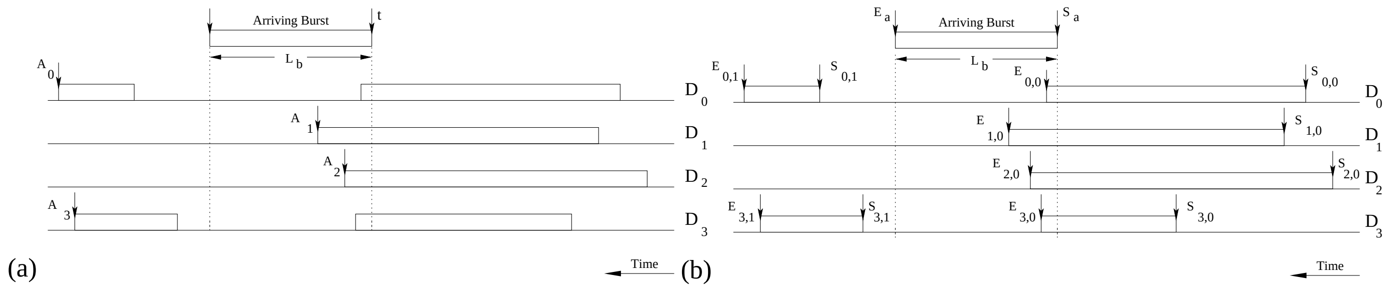


Figure 5. Initial data channel status (a) without void filling (b) with void filling.

- $LAUT_i$: LAUT of the i^{th} data channel, $i = 1, 2, ..., W$, for non-void filling scheduling algorithms.

- $S_{(i,j)}$ and $E_{(i,j)}$: Starting and ending times of each scheduled burst, j, on every data channel, i, for void filling scheduling algorithms.

- $Overlap_i$: Duration of overlap between the unscheduled burst and scheduled burst(s). Overlap is used in non-void filling channel scheduling algorithms. The overlap is zero if the channel is available, otherwise the overlap is the difference between $LAUT_i$ and t_{ub}.

- Gap_i: If the channel is available, gap is the difference between t_{ub} and $LAUT_i$ for scheduling algorithms without void filling, and is the difference between t_{ub} and $E_{(i,j)}$ of previous scheduled burst, j, for scheduling algorithms with void filling. If the channel is busy, Gap_i is set to 0. Gap information is useful to select a channel for the case in which more than one channel is free.

- $Loss_i$: Number of packets dropped due to the assignment of the unscheduled burst on i^{th} data channel. The primary goal of all scheduling algorithms is to minimize loss; hence, loss is the primary factor for choosing a data channel. In case the loss on more than one channel is the same, then other channel parameters are used to reach a decision on the selection of data channel.

- $Void_{(i,k)}$: Duration of k^{th} void on i^{th} data channel. This information is relevant to void filling algorithms. A void is the duration between the $S_{(i,j+1)}$ and $E_{(i,j)}$ on a data channel. Void information is useful in selecting a data channel in case more than one channel is free.

Data channel scheduling algorithms can be broadly classified into two categories: without and with void filling. The algorithms primarily differ based on the type and amount of state information that is maintained at a node about every channel. In data channel scheduling algorithms without void filling, the $LAUT_i$ on every data channel D_i, $i = 0, 1, ..., W$, is maintained by the channel scheduler. In void filling algorithms, the starting time, $S_{(i,j)}$ and ending time, $E_{(i,j)}$ are maintained for each burst on every data channel, where, $i = 0, 1, ..., W$, is the i^{th} data channel and $j = 0, 1, ..., N_b$, is the j^{th} burst on channel i.

We first describe the existing scheduling algorithms without and with void filling. We then describe new segmentation-based scheduling algorithms.

3.1. Existing Channel Scheduling Algorithms

Latest Available Unscheduled Channel (LAUC): The LAUC or Horizon[12] scheduling algorithm keeps track of the LAUT on every data channel and assigns the data burst to the latest available unscheduled data channel. Let the initial data channel assignment for the channel scheduling algorithms without void filling (Fig. 5(a)) and with void filling (Fig. 5(b)) be as shown. In Fig. 5(a), the $LAUT_i$ on every data channel D_i, $i = 0, 1, ..., W$, is maintained by the scheduler. In Fig. 5(b), the starting time, $S_{(i,j)}$ and ending time, $E_{(i,j)}$, where i refer to the i^{th} data channel and j is the j^{th} burst on channel i, are maintained for each burst on every data channel. The LAUC algorithm can be illustrated in Fig. 5(a). No channel is available for the duration of the unscheduled burst. Hence, the entire unscheduled burst is dropped, even though the contention period with bursts on data channel D_2 is minimal. The worst-time complexity of the LAUC algorithm is $O(\log W)$.

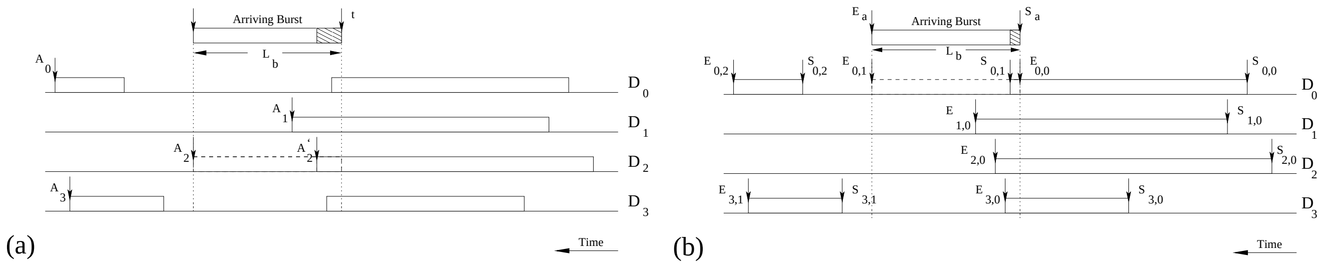


Figure 6. Illustration of non-preemptive (a) NP-MOC scheduling algorithm, and (b) NP-MOC-VF scheduling algorithm.

Latest Available Unscheduled Channel with Void Filling (LAUC-VF): The LAUC-VF,[13] scheduling algorithm maintains the starting and ending times for each scheduled data burst on every data channel. The goal of this algorithm is to utilize voids between two data burst assignments. The channel with a void that minimizes the gap is chosen. The LAUC-VF algorithm is illustrated on Fig. 5(b). No channel is available for the duration of the unscheduled burst; hence, the entire unscheduled burst is dropped, even though the contention period with bursts on data channel D_0 is minimal. If N_b is the number of bursts currently scheduled on every data channel, then a binary search algorithm can be used to check if a data channel is eligible. Thus, the worst-time complexity of the LAUC-VF algorithm is $O(\log(WN_b))$.

3.2. Segmentation-based Non-preemptive Scheduling Algorithms

Non-preemptive Minimum Overlap Channel (NP-MOC): NP-MOC algorithm is an improvement of the existing LAUC scheduling algorithm. The NP-MOC scheduling algorithm keeps track of the LAUT on every data channel. For a given unscheduled burst, the scheduling algorithm considers all outgoing data channels and calculates the overlap on every channel and chooses the data channel with minimum overlap.

NP-MOC ALGORITHM (t_{ub})

```
$tempOverlap \leftarrow INFINITY;$
$tempGap \leftarrow INFINITY;$
$tempChannel \leftarrow -1;$
for each $i \in Data\ Channel$
  { if $(Overlap_i\ is\ ZERO)\ and\ (Gap_i < tempGap)$
      { $tempGap \leftarrow Gap_i;$
        $tempChannel \leftarrow i;$
if $(tempChannel <> -1)$
  { $Schedule\ the\ Unscheduled\ Burst\ on\ D_i;$
    $Stop;$
else
  { for each $i \in Data\ Channel$
      { if $(Overlap_i < tempOverlap)$
          { $tempOverlap \leftarrow Overlap_i;$
            $tempChannel \leftarrow i;$
if $(tempChannel <> -1)$
  { $Resolve\ Contention\ using\ Non-preemptive\ Segmentation$
    $Schedule\ the\ Unscheduled\ Burst\ on\ D_i;$
    $Stop;$
else
  { $Drop\ Unscheduled\ Burst;$
    $Stop;$
```

The details of NP-MOC are given above. For example, applying the NP-MOC algorithm to the example in Fig. 5(a), we see that data channel D_2 has the minimum overlap, and the unscheduled burst is scheduled on D_2 (Fig. 6(a)). Here, only the overlapping segments of the unscheduled burst are dropped instead of the entire unscheduled burst as in the case of LAUC . The worst-time complexity of the NP-MOC algorithm is $O(\log W)$.

Non-preemptive Minimum Overlap Channel with Void Filling (NP-MOC-VF): The NP-MOC-VF scheduling algorithm maintains the starting and ending times of each data burst on every data channel. The goal is to utilize the voids between data burst assignments on every data channel. The data channel with a void that minimizes the Gap_i is chosen in case of more than one available channel. If no channel is free, the channel with minimum loss is assigned to the unscheduled

Table 1. Comparison of Segmentation-based Non-preemptive Scheduling Algorithms

Algorithm	Time Complexity	State Information
LAUC	O(log W)	$LAUT_i$, Gap_i
LAUC-VF	O(log(WN_b))	$S_{(i,j)}$, $E_{(i,j)}$, Gap_i
NP-MOC	O(log W)	$LAUT_i$, Gap_i
NP-MOC-VF	O(log(WN_b))	$S_{(i,j)}$, $E_{(i,j)}$, Gap_i

burst. The details of NP-MOC-VF are given below. For example, applying the NP-MOC-VF algorithm to the example in Fig. 5(a), we see that data channel D_0 has the minimum overlap, and the unscheduled burst is scheduled on D_0 (Fig. 6(b)). Here, only the overlapping segments of the unscheduled burst are dropped instead of the entire unscheduled burst as in the case of LAUC-VF. The worst-time complexity of the NP-MOC-VF algorithm is $O(\log(WN_b))$.

NP-MOC-VF ALGORITHM (t_{ub})

```
$tempLoss \leftarrow INFINITY;$
$tempGap \leftarrow INFINITY;$
$tempChannel \leftarrow -1;$
for each $i \in Data\ Channel$
{
    if $(Loss_i\ is\ ZERO)\ and\ (Gap_i < tempGap)$
    {
        $tempGap \leftarrow Gap_i;$
        $tempChannel \leftarrow i;$
    }
}
if $(tempChannel <> -1)$
{
    $Schedule\ the\ Unscheduled\ Burst\ on\ D_i;$
    $Stop;$
}
else
{
    for each $i \in Data\ Channel$
    {
        if $(Loss_i < tempLoss)$
        {
            $tempLoss \leftarrow Loss_i;$
            $tempChannel \leftarrow i;$
        }
    }
}
if $(tempChannel <> -1)$
{
    $Resolve\ Contention\ using\ Non-preemptive\ Segmentation$
    $Schedule\ the\ Unscheduled\ Burst\ on\ D_i;$
    $Stop;$
}
else
{
    $Drop\ Unscheduled\ Burst;$
    $Stop;$
}
```

Table I compares all the above channel scheduling algorithms in terms of time complexity and the amount of state information stored. We observe that the time complexity of the non-void filling algorithms is less than that of the void filling algorithms. Also, void filling algorithms, such as, LAUC-VF and NP-MOC-VF, store more state information as compared to non-void filling algorithms, such as, LAUC and NP-MOC.

4. SEGMENTATION-BASED NON-PREEMPTIVE SCHEDULING ALGORITHMS WITH FDLS

There has been substantial work on scheduling using FDLs in OBS.[9, 12, 14] In this section, we propose a number of segmentation-based non-preemptive scheduling algorithms incorporating FDLs. Based on the two FDL architectures presented in Section 2, we have two families of scheduling algorithms. Scheduling algorithms based on the input-buffer FDL node architecture are called *delay-first* scheduling algorithms, while scheduling algorithms based on the output-buffer FDL node architecture are called *segment-first* scheduling algorithms. In both schemes, we assume that full wavelength conversion, FDLs, and segmentation techniques are used to resolve burst contention for an output data channel. However, the order of applying the above techniques depends on the FDL architecture. In delay-first schemes, we resolve contention by wavelength conversion, FDLs, and segmentation, in that order, while in segment-first schemes, we resolve contention by wavelength conversion, segmentation, and FDLs, in that order. Before going on to the detailed description of the schemes, it is necessary to discuss the motivation for developing two different schemes. In delay-first schemes, FDLs are primarily used to delay the entire burst, while in segment-first schemes, FDLs are primarily used to delay the segmented bursts. Delaying the entire burst and then segmenting the burst keeps the packets in order; however, when delaying segmented

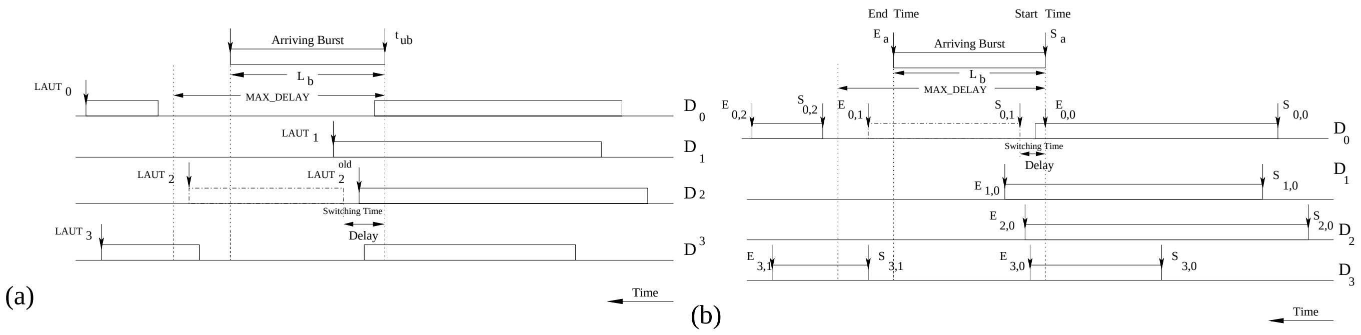


Figure 7. Illustration of (a) NP-DFMOC algorithm, and (b) NP-DFMOC-VF algorithm.

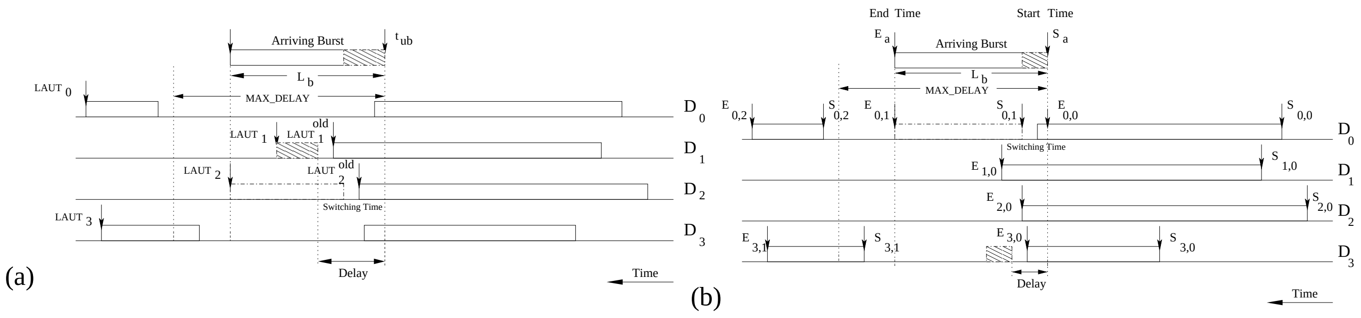


Figure 8. Illustration of (a) NP-SFMOC algorithm, and (b) NP-SFMOC-VF algorithm.

bursts, packet order is not always maintained. In general, segment-first schemes will incur lower delays than delay-first schemes. In both the schemes, the scheduler has to additionally know MAX_DELAY, i.e., the maximum delay provided by the FDLs.

We will now describe the segmentation-based non-preemptive scheduling algorithms which use segmentation, wavelength conversion, and FDLs.

4.1. Delay-First Scheduling Algorithms

Non-preemptive Delay-First Minimum Overlap Channel (NP-DFMOC): The NP-DFMOC algorithm calculates the overlap on every channel and then selects the channel with minimum overlap. If a channel is available, then the unscheduled burst is scheduled on the free channel with the minimum gap. If all channels are busy and the minimum overlap is greater than or equal to the sum of the unscheduled burst length and MAX_DELAY, then the entire unscheduled burst is dropped. Otherwise, the unscheduled burst is delayed for the duration of the minimum overlap and scheduled on the selected channel. In case the minimum overlap is greater than MAX_DELAY, the unscheduled burst is delayed for MAX_DELAY and the non-overlapping burst segments of the unscheduled burst is scheduled, while the overlapping burst segments are dropped. For example, in Fig. 7(a), the data channel D_2 has the minimum overlap, thus the unscheduled burst is scheduled on D_2 after providing a delay using FDLs.

Non-preemptive Delay-First Minimum Overlap Channel with Void Filling (NP-DFMOC-VF): The NP-DFMOC-VF algorithm calculates the delay until the first void on every channel and then selects the channel with minimum delay. If a channel is available, the unscheduled burst is scheduled on the free channel with minimum gap. If all channels are busy and the starting time of the first void is greater than or equal to the sum of the end time, E_a, of the unscheduled burst and MAX_DELAY, then the entire unscheduled burst is dropped. Otherwise, the unscheduled burst is delayed until the start of the first void on the selected channel, where the non-overlapping burst segments of the unscheduled burst are scheduled, while the overlapping burst segments are dropped. In case the start of the first void is greater than the sum of the start time, S_a, of the unscheduled burst and MAX_DELAY, then the unscheduled burst is delayed for MAX_DELAY and the non-overlapping burst segments of the unscheduled burst are scheduled, while the overlapping burst segments are dropped. For example, consider Fig. 7(b). By applying the NP-DFMOC-VF algorithm, the data channel D_0 has the minimum delay, thus the unscheduled burst is scheduled on D_0 after delaying the burst using FDLs. In this case, only the overlapping segments of the burst are dropped instead of the entire burst as in the case of LAUC-VF.

Table 2. Comparison of Segmentation-based Non-preemptive Scheduling Algorithms with FDLs

Algorithm	Time Complexity	State Information
LAUC	O(log W)	$LAUT_i$, Gap_i
LAUC-VF	O(log(WN_b))	$S_{(i,j)}$, $E_{(i,j)}$, Gap_i
NP-DFMOC	O(log W)	$LAUT_i$, Gap_i
NP-DFMOC-VF	O(log(WN_b))	$S_{(i,j)}$, $E_{(i,j)}$, Gap_i
NP-SFMOC	O(log W)	$LAUT_i$, Gap_i
NP-SFMOC-VF	O(log(WN_b))	$S_{(i,j)}$, $E_{(i,j)}$, Gap_i

4.2. Segment-First Scheduling Algorithms

Non-preemptive Segment-First Minimum Overlap Channel (NP-SFMOC): The NP-SFMOC algorithm calculates the overlap on every channel and then selects the data channel with minimum overlap. If a channel is available, the unscheduled burst is scheduled on the free channel with the minimum Gap_i. If all channels are busy and the minimum overlap is greater than or equal to the sum of the unscheduled burst length and MAX_DELAY, then the entire unscheduled burst is dropped. Otherwise, the unscheduled burst is segmented (if necessary) and the non-overlapping burst segments are scheduled on the selected channel, while the overlapping burst segments are re-scheduled. Next, the algorithm calculates the overlap on all the channels for the re-scheduled burst segments. The re-scheduled burst segments are delayed for the duration of the minimum overlap and scheduled on the selected channel. In case the minimum overlap is greater than MAX_DELAY, then the re-scheduled burst segments are delayed for MAX_DELAY and the non-overlapping burst segments of the re-scheduled burst segments are scheduled, while the overlapping burst segments are dropped. For example, in Fig. 8(a), we observe that the data channel D_2 has the minimum overlap for the unscheduled burst, thus the unscheduled burst is scheduled on D_2, and the re-scheduled burst segments are scheduled on D_1.

Non-preemptive Segment-First Minimum Overlap Channel with Void Filling (NP-SFMOC-VF): The NP-SFMOC-VF algorithm calculates the loss on every channel and then selects the channel with minimum loss. If a channel is available, the unscheduled burst is scheduled on the free channel with minimum gap. If all channels are busy and the starting time of the first void is greater than or equal to the sum of the end time, E_a, of the unscheduled burst and MAX_DELAY, then the entire unscheduled burst is dropped. If the starting time of the first void is greater than or equal to the end time, E_a, of the unscheduled burst, the NP-DFMOC-VF algorithm is employed. Otherwise, the unscheduled burst is segmented (if necessary) and the non-overlapping burst segments are scheduled on the selected channel, while the overlapping burst segments are re-scheduled. For the re-scheduled burst segments, the algorithm calculates the delay required until the start of the next void on every channel and selects the channel with minimum delay. The re-scheduled burst segments are delayed until the start of the first void on the selected channel. The non-overlapping burst segments of the re-scheduled burst are scheduled, while the overlapping burst segments are dropped. In case the start of the next void is greater than the sum of the start time, S_a, of the unscheduled burst and MAX_DELAY, the re-scheduled burst segments are delayed for MAX_DELAY and the non-overlapping burst segments of the re-scheduled burst are scheduled, while the overlapping burst segments are dropped. For example, in Fig. 8(b), we observe that the data channel D_0 has the minimum loss, thus the unscheduled burst is scheduled on D_0, and the unscheduled burst segments are scheduled on D_3 (as it incurs the minimum delay) after providing a delay using FDLs.

Table II compares all of the discussed segmentation-based non-preemptive channel scheduling algorithms with FDLs in terms of time complexity and the amount of state information stored. We can observe that the time complexity of the non-void filling algorithms is less than the void filling algorithms. Also, void filling algorithms, such as, LAUC-VF, NP-DFMOC-VF, and NP-SFMOC-VF, store more state information as compared to non-void filling algorithms, such as, LAUC, NP-DFMOC, and NP-SFMOC.

5. NUMERICAL RESULTS

In order to evaluate the performance of the proposed channel scheduling algorithms, a simulation model is developed. Burst arrivals to the network are Poisson, and each burst length is an exponentially generated random number rounded to the nearest integer multiple of the fixed-sized packet length of 1250 bytes. The average burst length is 100 μs. The link transmission rate is 10 Gb/s. Current switching technologies provide us with a range of switching times from a few ms (MEMs)[15] to a few ps (SOA-based).[16] We assume a conservative switch reconfiguration time of 10 μs. The burst header processing time at each node depends on the architecture of the scheduler and the complexity of the scheduling algorithm.

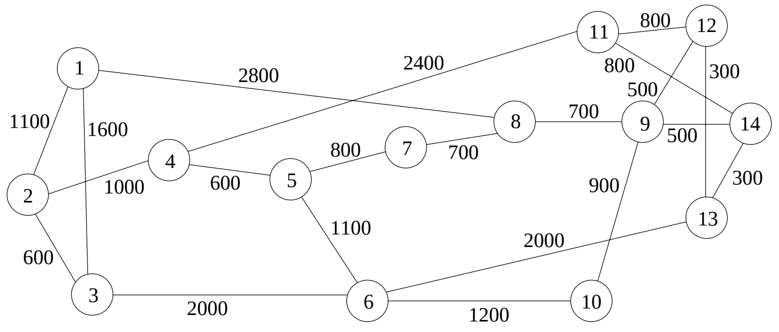


Figure 9. 14-Node NSF Network.

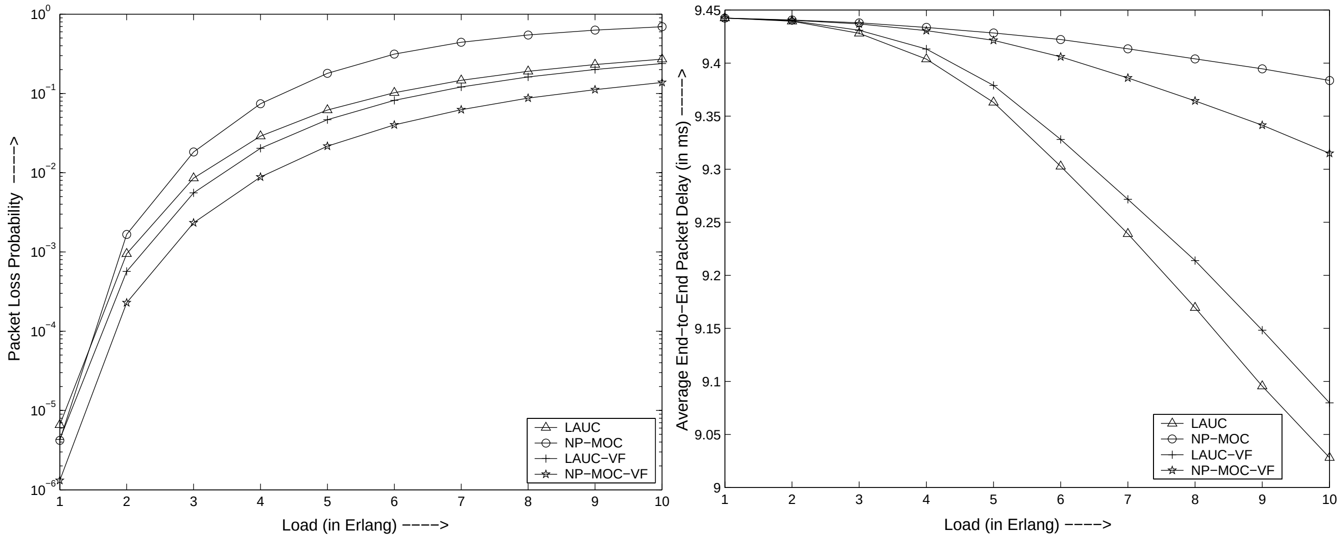


Figure 10. (a) Packet loss probability versus load, and (b) average end-to-end delay versus load for different scheduling algorithms with 8 data channels on each link, for the NSF network.

Based on current CPU clock speeds and a conservative estimate of the number of instructions required, we assume burst header processing time to be 2.5 μs. We know that in any optical buffer architecture, the size of the buffers is severely limited, not only by signal quality concerns, but also by physical space limitations. To delay a single burst for 5 μs requires over a kilometer of fiber. Due to this size limitation of optical buffers, we consider a maximum FDL delay of 0.01 ms. Traffic is uniformly distributed over all sender-receiver pairs. Fixed minimum-hop routing is used to find the path between all node pairs. All the simulation are implemented on the standard 14-node NSF network shown in Fig. 9, where link distances are in km.

Figure 10(a) plots the total packet loss probability versus load for different channel scheduling algorithms, with 8 data channels on each link. We observe that the segmentation-based channel scheduling algorithms perform significantly better than algorithms without segmentation. The proposed segmentation-based scheduling algorithms perform better than the algorithms without segmentation because, when contention occurs, only the overlapping packets from one of the bursts are lost instead of the entire burst. We see that NP-MOC suffers lower loss as compared to LAUC. Also, NP-MOC-VF performs better than LAUC-VF. We can also observe that NP-MOC and NP-MOC-VF are the best algorithms without and with void filling respectively. Also, the algorithms with void filling perform better than algorithms without void filling as expected.

Figure 10(b) plots the average end-to-end delay versus load for different channel scheduling algorithms, with 8 data channels on each link. We observe that the segmentation-based channel scheduling algorithms have higher average end-to-end packet delay than existing channel scheduling algorithms without segmentation. The higher delay for scheduling algorithms with segmentation is due to the higher probability of a successful transmission between source-destination pairs which are farther apart, while in traditional scheduling algorithms the entire burst is dropped in case of a contention; hence,

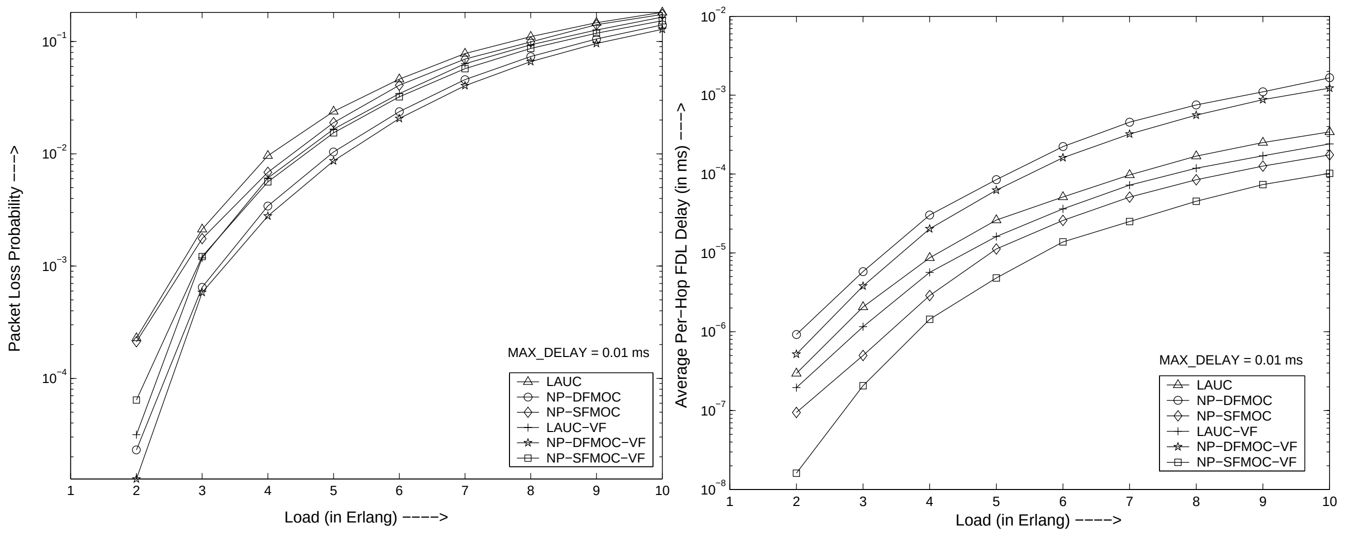


Figure 11. (a) Packet loss probability versus load, and (b) average per-hop FDL delay versus load for different scheduling algorithms with 8 data channels on each links, for the NSF network.

source-destination pairs close to each other have a higher probability of making a successful transmission, which results in lower average end-to-end packet delay. We see that the NP-MOC algorithm has higher delay than the LAUC algorithm. Also, the NP-MOC-VF algorithm has higher delay than the LAUC-VF algorithm. We can also observe that LAUC has the least average end-to-end packet delay among all the algorithms.

Figure 11(a) plots the total packet loss probability versus load for different channel scheduling algorithms. We observe that the channel scheduling algorithms with burst segmentation perform better than algorithms without burst segmentation at most loads. Also, the delay-first algorithms have lower loss as compared to the segment-first algorithms. This is due to the possible blocking of the re-scheduled burst segment by the recently scheduled non-overlapping burst segment in the segment-first algorithms. The loss obtained by delay-first algorithms is the lower bound on delay for the segment-first algorithms. We observe that at any given load, the NP-DFMOC and NP-DFMOC-VF algorithms perform the best, since the unscheduled burst is delayed; and in case there is still a contention, the burst is segmented and only the overlapping burst segment is dropped. The segment-first algorithms lose packets proportional to the switching time every time there is a contention, while the LAUC and LAUC-VF algorithms delay the burst in case of a contention and schedule the burst if the channel is free after the provided delay. Hence, at low loads, LAUC-VF performs better than NP-SFMOC-VF, and, as the load increases, NP-SFMOC-VF performs better. Therefore a substantial gain is achieved by using segmentation and FDLs.

Figure 11(b) plots the average per-hop FDL delay versus load for different channel scheduling algorithms. We observe that the delay-first algorithms have higher per-hop FDL delay as compared to the segment-first algorithms, since FDL is the primary contention resolution technique in the former and segmentation is the primary contention resolution technique in the latter. We also observe that the per-hop FDL delay of void filling algorithms is lower than non-void filling, since the scheduler can of assign the arriving bursts to closer voids that incur lower FDL delay as compared to scheduling the bursts at the end of the horizon (LAUT) in the case of non-void filling algorithms. Hence, we can carefully choose either delay-first or segment-first schemes based on loss and delay tolerances of input IP packets.

When a high MAX_DELAY value is used, algorithms which use FDLs as the primary contention resolution technique, such as, LAUC, LAUC-VF, NP-DFMOC, NP-DFMOC-VF outperform the algorithms which use segmentation as the primary contention resolution technique, such as, NP-SFMOC, NP-SFMOC-VF.[10]

6. CONCLUSION

In this paper, we considered burst segmentation and FDLs for burst scheduling in optical burst-switched networks, and we proposed a number of channel scheduling algorithms for OBS networks. The segmentation-based scheduling algorithms perform better than the existing scheduling algorithms with and without void filling in terms of packet loss. We also introduced two categories of scheduling algorithms based on the FDL architecture. The delay-first algorithms are suitable for transmitting packets which have higher delay tolerance and strict loss constraints, while the segment-first algorithms are suitable for transmitting packets which have higher loss tolerance and strict delay constraints.

Areas of future works include extending the proposed algorithms to support QoS. In the case of providing QoS support, the priority of the burst can be stored in the BHP,[11] and the scheduling algorithm can dynamically decide which of contending bursts or burst segments to drop using burst segmentation or to delay using FDLs. Hence a combination of preemptive and non-preemptive segmentation-base techniques can be used to provide service differentiation. It would also be useful to develop an analytical model for the proposed scheduling algorithms.

REFERENCES

1. C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *Journal of High Speed Networks*, vol. 8, no.1, pp. 69-84, Jan. 1999.
2. S. Yao, B. Mukherjee, S.J.B. Yoo, and S. Dixit, "All-Optical Packet-Switched Networks: A Study of Contention Resolution Schemes in an Irregular Mesh Network with Variable-Sized Packets," *Proceedings, SPIE OptiComm 2000*, Dallas, TX, pp. 235-246, Oct. 2000.
3. I. Chlamtac, A. Fumagalli, L. G. Kazovsky, et al., "CORD: Contention Resolution by Delay Lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014-1029, June 1996.
4. R. Ramaswami and K.N. Sivarajan, "Routing and Wavelength Assignment in All-Optical Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 489-500, Oct. 1995.
5. A. S. Acampora and I. A. Shah, "Multihop Lightwave Networks: A Comparison of Store-and-Forward and Hot-Potato Routing," *IEEE Transactions on Communications*, vol. 40, no. 6, pp. 1082-1090, June 1992.
6. A. Bononi, G. A. Castanon, and O. K. Tonguz, "Analysis of Hot-Potato Optical Networks with Wavelength Conversion, " *IEEE Journal of Lightwave Technology*, vol. 17, no. 4, pp. 525-534, April 1999.
7. F. Forghieri, A. Bononi, and P. R. Prucnal, "Analysis and Comparison of Hot-Potato and Single-Buffer Deflection Routing in Very High Bit Rate Optical Mesh Networks," *IEEE Transactions on Communications*, vol. 43, no. 1, pp. 88-98, Jan. 1995.
8. V.M. Vokkarane, J.P. Jue, and S. Sitaraman, "Burst Segmentation: an Approach for Reducing Packet Loss in Optical Burst-Switched Networks," *Proceedings, IEEE International Conference on Communications (ICC) 2002*, New York, NY, vol. 5, pp. 2673-2677, April 2002.
9. Y. Xiong, M. Vanderhoute, and H.C. Cankaya, "Control Architecture in Optical Burst-Switched WDM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838-1854, Oct. 2000.
10. V.M. Vokkarane, G.P.V. Thodime, V.B.T. Challagulla, and J.P. Jue, "Channel Scheduling Algorithms using Burst Segmentation and FDLs for Optical Burst-Switched Networks," *Proceedings, IEEE International Conference on Communications (ICC) 2003*, Anchorage, AK, vol. 2 , pp. 1443 -1447, May 2003.
11. V.M. Vokkarane and J.P. Jue, "Prioritized Routing and Burst Segmentation for QoS in Optical Burst-Switched Networks," *Proceedings, Optical Fiber Communication Conference (OFC) 2002*, Anaheim, CA, WG6, pp. 221-222, March 2002.
12. J.S. Turner, "Terabit Burst Switching," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 3-16, 1999.
13. L. Tancevski, A. Ge, G. Castanon, and L. Tamil, "A New Scheduling Algorithm for Asynchronous, Variable Length IP Traffic Incorporating Void Filling," *Proceedings, Optical Fiber Communication Conference (OFC) 1999*, vol. 3, pp. 180-182, Feb. 1999.
14. C. Gauger, "Dimensioning of FDL Buffers for Optical Burst Switching Nodes," *Proceedings, IFIP Optical Network Design and Modeling (ONDM 2002)*, Torino, 2002.
15. A. Neukermans and R. Ramaswami, "MEMS Technology for Optical Networking Applications," *IEEE Communications Magazine*, vol. 39, no. 1, pp. 62-69, Jan. 2001.
16. L. Rau, S. Rangarajan, D.J. Blumenthal, H.-F.Chou, Y.-J. Chiu, and J.E. Bowers, "Two-hop All-optical Label Swapping with Variable Length 80 Gb/s Packets and 10 Gb/s Labels using Nonlinear Fiber Wavelength Converters, Unicast/Multicast Output and a Single EAM for 80- to 10 Gb/s Packet Demultiplexing," *Proceedings, Optical Fiber Communication Conference (OFC) 2002*, pp. FD2-1-FD2-3, Anaheim, CA, March 2002.