Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 48 (2005) 891–909
Computer Networks
www.elsevier.com/locate/comnet

On ordered scheduling for optical burst switching

M.H. Phùng [a,*], K.C. Chua [a], G. Mohan [a], M. Motani [a], T.C. Wong [b], P.Y. Kong [b]

[a] *Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117576, Singapore*
[b] *Institute for Infocomm Research, Singapore*

Received 4 September 2003; received in revised form 1 November 2004; accepted 22 November 2004
Available online 30 December 2004
Responsible Editor: A. Fumagalli

Abstract

Optical burst switching (OBS) is a promising optical networking paradigm for efficient transport of bursty IP traffic over wavelength division multiplexing (WDM) optical Internet networks. In OBS, the header of a burst is sent in advance of the data burst to reserve a wavelength channel at each optical switching node along the path. The nodes use a scheduling algorithm to assign wavelengths to incoming bursts. Our work is motivated from the observation that existing scheduling algorithms assign a wavelength to a burst when its header arrives at the node. Thus, information about other bursts whose headers arrive later is not available when the scheduling decision is made. This leads to sub-optimal scheduling decisions and unnecessary burst dropping. The key idea in our proposed algorithm, Ordered Scheduling, is to defer making the scheduling decision until just before the burst arrival in order to have full knowledge about other bursts. The effectiveness of the proposed algorithm is studied through simulation and the computational complexity and signalling overhead are analysed.

Keywords: Wavelength division multiplexing; Optical burst switching; Scheduling

1. Introduction

The bandwidth demand in the Internet is growing rapidly due to emerging multimedia applications such as video on demand, video conferencing and Internet telephony. In order to meet this bandwidth demand, wavelength division

* Corresponding author. Tel.: +65 68725709.
E-mail address: engp1627@nus.edu.sg (M.H. Phùng).

1389-1286/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2004.11.021

multiplexing (WDM) [1] has emerged as the transmission technology of choice for the next-generation Internet backbone. It provides enormous bandwidth with its ability to support up to hundreds of gigabit channels in a single fibre. However, this makes the mismatch between electronic processor speed (currently a few GHz) and the optical transmission rate more acute. Recently, all-optical WDM networks have received much attention because they can bypass the opto-electronic bottleneck in electronic routers.

Existing optical switching techniques can be broadly classified into optical circuit switching, optical packet switching and optical burst switching approaches. In the optical circuit switching approach [2], when a pair of routers need to communicate, a dedicated WDM channel, or lightpath, needs to be established first between them using the wavelength routing capability of the optical layer. The lightpath is released when the communication ends. Although this approach removes much of the opto-electronic bottleneck and layering overhead in the current Internet backbone, it may not be able to make efficient use of the bandwidth to support bursty traffic due to the coarse and dedicated bandwidth allocation. In optical packet switching [3,4], optical packets are sent along with their headers into the network without any prior reservation. Upon reaching an optical switch, a packet will be optically buffered while its header is extracted and processed electronically. Since optical packets from different connections can share the same link, a high utilisation level can be achieved through statistical multiplexing. However, this approach requires technologies that are currently immature and expensive such as optical synchronisation, buffering, and the extraction of headers from optical packets.

Optical burst switching (OBS) [5–8] is an optical networking paradigm that combines the best virtues of both the circuit switching and packet switching approaches. In an OBS network, packets are assembled into large bursts at network ingress and disassembled back into packets at network egress. When a burst is ready for transmission, the ingress node sends a header packet toward the egress node on a dedicated control channel to reserve resources at core nodes along the path. The data burst follows after an offset time in a data channel without waiting for an acknowledgement. The offset time is meant to allow enough time for processing the header and resource reservation. OBS achieves high link utilisation through statistical multiplexing. Also, the physical separation of transmission and switching of data bursts and their header packets helps to provide flexible electronic processing of headers at optical core nodes and end-to-end transparent optical paths for transporting data bursts.

In the literature, there are a number of OBS techniques that differ mostly in the way wavelengths[1] are reserved. In just-in-time (JIT) [7], an output wavelength is reserved as soon as a header packet arrives at a node and released only after a release signal is received. In Terabit Burst Switching [6], wavelengths are also reserved as the header packet arrives but burst length information is carried in the header packet to enable automatic release of the wavelengths. In the most popular technique, just-enough-time (JET) [5,8], offset information is put in the header packet in addition to burst length information. This enables a node to calculate the burst arrival time and reserve a wavelength just in time for the burst, saving some channel holding time between the header arrival and the burst arrival, which can be used for other bursts.

Since a WDM link normally has many wavelengths and burst arrival at a node is dynamic, a burst scheduling algorithm that chooses which of the available wavelengths to assign to an incoming burst is needed. This scheduling algorithm should be able to pack as many burst reservations onto a fixed number of wavelengths as possible in order to increase link utilisation. Such an algorithm is especially needed for the JET OBS technique, where the scheduling window is usually fragmented by burst reservations with different offsets. The general approach of existing scheduling algorithms is to schedule incoming bursts in the order

[1] Since a channel in WDM networks usually occupies the whole wavelength, the terms wavelength and channel are used interchangeably in this paper.

of their header arrivals. It implies that in a particular scheduling window, the node will schedule bursts with large offsets without knowledge of those with small offsets whose headers will arrive later. This may result in suboptimal wavelength allocation and unnecessary burst loss.

The focus of this paper is to develop a new burst scheduling algorithm called Ordered Scheduling that can overcome the above problem. The basic idea is to schedule bursts in the order of the burst arrivals instead of the header arrivals. The algorithm places the incoming reservations in a buffer and defers the scheduling until just before the actual burst arrives. By that time, the algorithm will have had full knowledge of all other burst reservations and thus be able to remove the negative effect of header arrival dynamics. We show through simulation that Ordered Scheduling can outperform traditional burst scheduling algorithms when the scheduling window is long and fragmented. Such a condition occurs when offset-based QoS schemes are used or Fibre Delay Line (FDL) buffers are present at core nodes, which is very common.

The rest of the paper is organised as follows. Section 2 presents a brief overview of the existing burst scheduling algorithms. In Section 3, the proposed burst scheduling algorithm, Ordered Scheduling, is described in detail. It includes a discussion on the computational complexity and signalling overhead of the algorithm. Section 4 presents the simulation results. Finally, concluding remarks are given in Section 5.

2. Related work

Most of the existing burst scheduling algorithms are developed for the JET OBS technique. This is because in JET, just enough time of a wavelength channel is reserved to cover the duration of a burst. That leaves several free time intervals between the scheduled reservations, which are called voids, to be used by bursts whose headers arrive later. This fragmentation of the scheduling window calls for efficient scheduling algorithms to maximise wavelength utilisation. Existing burst scheduling algorithms can be classified into two categories: without and with void filling.

Algorithms without void filling are proposed in [6,8], for example. They only maintain the unscheduled time for each data channel, or the time from which the channel is unscheduled. An incoming burst requesting reservation from time t is assigned a channel that has the unscheduled time closest to but not exceeding t (the latest available channel). The drawback of these algorithms is the inefficient use of data channels as the voids between data bursts are not utilised. They are suitable for the Terabit Burst Switching technique or JET technique with minimal offset time. The latter can be realised using OBS switches with input FDL buffers to compensate for header processing time.

Some initial scheduling algorithms with void filling are proposed in [8,9]. They are similar to the above algorithms except that they use suitable voids between scheduled bursts as well as unscheduled channels to schedule an incoming burst. The most popular variant is *Latest Available Unused Channel with Void Filling* (LAUC-VF) where the latest available unused channel, which is either an unscheduled channel or a suitable void, is assigned to an incoming burst. Because these algorithms can utilise the voids, they are more efficient than algorithms without void filling. They are suitable for JET OBS networks with extended offset time, which appears when output FDL buffers or offset-based QoS mechanisms are in place.

Since the above algorithms schedule bursts in the order of their header arrivals, they do not have knowledge of small-offset bursts when scheduling large-offset bursts. Thus, small offset bursts whose headers arrive later may be denied reservation unnecessarily. This is illustrated by the scheduling scenario in Fig. 1(a). The burst rescheduling approach [10] is proposed to solve this problem. It still schedules a burst immediately when its header arrives at a node. However, when the next header comes, the node may reschedule some existing reservations to other wavelengths to improve the schedulability of the new burst. This is possible because scheduling is done well in advance of actual burst arrivals. After rescheduling, the change in output wavelengths of the bursts is signalled to the next node via the control channel. This operation makes the burst scheduling more "compact"

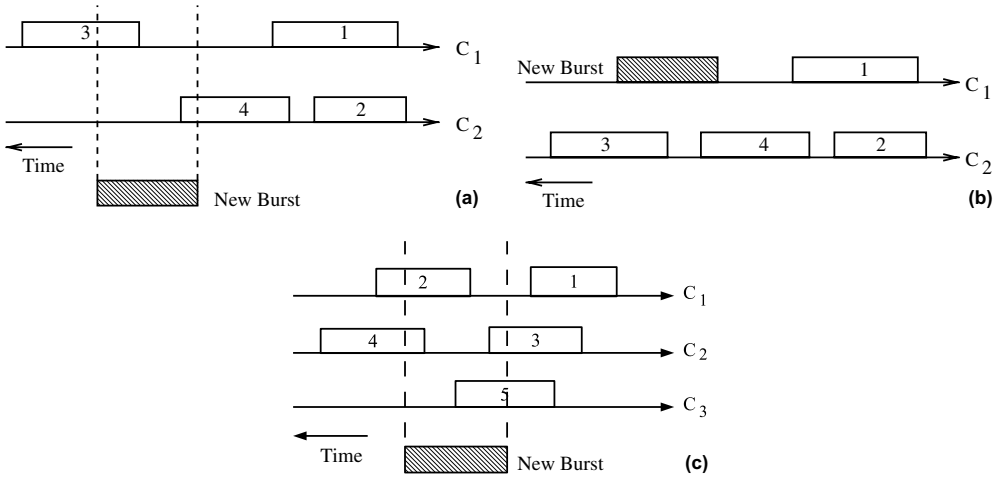

Fig. 1. Illustration of various burst scheduling scenarios. (a) LAUC-VF unnecessarily denies reservation to the new burst. (b) The new burst is scheduled by rescheduling burst 3. (c) Two rescheduling operations are required to schedule the new burst.

and increases wavelength utilisation. The benefit of burst rescheduling is illustrated in Fig. 1(b).

Although the burst rescheduling technique can improve wavelength utilisation, it still has some limitations. Burst rescheduling can be viewed as a re-optimisation exercise that is done when new information brought by newly arriving headers becomes available. To achieve the highest utilisation level, this exercise has to be carried out for every arriving header. This may cause a burst to be rescheduled multiple times, which leads to wastage of both computational resource and bandwidth on the control channel. Moreover, there may be some situations such as in Fig. 1(c) where multiple bursts need to be rescheduled in order to accommodate a new burst (i.e. multi-level rescheduling). With the number of wavelength channels per fibre typically in the order of hundreds, the computational complexity of multi-level rescheduling may be unacceptable.

3. Ordered Scheduling

3.1. Overview

Motivated by the observation in the previous section, Ordered Scheduling is proposed to optimise burst scheduling in OBS. We assume full wavelength conversion capability at an OBS node, as does most of the current OBS literature. In this algorithm, the scheduling of a burst consists of two phases. In the first phase, when a header packet arrives at a node, an admission control test is carried out to determine whether the burst can be scheduled. If the burst fails the test, it is dropped. Otherwise, a reservation object that contains the burst arrival time and duration is created and placed in an electronic buffer while the header is passed on to the next node. The buffer is in the form of a priority queue[2] with higher priority corresponding to earlier burst arrival time. The second phase starts just before the burst arrival time. Because of the priority queue, the reservation object of the incoming burst should be at the head of the queue. It is dequeued and a free wavelength is assigned to the burst. A special NOTIFY packet is immediately generated and sent to the next downstream node to inform it of the wavelength that the burst will travel on.

A simple example shown in Fig. 2 helps to illustrate the main concept of Ordered Scheduling. The left section of the figure shows the order of the incoming bursts and the order of the header pack-

[2] A priority queue is a data structure that always has the highest priority element at the head of the queue.

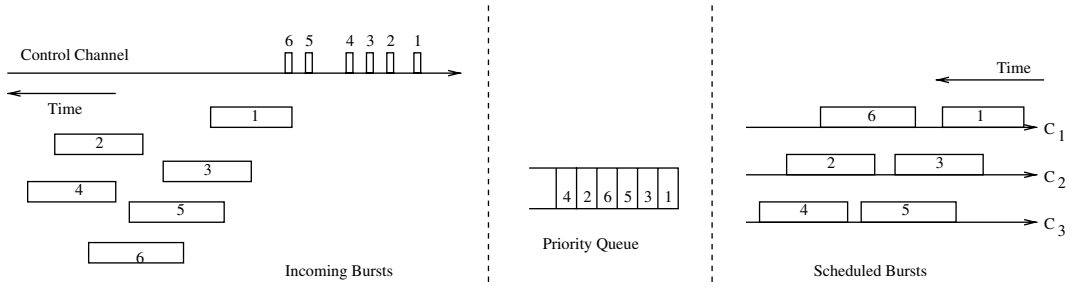

Fig. 2. Illustration of the main concept of Ordered Scheduling.

ets in the control channel. The middle section shows that the reservation objects are placed in the priority queue in the order of burst arrivals. Finally, the scheduled bursts are shown in the right section of the figure. The node simply dequeues a reservation object from the priority queue and assigns a free wavelength to it. Any free wavelength can be assigned to a newly dequeued reservation object since it has passed the admission control test. In the example, we use a round robin assignment because it is the easiest way to implement.

The admission control test for an output link without an FDL buffer is given below.

A burst requesting reservation for the time interval $[t_0, t_1]$ *can be scheduled on an output link with* M *wavelengths if* $\forall t \in (t_0, t_1)$*, the number of existing reservations containing* t *is no more than* $M - 1$.

(Note: A reservation for interval $[t_0, t_1]$ is said to contain t if $t_0 < t < t_1$.)

When an output link is equipped with an FDL buffer, which can be thought of as a collection of fibres (or FDLs) with different lengths, a node has the option of delaying a burst by routing it through one of the FDLs. In this case, the above admission control test is extended as follows. If a burst fails to reserve an output wavelength at its original arrival time t_0, the node searches through the FDLs in order of increasing length. Let the length of the FDL in consideration be D_{FDL}. The node first checks if the FDL is free during the interval $[t_0, t_1]$. If the FDL already has another reservation overlapping that interval, the node simply proceeds to the next FDL. Otherwise, it reserves the FDL for that interval. The node then executes the admission control test for the new reservation interval $[t_0 + D_{\mathrm{FDL}}, t_1 + D_{\mathrm{FDL}}]$. If the test succeeds, the burst is admitted and the search stops. Otherwise, the node undoes the reservation on the FDL and proceeds to the next FDL. If all the FDLs have been searched, the burst is dropped.

It should be noted that passing the admission control test is necessary but not sufficient for a burst to be scheduled. The test guarantees that at any infinitesimal time slot δt within the reservation interval $[t_0, t_1]$ of a burst, there exists a free wavelength. However, it does not guarantee that those free time slots are located on the same wavelength for the entire reservation interval, which is required for them to be usable by the new burst. The key to ensure that they are on the same wavelength is to schedule bursts in the order of their arrival times as is done in the second phase.

The admission control test is important because it prevents resource wastage due to over-admitting bursts. In Ordered Scheduling, header packets are passed on to the next node before scheduling takes place. Thus, without the admission control test, an incorrectly admitted burst will have its header forwarded to downstream nodes to make further reservations. However, the node that makes the incorrect admission will not be able to schedule the burst. Without having the optical switch configured for it, the burst will be lost upon arrival and resources reserved at downstream nodes will be wasted.

In comparing Ordered Scheduling with the current scheduling algorithms surveyed in Section 2, we note that Ordered Scheduling is optimal in the sense that given some previously admitted burst reservations, if an incoming burst reservation cannot be admitted by Ordered Scheduling,

it cannot be scheduled by any other scheduling algorithm. This is because by definition, if a new burst reservation fails the admission control test, there exists a time slot within its reservation interval in which all the data wavelengths are occupied. Therefore, the only way to schedule the new burst is to preempt some existing reservations. We will revisit the discussion on the optimality of Ordered Scheduling in Section 3.6.

3.2. Admission control test

The admission control test in Section 3.1 is presented in continuous form. However, this form may not be practical or feasible to implement. A simple solution would be to divide the time axis into slots. A burst that reserves any portion of a time slot, however small, will be considered as occupying the whole time slot. The admission control routine simply needs to keep track of the number of admitted bursts N_{occupied} that occupy each time slot and compare it to the total number of data wavelengths M. A new burst will be admitted only if $N_{\text{occupied}} < M$ for all the slots it will occupy. In the rest of the paper, this version will be called basic Ordered Scheduling. It is suitable if the optical switches in the network also operate in a slotted fashion as described in [8]. In that case, the time slots chosen by Ordered Scheduling should simply be set to be the same as the time slots of the underlying optical switches.

The basic slotted approach, however, may degrade the system performance if the underlying optical switches can operate in a truly asynchronous fashion. Due to its discrete nature, it does not consider the case where two bursts can occupy the same wavelength in a slot and thus may lead to unnecessary burst loss. This may be alleviated by having the slot size much smaller than the average burst size. However, that will increase the processing time and/or hardware complexity.

We describe here an enhanced version of the above slotted approach, which is called enhanced Ordered Scheduling. Instead of a single number to indicate the number of bursts occupying a time slot, the admission control routine now keeps three data entities for each time slot: (i) N_{total} is the total number of bursts that occupy the slot, whether wholly or partly; (ii) *heads* is the list of the start times of the bursts that have the start of their reservation periods fall into the slot, sorted in increasing order and (iii) *ends* is the list of the end times of the bursts that have the end of their reservation periods fall into the slot, sorted in increasing order. The bookkeeping of the two approaches is illustrated in Fig. 3.

When a header packet arrives at a node to request reservation, the admission control routine pretends that the burst has passed the test and updates the database of all the time slots involved, i.e. the time slots of the burst corresponding to the arriving header. N_{total} is incremented by one for each of the time slots. In addition, for the time slots containing the start or the end of the burst, an entry is added to *heads* or *ends*, respectively. The admission control routine then checks all the involved time slots. For a particular slot, if N_{total} is larger than the number of wavelengths M, entries in the *heads* list will be "matched" to those in the *ends* list to reduce the number of occupied wavelengths. A pair of bursts are considered matched for a given slot if the start time of the one in the *heads* list is no greater than the end time of the other in the *ends* list. The actual number of occupied wavelengths is N_{total} minus the number of matched pairs. If this number is smaller than M for all the involved slots, the new burst is schedulable and admitted. Otherwise, its header is

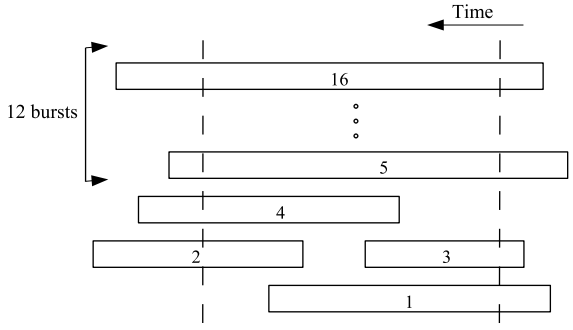

Fig. 3. Illustration of the bookkeeping for a typical time slot (with the slot size exaggerated and the unshown bursts occupying the whole slot) for the two implementations of the admission control test. For the basic version, $N_{\text{occupied}} = 16$. For the enhanced version, $N_{\text{total}} = 16$ and there are two entries in each of *heads* and *ends*.

dropped and its information previously inserted in the time slots' database is removed.

The matching operation is facilitated by the fact that *heads* and *ends* are kept in increasing order. The algorithm simultaneously goes from the beginning to the end on both lists, checking their entries against each other. Let i and j be the current indices on *heads* and *ends*. If $heads[i] \leqslant ends[j]$ then a match is recorded and i and j are incremented by one. Otherwise, only j is incremented to point to the next larger entry in *ends*. The process is repeated until either i or j passes the end of the list.

The formal description of the algorithm is presented in Algorithm 1. Denote $[t_0, t_1]$ as the requested reservation interval; *slot* as the object representing a particular time slot and s_0 and s_1 as the slots that contain t_0 and t_1, respectively. Also, let $slot.insert\_head(t)$ and $slot.insert\_end(t)$ be the functions that insert t into the sorted lists *heads* and *ends* of *slot*, respectively. The main test procedure uses three sub-functions $insert(t_0, t_1)$, *match*() and $remove(t_0, t_1)$. The first two sub-functions are presented below the main test procedure while the last one is omitted because it is similar to $insert(t_0, t_1)$.

Algorithm 1. Admission control test

(1) $accept \leftarrow$ **true**
(2) **for** $slot = s_0$ **to** s_1
(3) $slot.insert(t_0, t_1)$
(4) **if** $slot.N_{\text{total}} - slot.match() > M$
(5) $accept \leftarrow$ **false**
(6) **if** $accept$ = **false**
(7) **for** $slot = s_0$ **to** s_1
(8) $slot.remove(t_0, t_1)$

INSERT(t_0, t_1)
(1) $N_{\text{total}} \leftarrow N_{\text{total}} + 1$
(2) **if** *slot* contains t_0 **then** $insert\_head(t_0)$
(3) **if** *slot* contains t_1 **then** $insert\_end(t_1)$

MATCH()
(1) $i \leftarrow 0$
(2) $j \leftarrow 0$
(3) $matched \leftarrow 0$
(4) **while** Neither i nor j have passed the end of *heads* and *ends*, respectively
(5) **if** $heads[i] \leqslant ends[j]$
(6) $matched \leftarrow matched + 1$
(7) $i \leftarrow i + 1$
(8) $j \leftarrow j + 1$
(9) **return** *matched*

It is worth noting that the above algorithm will work properly even if some burst sizes are smaller than the slot size. In that case, some bursts may fall entirely within a single time slot as illustrated in Fig. 4. For each burst falling entirely within the time slot, N_{total} is incremented by one and one entry is added to each of *heads* and *ends*. In the example shown, $N_{\text{total}} = 4$ and there are three entries in each of *heads* and *ends*. The *match*() function treats each entry as a separate burst. It performs the matching operation as in Fig. 4(b) and returns three matches. So the number of occupied wavelengths is $N_{\text{total}} - match() = 1$, which is the correct result.

In terms of loss performance, we observe that enhanced Ordered Scheduling is optimal since it faithfully implements the test in continuous form. Its outperformance compared to the basic version is illustrated in Fig. 3 where the basic version reports that 16 wavelengths are occupied for the

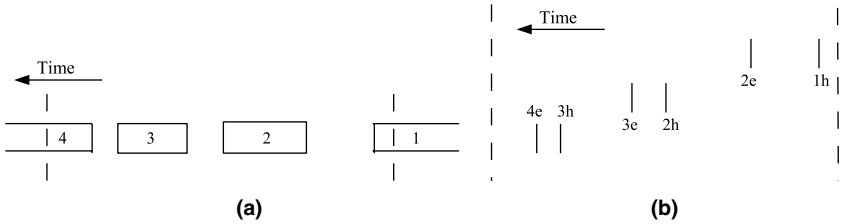

Fig. 4. Illustration of the matching operation when some bursts fall entirely within the slot. (a) Actual burst pattern; (b) burst pattern seen by the algorithm.

time slot while the enhanced version reports only 15 occupied wavelengths. The disadvantage of the enhanced version is that it is more complex. This will be explored in the next section.

3.3. Practical implementation and complexity analysis

As the burst rates in optical backbone networks are very high, the scheduling of a burst has to be done very quickly. To achieve that, parallel computation is usually necessary. In addition, complexity analysis is important because it offers insights into how various parameters are best configured. In this section, we will address these issues in the context of Ordered Scheduling.

3.3.1. Admission control test

The slotted structure of the two admission control implementations is particularly suitable for parallel processing since each slot of a burst is processed independently. Let S be the maximum number of slots in the scheduling window. A simple parallel solution is to have S processing elements in the admission control unit with each processing element responsible for one slot. When a burst header arrives, the processing elements corresponding to the slots covered by the burst will compute the admission control test simultaneously.

The time complexity analysis for the basic and enhanced versions of the admission control test is as follows. For basic Ordered Scheduling, a processing element needs to perform at most one comparison and one update of N_{occupied} per burst. Therefore, the required processing time is constant and takes less than 1 ns assuming a processing speed in the order of 10^9 operations per second. For enhanced Ordered Scheduling, the processing element also needs to perform one comparison and one update. In addition, it needs to do the matching operation when necessary. Assuming that the slot size is smaller than the minimum burst size, the number of elements in *heads* and *ends* is M in the worst case. So the worst case complexity of the matching operation is $O(M)$. Also, the update of *heads* and *ends* at the two slots at the two ends of a burst takes $O(\log M)$. Therefore, the overall worst case time complexity is $O(1) + O(M) + O(\log M) = O(M)$. In a normal case, however, the size of *heads* and *ends* is about M/K where K is the average number of slots per burst. Hence, the average complexity is $O(M/K)$ per matching operation. The overall average complexity is $O(1) + O(M/K) + O(\log M) = O(M/K + \log M)$. Let us consider an example with $M = 256$ and $K = 16$, *heads* and *ends* will have about 16 elements on average. A worst case estimate of the processing time is 50 ns, which includes the execution of *match*() and *remove*(t_0, t_1). The average processing time is much smaller as *match*() and *remove*(t_0, t_1) are only executed in heavy loading conditions.

The required number of processing elements is inversely proportional to the slot size, or proportional to the average number of slots per burst K. Therefore, although basic Ordered Scheduling has the advantage of fast processing compared to the enhanced version, its drawback is that it requires a much larger number of processing elements to ensure good dropping performance. For enhanced Ordered Scheduling, there is a tradeoff between processing speed and hardware complexity. A small value of K will reduce the required number of processing elements but will lead to longer execution time and vice versa.

For LAUC-VF, it is possible to perform a parallel search across the wavelengths to find all the unused wavelengths. Then the search results are compared to each other to find the latest available one. These operations can be performed in $O(\log M)$ time, which is better than enhanced Ordered Scheduling and worse than basic Ordered Scheduling. In terms of hardware complexity, LAUC-VF requires one processing element for each wavelength with each processing element being fairly complex. If the number of wavelengths per link is large, which is usually the case, the hardware requirement for LAUC-VF will be larger than Ordered Scheduling.

3.3.2. Priority queue

The queueing operations on the priority queue are common to both versions of Ordered Scheduling. Its complexity depends on the specific implementation of the underlying priority queue. Some efficient implementations of priority queues using

pipelined heap are reported in the literature [11,12]. They have O(1) time complexity with regard to queue size. Implemented on conservative technologies such as 0.35-μm and 0.18-μm CMOS, they can achieve up to 200 million queueing operations per second for queues with up to 2^{17} entries. The queue size depends on the size of the scheduling window, which in turn depends on offset times and FDL buffer depth, and the burst arrival rate. We observe that the above priority queue implementations can accommodate any queue size of practical interest. The queue size only affects the amount of required memory.

3.3.3. Overall time complexity

The computational work in admission control and priority queue operations can be pipelined. That is, as soon as the admission control routine finishes with a header and passes it to the priority queue, it can handle the next header while the first header is being enqueued. Therefore, the overall complexity is the maximum of the two parts. With parallel processing, the time complexity for basic Ordered Scheduling is O(1). The worst case and average time complexities of enhanced Ordered Scheduling are $O(M)$ and $O(M/K + \log M)$, respectively.

3.4. Timing in Ordered Scheduling

The timing in an operation cycle of Ordered Scheduling is shown in Fig. 5. At the beginning of an operation, when a header packet arrives at node A, the admission control test takes t_{admit}. If the burst reservation is successfully admitted, its header is sent to the next downstream node B while the reservation object is placed in the priority queue, which takes t_{queue}. At a suitable time, the object is removed from the queue, which also takes t_{queue} since for most implementations of the priority queue, enqueue and dequeue operations take approximately the same amount of time. It takes t_{NOTIFY} to transmit the NOTIFY packet at node A and to receive it at node B. At both nodes, the optical switching matrices are configured t_{config} before the burst arrival. There is a guard time t_{guard} between the receipt of the NOTIFY packet and the start of the optical switch configuration.

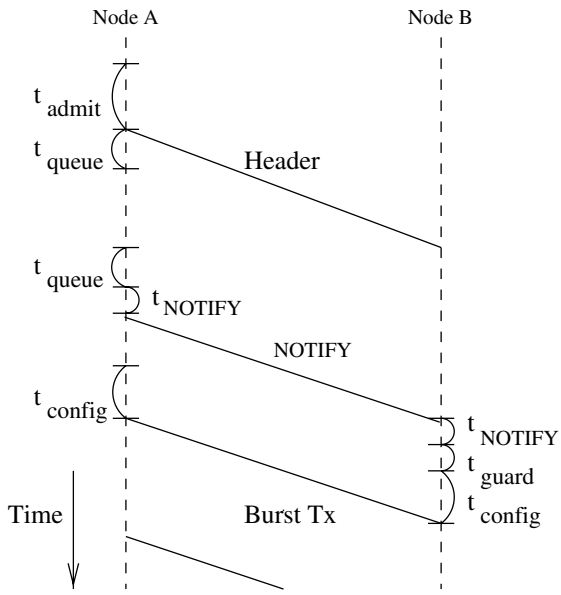

Fig. 5. Timing diagram for scheduling a burst.

This is to ensure that timing variations in various operations will not cause the NOTIFY packet to arrive late for the optical switch configuration. One of those timing variations may result from a large number of burst arrivals in a short interval. Without t_{guard}, this could overload the scheduler and render it unable to make scheduling decisions in time for optical switch configuration at the downstream node.

From the timing diagram, we can calculate the minimum time required to schedule a burst under Ordered Scheduling as

$$T_0 = t_{\text{admit}} + 2t_{\text{queue}} + 2t_{\text{NOTIFY}} + t_{\text{guard}} + t_{\text{config}}.$$

We attempt an estimate for T_0 here. From the previous section, we have $t_{\text{admit}} = 50$ ns in the worst case and $t_{\text{queue}} = 5$ ns. For t_{config}, according to [13], the semiconductor optical amplifier (SOA) technology can achieve a switching time t_{config} of 1 ns or less. For t_{NOTIFY}, assuming 8-byte NOTIFY packets as in Section 3.5, it will take $t_{\text{NOTIFY}} = 6.4$ *ns to transmit or receive a NOTIFY packet at 10 Gb/s. Assuming a 10% guard time, the estimate for* T_0 is 80 ns in the worst case. With rapid advances in electronics, we expect the figure to go down significantly in the near future.

T_0 can also be taken as the minimum required time offset of a burst. Of the time components on the right hand side of the above equation, t_{admit} and t_{queue} are variable. Therefore, in setting the minimum offset time, some upper bounds on t_{admit} and t_{queue} should be used. If for some reasons, t_{admit} or t_{queue} exceed their upper bounds, an arriving burst may have an offset smaller than T_0. In that case, if it passes the admission control test, the node may bypass the priority queue and go straight to assigning a wavelength to it. The reason is because its offset is so small, it would certainly end up at the head of the priority queue had it been put into the queue.

3.5. Signalling overhead

Compared to traditional burst scheduling schemes, there is more signalling overhead in Ordered Scheduling due to the need to send the NOTIFY packets. One NOTIFY packet is required for every burst. Therefore, at a glance, the signalling load appears to be doubled. However, since the size of a NOTIFY packet is much smaller than that of a header packet, the increase in signalling load is much smaller. In this section, we will estimate the signalling load increase and explore ways to reduce it.

The control packet formats in normal scheduling schemes and in Ordered Scheduling are shown in Fig. 6. In the example, High Level Data Link Control Protocol (HDLC) is used as the data link protocol. The address field in HDLC header is omitted and the control field uses 8 bits. In the header packet format, the offset and burst size fields both occupy 16 bits, which allow a resolution in the order of nanosecond to be specified. The wavelength ID (WID) field is set at 10 bits to accommodate links with up to 1024 wavelengths. Finally, the burst ID (BID) field is used to assign a unique ID to each burst in one scheduling window. Its size of 14 bits allows up to 16,384 unique burst IDs.

The increase in signalling load due to the use of NOTIFY packets can be calculated as follows. If we use a label stack depth $D = 3$, the length of a header packet will be $H = 23$ bytes in both normal scheduling schemes and Ordered Scheduling and

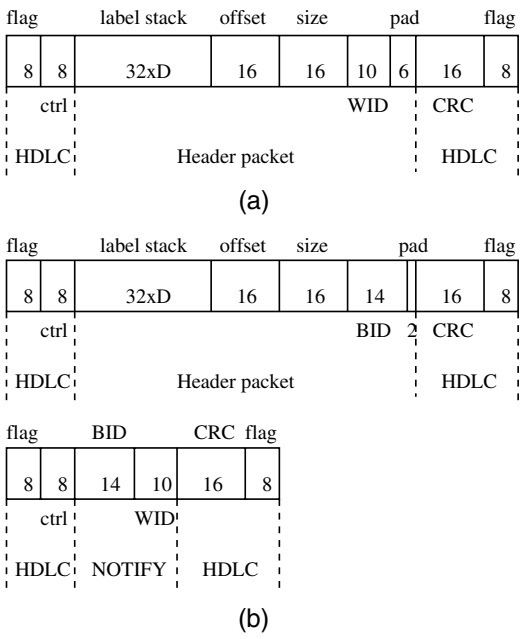

Fig. 6. Example of control packet formats. (a) Normal header packet format; (b) control packet formats in Ordered Scheduling.

the length of a NOTIFY packet will be $H_N = 8$ bytes. The percentage increase is

$$\frac{H_N}{H + H_N} = \frac{8}{23 + 8} = 0.2581 = 25.81\%.$$

It can be observed from Fig. 6 that the HDLC header and trailer occupy a large portion of a frame carrying a NOTIFY packet. Thus, higher efficiency can be achieved if several NOTIFY packets are carried in one HDLC frame. In order to satisfy the strict timing requirement detailed in Section 3.4, we need to start dequeueing bursts at node A earlier than the deadline dictated by the timing diagram. The NOTIFY packets are then collected and sent in a batch to node B. For example, assuming an average burst arrival rate of 10^7 bursts per second, bursts will need to be dequeued 1 μs earlier to collect 10 NOTIFY packets. The length of the HDLC frame carrying 10 NOTIFY packets will be $H_N = 5 + 3 \times 10 = 35$ bytes. So the percentage of increase in signalling load is now

$$\frac{H_N}{10H + H_N} = \frac{35}{10 \times 23 + 35} = 0.1321 = 13.21\%.$$

3.6. Ordered Scheduling from the queueing theory perspective

In the literature to date, most performance analyses of OBS schemes such as those in [14–16] have made use of queueing theory. In a queueing model of an OBS node, output wavelengths become servers, and when a burst is being transmitted, it is being "served". The current approach is to replace the offset times and advanced reservation in OBS with preemptive priority in the queueing model. To illustrate this approach, consider two traffic classes 1 and 2 with class 1 having a larger offset. Also, assume further that the difference in offset times is larger than the maximum burst duration of class 2. Under these conditions, bursts from class 1 always win those from class 2 in inter-class resource contention because headers of class 1 bursts always arrive before those of class 2 bursts. The same effect can be achieved if we consider the headers do not exist and give class 1 bursts strict preemptive priority over class 2 bursts.

The above queueing model assumes that a burst is not tied to any particular server prior to its being served. This assumption only holds if the scheduling decision is made at the burst arrival instant, which is what our proposed algorithm does. Thus, only Ordered Scheduling can achieve the theoretical upper bound in blocking performance set by the queueing model. To illustrate this from the perspective of the queueing model, consider the burst arrival pattern in Fig. 7 that is served by two channels. Bursts 1 and 2 have a larger offset time than bursts 3 and 4. Hence, in the corresponding queueing model, bursts 1 and 2 belong to class 1, which has preemptive priority over bursts 3 and 4 of class 2. When burst 2 arrives at the node, burst 4 is being served by server S_1 (channel 1) while server S_2 is free. In this situation, the queueing model and Ordered Scheduling would assign burst 2 to server S_2. However, the LAUC-VF algorithm assigns burst 2 to server S_1 instead and preempts burst 4. The reason for this mistake is because LAUC-VF schedules a burst when its header arrives at the node. Therefore, when burst 2 is scheduled, LAUC-VF has no knowledge about burst 4 whose header will arrive later.

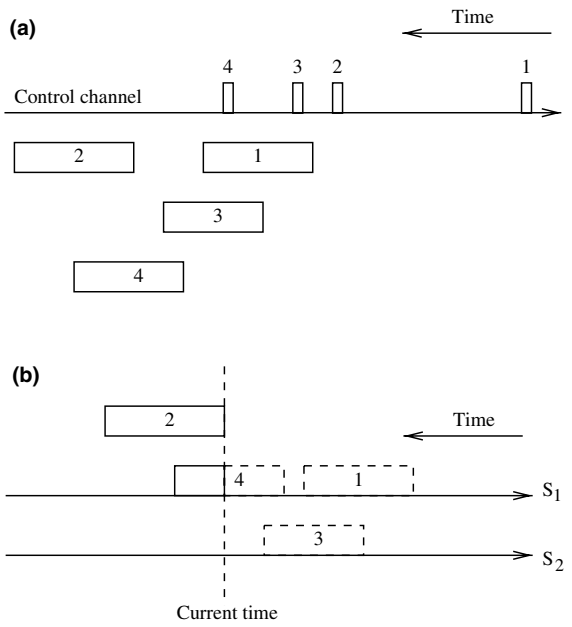

Fig. 7. Illustration of the queueing model approach to analysing OBS. (a) OBS with advanced reservation; (b) corresponding queueing model.

4. Simulation study

In this section, we study the burst dropping performance of Ordered Scheduling through simulation. Both versions of Ordered Scheduling are investigated. The slot sizes are 1 μs and 0.1 μs for the enhanced and basic versions, respectively, unless otherwise stated. We also simulate LAUC-VF under the same condition for comparison. LAUC-VF is chosen as a performance benchmark because it is one of the most efficient burst scheduling algorithms that schedule bursts as their headers arrive. Each simulation is run 10 times and each run ends after 10^6 bursts are generated or 10^3 lost bursts are recorded, whichever later. We plot the results with error bars representing 95% confidence intervals; however, in some cases, the deviation from the mean is so small that the error bars are not visible.

The traffic model in use is similar to that in [8]. Specifically, each ingress node receives an IP packet stream from adjacent access networks. The IP packet length distribution is modelled

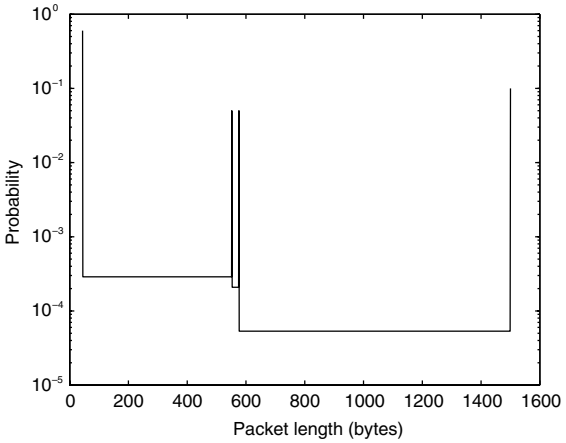

Fig. 8. Probability distribution of packet length.

according to that reported in [17] and is shown in Fig. 8. Let l be the packet length in bytes. We have $P[l = 44] = 0.6$, $P[44 < l < 552] = 0.145$, $P[l = 552] = 0.05$, $P[552 < l < 576] = 0.005$, $P[l = 576] = 0.05$, $P[576 < l < 1500] = 0.05$ and $P[l = 1500] = 0.1$. IP packets arrive according to a Poisson process. This choice of packet arrival distribution is justified by a study in [18], which reports that IP traffic tends towards Poisson at very high speed links and very short time scale.

The burst assembly algorithm is a simple time-based algorithm with a time limit T_{limit}. There are separate assembly queues for each egress node and incoming IP packets choose a queue with equal probability. When the first IP packet that forms a burst arrives at an assembly queue, a timer is started from zero. Subsequent IP packets are appended to the assembly queue. A burst will be created when the timer exceeds T_{limit}. In the experiments, we set the time limit T_{limit} such that the maximum burst duration is 2.5 μs. So the average number of slots per burst are approximately 2 and 20 for enhanced and basic Ordered Scheduling, respectively. Another aspect of the assembly mechanism is that we assume no packet framing overhead inside a burst and guard bands at the head and the end of a burst. That is, the size of a burst is exactly the sum of all packets inside the burst.

The simulation study consists of three sets of experiments. The first two sets are carried out for a topology with a single core node. They aim to investigate the effects of traffic conditions and hardware configurations on the performance of the algorithms, respectively. The final experiment set is carried out for a whole network to investigate the effect of network topology on the performance trend among the algorithms.

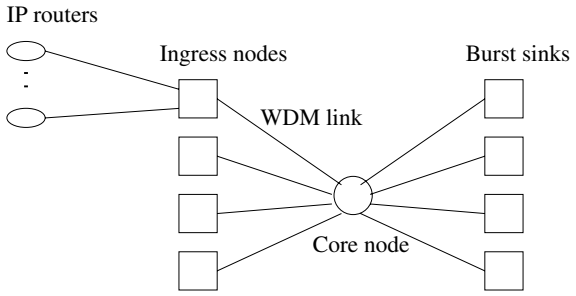

Fig. 9. Simulated OBS topology for a single core node.

The simulation topology for the first two sets of experiments is shown in Fig. 9. There are four ingress nodes and four burst sinks connected to the core node. The burst sinks represent egress nodes in a real network. No burst dropping is assumed on the links between the ingress nodes and the core node. It only occurs on the output links of the core node. Since IP packets are destined for each sink with equal probability, the average offered loads on each output link of the core node are equal. The reported burst dropping probabilities are the average values on the four output links.

4.1. Effects of traffic conditions

In this set of experiments, the configuration is as follows. The links connecting the OBS nodes are made up of a single optical fibre per link. Each optical fibre has eight data wavelengths. The number of wavelengths for control and signalling purposes is assumed to be large enough so that no control message will be lost. The core node has an FDL buffer with six FDLs of lengths $5\ \mu s, 10\ \mu s, \ldots, 30\ \mu s$. Except for the number of wavelengths per link, this represents a typical network configuration. The number of wavelengths per link is chosen to be small so that the simulation time will not become prohibitively long. In the

next section, we will investigate the effect of this parameter on the performance of the algorithms.

Firstly, we examine the effect of varying offered load to the core node, which is defined as the ratio between the total average rate of incoming IP packets in bits per second (bps) and the total output link capacity of the core node also in bps. For this experiment, two offset-based QoS classes as described in [14] with equal loading are used. Class 2 is given higher priority than class 1 by assigning an extra offset of 3 μs. This offset difference is larger than the maximum burst size so that full isolation between the two classes is achieved. The arrival rate ranges from 3.3 bursts per μs to 4.15 bursts per μs, or from 0.74 to 0.92 in terms of offered load. Offered loads lower than 0.74 are not considered because they would make the loss probability of class 2 too small to measure through simulation. On the other hand, offered loads larger than 0.92 would make the loss probability of class 1 too large to be of practical interest.

The simulation results are plotted in Fig. 10. They show that the dropping probabilities increase with increasing offered load, which is natural. Among the algorithms, enhanced Ordered Scheduling has the best dropping performance followed by basic Ordered Scheduling and then LAUC-VF. This order of dropping performance among the algorithms is as expected based on the discussion in the previous sections. The order of performance

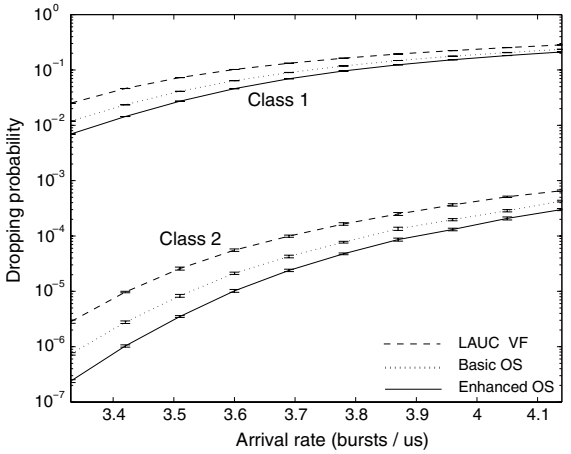

Fig. 10. Burst dropping performance under different traffic loading.

is the same in virtually all of the following experiments so we will not comment on it again except when the order is different. We note that the differences in performance are greater at lower load. This is because at low load, there are more free wavelengths to choose from to assign to an incoming burst reservation and LAUC-VF is more likely to make suboptimal wavelength assignment decisions due to incomplete knowledge of other burst reservations. Between the two classes, we observe that the performance improvement of Ordered Scheduling over LAUC-VF is greater for class 2 than it is for class 1. The reason for this is also related to loading. Since full isolation is achieved between the two classes, the effective loading for class 2 traffic is only half of that for class 1 traffic. So as the above reasoning goes, the improvement for class 2 is larger.

The effect of traffic class composition is considered next. We use the same traffic parameters as above except that the overall offered load is fixed at 0.9 and the offered load of each class is varied. As the proportion of class 1 traffic varies from 0 to 1, we observe in Fig. 11(a) that the overall traffic loss rate follows a bell-shaped curve, which is slightly tilted towards the left. The dropping probabilities peak when the burst rates from the two classes are comparable and are at the lowest at the two extremities where traffic from only one class is present. This effect can be explained from the queueing model point of view. As the traffic composition becomes more balanced, more class 1 bursts are preempted by those from class 2. When burst B_1 from class 1 is preempted by burst B_2 from class 2, the effective size of B_2 is its actual size plus the portion already served of B_1. As mentioned in [16], if the burst size distribution is not exponential, that will increase the effective burst size and the burst loss rates. This negative effect of burst preemption is present in all the algorithms, unlike the fragmentation of the scheduling window that only affects LAUC-VF. We also note that at the two extremes when there is only one traffic class, FDL buffers in the core node can still delay bursts and create fragmentation in the scheduling window. Therefore, there are performance differences among the algorithms even when there is only one traffic class.

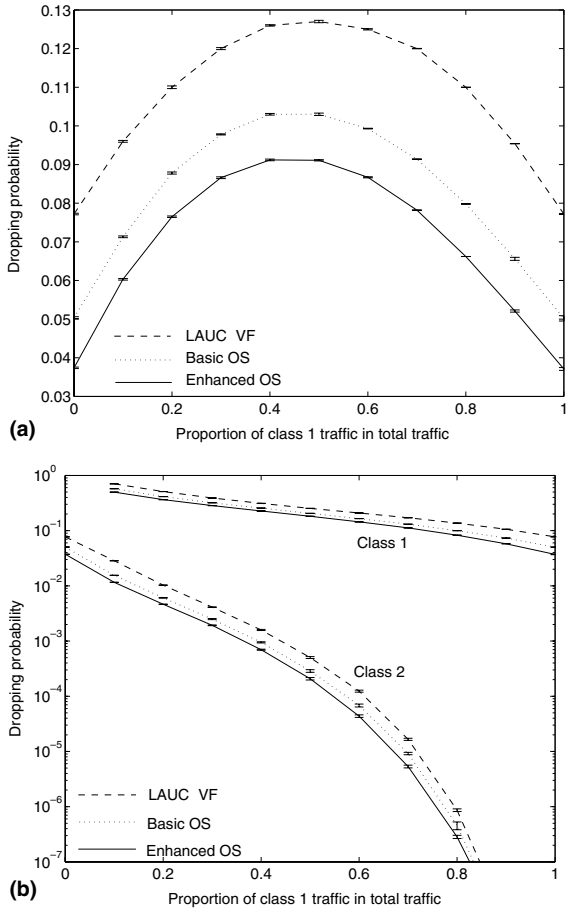

Fig. 11. Effect of traffic composition on burst dropping performance. (a) Overall performance; (b) performance of the two classes.

The loss rates of individual classes are shown in Fig. 11(b). It is seen that as the proportion of low priority traffic increases, the loss rates of both classes drop. For class 1, preemption by class 2 bursts make up a large part of its burst loss. Therefore, when there is less class 2 traffic, preemption occurs less frequently, which leads to the drop in class 1 burst loss. For class 2, the only cause for burst loss is intra-class contention since it is fully isolated from class 1. Thus, when its traffic rate decreases, contention rate rapidly decreases and so does the burst loss. This result implies that very low burst dropping probability can be achieved for high priority traffic even though the overall utilisation is high.

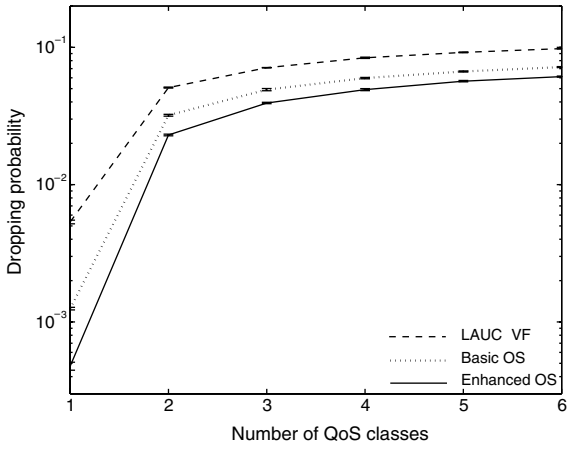

Fig. 12. Overall dropping performance with different number of QoS classes.

The final traffic parameter to be investigated is the number of QoS classes. In this experiment, the overall offered load is 0.8 and all traffic classes have equal loading. We plot the overall dropping probabilities in Fig. 12. It shows that the overall dropping probability increases as the number of classes increases. This is as expected because as the number of classes increases, the scheduling window is more fragmented, which results in increasing loss probability. A notable aspect is the large increase in loss probabilities moving from one to two classes. This is caused by the large increase in the degree of fragmentation in the scheduling window when moving from one class to two classes.

4.2. Effects of hardware configuration

In the first experiment of this section, we study the impact of FDL buffer depth on the performance of the algorithms. Two kinds of data traffic are considered: one with a single QoS class and the other with two offset-based QoS classes. This is because the effects on dropping performance are slightly different between having one and two QoS classes. The offered loads for both cases are set at 0.8. The FDL buffer in use consists of a number of FDLs according to the buffer depth. The lengths of the FDLs are regularly spaced starting from 5 μs with length spacing being 5 μs.

From Fig. 13, we see that the overall trend is improving loss performance with increasing number of FDLs. This is because when an FDL buffer is introduced, if a node cannot schedule a burst at its original arrival time, the node can try delaying the burst and scheduling it at a later time. The larger the number of FDLs there are in a buffer, the more options the node has in delaying bursts, which improves dropping performance. We also note that the curves for LAUC-VF tend to level off. This can be explained by the fact that the scheduling window is increasingly fragmented as more FDLs are introduced. For LAUC-VF, this negative effect opposes and neutralises the beneficial effect of having more FDLs, which explains the levelling off of the curve for LAUC-VF. Ordered Scheduling, on the other hand, is not affected due to its deferment of scheduling decisions. The above effect is more pronounced in Fig. 13(a) than it is in Fig. 13(b) because having two offset-based QoS classes already introduces significant fragmentation in the scheduling window so the additional fragmentation caused by more FDLs has less effect.

The impact of slot size on the performance of basic Ordered Scheduling is studied next. For this and the remaining experiments, input traffic with two QoS classes is used. In this experiment, the loss performance of basic Ordered Scheduling with different slot sizes is measured at an overall offered load of 0.6 and compared to enhanced Ordered Scheduling and LAUC-VF. The results are plotted in Fig. 14. Since the performance of the latter two algorithms is not affected by slot size, their loss curves show up as horizontal lines. On the other hand, as the slot size gets larger, the dropping performance of basic Ordered Scheduling rapidly worsens due to its discrete implementation of the admission control test. At a slot size of 1 μs, which is what is used by enhanced Ordered Scheduling, the dropping probability of basic Ordered Scheduling is nearly three orders of magnitude larger than enhanced Ordered Scheduling. These results confirm the necessity to use much smaller slot sizes

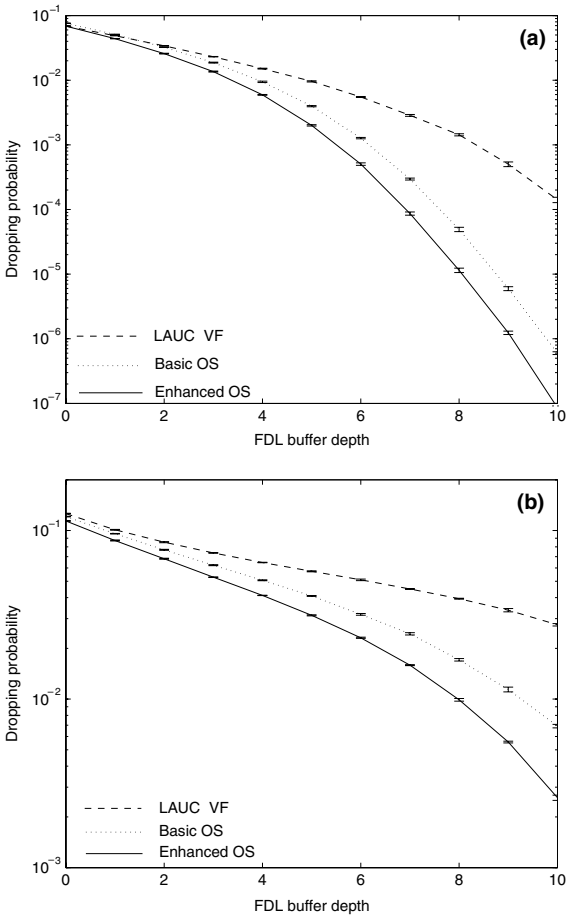

Fig. 13. Burst dropping performance with different buffer depth. (a) Single QoS class; (b) two QoS classes.

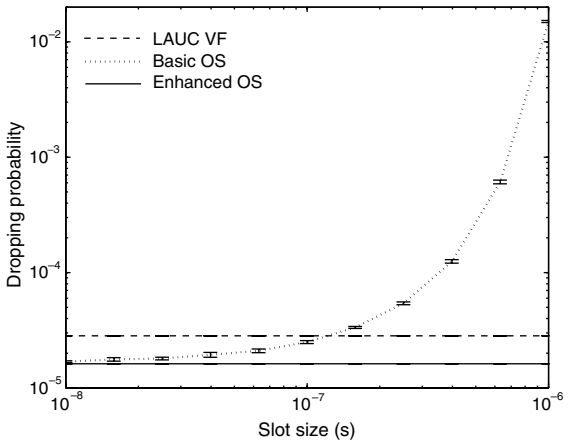

Fig. 14. Burst dropping performance of basic Ordered Scheduling with different slot size.

for basic Ordered Scheduling compared to enhanced Ordered Scheduling.

The final experiment in this section investigates the effects of the number of wavelengths per link on the performance of the algorithms. We include the performance results of a non-void filling scheduling scheme called Horizon in [6] or LAUC in [8]. The purpose is to see how its performance compares to those of other void filling algorithms at different numbers of wavelengths. We measure their dropping probabilities at an overall offered load of 0.8 and different numbers of wavelengths per link and plot the results in Fig. 15. It shows that the overall trend is decreasing loss probabilities with increasing number of wavelengths per link. This is the direct result of an OBS switch behaving like an $M|M|k|k$ loss system as described in [14,16]. We also observe that the performance of the Horizon scheme is poor when the number of wavelengths is low but gets very close to that of LAUC-VF when the number of wavelengths is high. Among the void filling algorithms, we see that the relative performance between enhanced Ordered Scheduling and LAUC-VF remains the same. However, the relative performance of basic Ordered Scheduling compared to the enhanced version gradually decreases as the number of wavelengths per link increases. This is also due to the discrete nature of basic Ordered Scheduling.

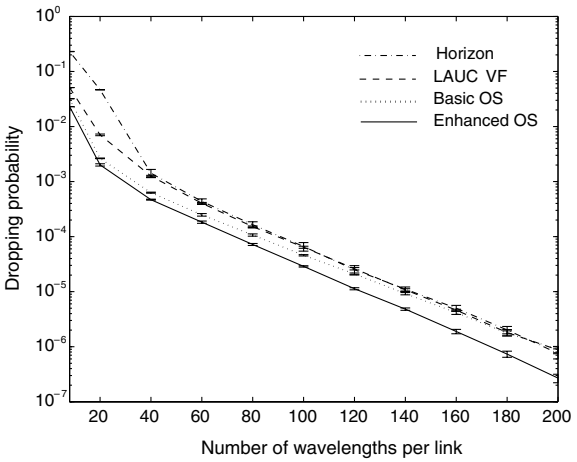

Fig. 15. Burst dropping performance with different number of wavelengths per link.

As a slot handles more and more bursts, the chance that basic Ordered Scheduling over-reports the number of occupied wavelengths as illustrated in Fig. 3 increases. From this experiment and the previous one, we note that the performance of Ordered Scheduling depends on the ratio between the number of slots per burst and the number of wavelengths per link.

4.3. Simulation study for a whole network

We now simulate the three scheduling algorithms in a whole network setting to see if the network topology affects the performance trend among the algorithms. For this experiment, the topology in Fig. 16, which is a simplified topology for the US backbone network, is used. The topology consists of 24 nodes and 43 links. The average node degree is 3.6. Shortest path routing is used to determine the transmission paths among nodes and the average hop length of the paths is 3. For simplicity, the propagation delays between adjacent nodes are assumed to have a fixed value of 10 ms. The links are bi-directional, each implemented by two uni-directional links in opposite directions.

The input traffic and hardware configuration for each node remain the same, i.e. two offset-based QoS classes, eight wavelengths per link and six FDLs per output link at each node. Each node has 23 separate assembly queues, one for every other node. An incoming IP packet to a node enters one of the queues with equal probability.

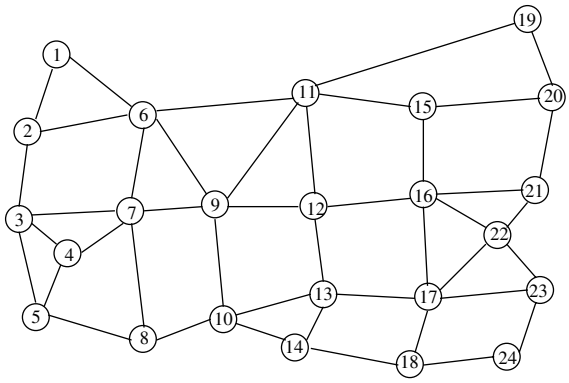

Fig. 16. Simulation topology for whole network.

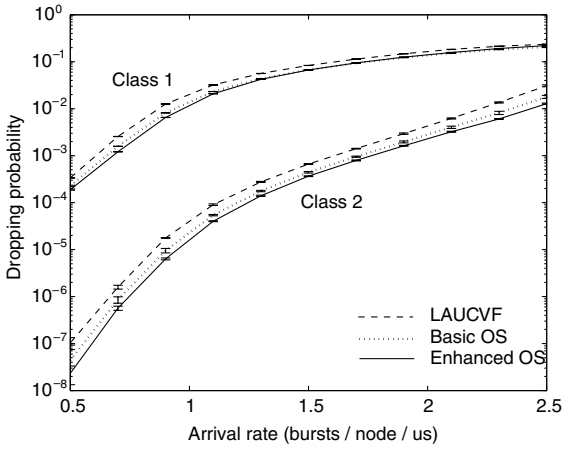

Fig. 17. Burst dropping performance for whole network under different traffic loading.

The IP packet arrival rates are the same for every node. The header processing time per node is assumed to be $\Delta = 1$ μs. For a path with H hops, the initial offset times assigned for bursts of classes 1 and 2 are $\Delta \cdot H$ μs and $\Delta \cdot H + 3$ μs, respectively.

We plot the dropping probabilities for the algorithms against the offered load to the network. The offered load is measured in terms of the number of departing bursts per node per μs. The simulation results are shown in Fig. 17. We observe that the performance trend is similar to that in Fig. 10. The order among the algorithms remains the same, i.e. enhanced Ordered Scheduling has the best performance, followed by basic Ordered Scheduling and LAUC-VF. We also see that the arrival rates used in this experiment are much smaller than those used in Fig. 10 but the ranges of the dropping probabilities are approximately the same. This is because in a network environment, many paths may converge at some nodes, causing bottlenecks. The offered loads to those bottlenecked nodes are much larger than the average offered load to the network and most of the burst loss in the network is concentrated there.

5. Conclusion

In this paper, we have proposed a new scheduling algorithm called Ordered Scheduling for optical burst switching networks with the objective of improving burst dropping performance while keeping the computational and signalling overheads manageable. Two versions of Ordered Scheduling are available: the basic version with discrete operation and the enhanced version with continuous operation. Extensive simulation studies show that the two versions of Ordered Scheduling outperform LAUC-VF, which is one of the most efficient existing scheduling algorithms, when the scheduling window is large and fragmented. Between the two versions, enhanced Ordered Scheduling always achieves good dropping performance in all traffic conditions and hardware configurations; the performance of basic Ordered Scheduling is dependent on the number of slots per bursts and the number of wavelengths per link.

Implementation and complexity issues are also discussed in the paper. We show that the proposed algorithm is particularly suitable for parallel implementation due to its slotted structure. When the number of wavelengths per link is large, the computational overhead is shown to be less than LAUC-VF. The proposed algorithm does introduce additional signalling overhead but it is small compared to the gain in dropping performance. Finally, the proposed algorithm is implemented entirely in the control and signalling domain. Since this is an electronic domain, no significant problem is expected when migrating from other scheduling algorithms.

Acknowledgement

M.H. Phùng is supported by the Singapore Millennium Scholarship number 2002SMS2300.

References

[1] C. Brackett, Dense wavelength division multiplexing networks: principles and applications, IEEE Journal on Selected Areas in Communications 8 (6) (1990) 948–964.
[2] I. Chlamtac, A. Ganz, G. Karmi, Lightpath communications: An approach to high bandwidth optical WAN's, IEEE Transactions on Communications 40 (7) (1992) 1171–1182.

[3] C. Guillemot et al., Transparent optical packet switching: The European ACTS KEOPS project approach, IEEE/OSA Journal of Lightwave Technology 16 (12) (1998) 2117–2134.
[4] D.K. Hunter, I. Andonovic, Approaches to optical Internet packet switching, IEEE Communications Magazine 38 (9) (2000) 116–122.
[5] C. Qiao, M. Yoo, Optical burst switching—a new paradigm for an optical Internet, Journal of High Speed Networks 8 (1) (1999) 69–84.
[6] J.S. Turner, Terabit burst switching, Journal of High Speed Networks 8 (1) (1999) 3–16.
[7] J.Y. Wei, R.I. McFarland Jr., Just-in-time signaling for WDM optical burst switching networks, IEEE/OSA Journal of Lightwave Technology 18 (12) (2000) 2019–2037.
[8] Y. Xiong, M. Vandenhoute, H.C. Cankaya, Control architecture in optical burst-switched WDM networks, IEEE Journal on Selected Areas in Communications 18 (10) (2000) 1838–1851.
[9] L. Tančevski, S. Yegnanarayanan, G. Castanon, L. Tamil, F. Masetti, T. McDermott, Optical routing of asynchronous, variable length packets, IEEE Journal on Selected Areas in Communications 18 (10) (2000) 2084–2093.
[10] S.K. Tan, G. Mohan, K.C. Chua, Algorithms for burst rescheduling in WDM optical burst switching networks, Computer Networks 41 (2003) 41–55.
[11] R. Bhagwan, B. Lin, Fast and scalable priority queue architecture for high-speed network switches, in: Proc. IEEE INFOCOM, 2000.
[12] A. Ioannou, M. Katevenis, Pipelined heap (priority queue) management for advanced scheduling in high-speed networks, in: Proc. IEEE International Conference on Communications, 7 June 2001, pp. 2043–2047.
[13] R. Ramaswami, K.N. Sivarajan, Optical Networks: A Practical Perspective, second ed., Morgan Kaufmann, Los Altos, CA, 2002.
[14] M. Yoo, C. Qiao, S. Dixit, QoS performance of optical burst switching in IP-over-WDM networks, IEEE Journal on Selected Areas in Communications 18 (10) (2000) 2062–2071.
[15] K. Dolzer, C. Gauger, J. Späth, S. Bodamer, Evaluation of reservations mechanisms for optical burst switching, AEÜ International Journal of Electronics and Communications 55 (1) (2001).
[16] H.L. Vu, M. Zukerman, Blocking probability for priority classes in optical burst switching networks, IEEE Communications Letters 6 (5) (2002) 214–216.
[17] K. Claffy, G. Miller, K. Thompson, The nature of the beast: Recent traffic measurements from an Internet backbone, in: INET'98, 1998. Available from: <http://www.caida.org/outreach/papers/1998/Inet98/Inet98.pdf>.
[18] J. Cao, W.S. Cleveland, D. Lin, D.X. Sun, The effect of statistical multiplexing on the long-range dependence of Internet packet traffic, Bell Labs, Tech. Rep., 2002. Available from: <http://cm.bell-labs.com/cm/ms/departments/sia/doc/multiplex.pdf>.

Minh Hoang Phùng received his B.Eng. degree in Telecommunications (first-class honors) from the University of Sydney, Australia in 2000. He is currently working towards his Ph.D. degree in Electrical and Computer Engineering at National University of Singapore. His research interests focus on WDM optical networks.

Kee-Chaing (K.-C.) Chua (M'87) received a Ph.D. degree in Electrical Engineering from the University of Auckland, New Zealand, in 1990. Following this, he joined the Department of Electrical Engineering at the National University of Singapore (NUS) as a Lecturer, became a Senior Lecturer in 1995 and an Associate Professor in 1999. From 1995 to 2000, he was the Deputy Director of the Center for Wireless Communications (now Institute for Infocomm Research), a national telecommunication R&D institute funded by the Singapore Agency for Science, Technology and Research. From 2001 to 2003, he was on leave of absence from NUS to work at Siemens Singapore where he was the founding head of the ICM Mobile Core R&D department. He has carried out research in various areas of communication networks and has published more than 150 papers in these areas in international refereed journals and conferences. His current interests are in ensuring end-to-end quality of service in wireless and optical networks. He is a recipient of an IEEE 3rd Millennium medal.

Gurusamy Mohan received the Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology (IIT), Madras in 2000. He joined the National University of Singapore in June 2000, where he is currently an Assistant Professor in the Department of Electrical and Computer Engineering. He has held a visiting position at Iowa State University, USA, during January–June 1999. His current research interests are in high speed multi-wavelength optical circuit and burst switching networks, IP-MPLS, and Sensor networks. He is the coauthor of the textbook "WDM Optical Networks: Concepts, Design, and Algorithms" published by Prentice Hall PTR, NJ, USA in November 2001. He is a member of IEEE.

Mehul Motani received the Ph.D. in Electrical and Computer Engineering from Cornell University in 2000. Since August 2000, he has been an Assistant Professor in the Department of Electrical and Computer Engineering at the National University of Singapore. From 1992 to 1996, he was a member of technical staff at Lockheed Martin, Ocean Radar and Sensor Systems Group in Syracuse, New York. His current research interests are in information theory and coding, wireless networks, and optical networks. He was awarded the Intel Foundation Fellowship for work related to his Ph.D. in 2000. He is currently the secretary of the IEEE Information Theory Society Board of Governors.

Tung Chong Wong received the B.Eng. and M.Eng. degrees from National University of Singapore (NUS) in 1992 and 1994, respectively, and the Ph.D. degree from the University of Waterloo, Canada, in 1999, all in Electrical Engineering. He is with the Institute for Infocomm Research, Singapore (formerly Centre for Wireless Communications, NUS and Institute for Communications Research, NUS) first as a research engineer and currently as a scientist, since 1994. His research interests are in communications networks, 3G/4G and ultra-wideband wireless mobile multimedia networks. His areas of research are in the medium access control, resource allocation with quality of service constraints and traffic policing with heterogeneous traffic.

He is a Senior Member of IEEE. He was on the Technical Program Committee of the IEEE WCNC 2003, IEEE WCNC 2005 and will be serving on the Technical Program Committee of the IEEE GLOBECOM 2005.

Peng-Yong Kong received the B.Eng. degree in Electrical and Electronic Engineering (first-class honors) from the Universiti Sains Malaysia in 1995, and the Ph.D. degree in Electrical and Computer Engineering from the National University of Singapore in 2002. He was an Engineer with Intel Malaysia from 1995 to 1998. After that, he was a Research Scholar with the Center for Wireless Communications, Singapore (currently the Institute for Infocomm Research), working towards the Ph.D. degree. Since 2001, he has been with the Institute for Infocomm Research, where he is currently a Scientist. He is also an Adjunct Assistant Professor with the Electrical and Computer Engineering Department, National University of Singapore. His research interests are primarily in medium access control protocols, traffic scheduling, traffic control and quality of service provisioning for both wired and wireless networks.