2

NODE-BASED QOS IMPROVEMENT MECHANISMS

Quality of Service (QoS) mechanisms in a network can be broadly divided into two categories: QoS improvement and QoS provisioning mechanisms. A QoS improvement mechanism can be defined as any mechanism that improves the general performance of the network. Although less obvious than QoS provisioning mechanisms, QoS improvement mechanisms are very important in enabling the network to provide satisfactory service to end users. They allow the network to accommodate more users and reduce the cost of data transmission.

This chapter focuses on QoS improvement mechanisms located at a node in an OBS network. A survey of the current state-of-the-art, including optical buffering, deflection routing, burst segmentation, wavelength conversion and channel scheduling is presented. Then two channel scheduling algorithms that take advantage of the offset times, a unique feature of OBS, to give good performance are presented in detail.

2.1 Contention Resolution Approaches

Since a wavelength channel may be shared by many connections in OBS networks, there exists the possibility that bursts may contend with one another at intermediate nodes. Contention occurs when multiple bursts from different input ports are destined for the same output port simultaneously. The general solution to burst contention is to move all but one burst "out of the way". An OBS

node has three possible dimensions to move contending bursts, namely, time, space and wavelength. The corresponding contention resolution approaches are optical buffering, deflection routing and wavelength conversion, respectively. In addition, there is another approach unique to OBS called burst segmentation.

2.1.1 Optical buffering

Typically, contention resolution in traditional electronic packet switching networks is implemented by storing excess packets in Random Access Memory (RAM) buffers. However, RAM-like optical buffers are not yet available. Currently, optical buffers are constructed from Fibre Delay Lines (FDLs) [1, 2, 3]. An FDL is simply a length of fibre and hence offers a fixed delay. Once a packet/burst has entered it, it must emerge after a fixed length of time later. It is impossible to either remove the packet/burst from the FDL earlier or hold it in the FDL longer. The fundamental difficulty facing the designer of an optical packet/burst switch is to implement variable-length buffers from these fixed-length FDLs.

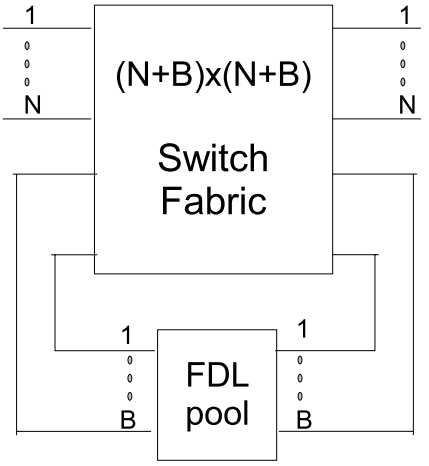
Current optical buffers may be categorised in different ways. They can be classified as either single-stage, i.e., having only one block of parallel delay lines, or multi-stage, which have several blocks of delay lines cascaded together. Single-stage optical buffers are easier to control, but multi-stage implementations may lead to more savings on the amount of hardware used. Optical buffers can also be classified as having feed-forward or feedback configurations. In a feed-forward configuration, delay lines connect the output of a switching stage to the input of the next switching stage. In a feedback configuration, delay lines connect the output of a switching stage back to the input of the same stage. Long holding time and certain degrees of variable delays can be easily implemented with a feedback configuration by varying the number of loops a packet/burst undergoes. However, each loop causes some loss in signal power. Therefore, a packet/burst cannot be stored indefinitely in a feedback architecture. In a feed-forward configuration, delay lines with different lengths must be used to achieve variable delays. This architecture attenuates all signals almost equally

because every packet/burst passes through the same number of switches. Hybrid combinations of feedforward and feedback architectures are also possible [4].

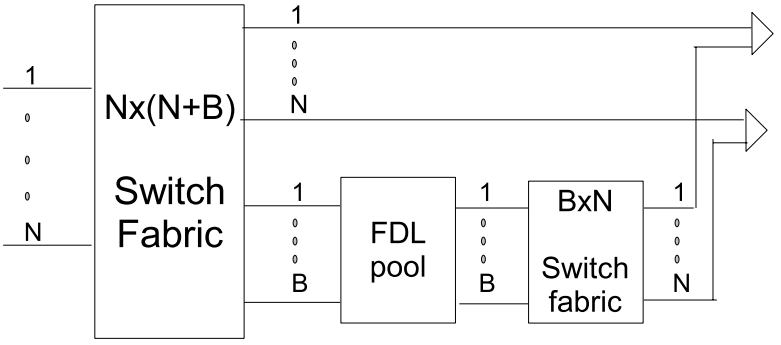
Based on the position of buffers, packet switches fall into one of three major categories: input buffering, output buffering and shared buffering. In input-buffered switches, a set of buffers is assigned for each input port. This configuration has poor performance due to the head-of-line blocking problem. Consequently, it is never proposed for purely optical implementation. In output-buffered switches, a set of buffers is assigned to each output port. Most optical switches emulate output buffering since the delay in each output optical buffer can be determined before the packet/burst enters it. Shared buffering is similar to output buffering except that all output ports share a common pool of buffers.

Due to their hardware-saving characteristics, multi-stage and/or shared-buffered architectures are predominant in optical switch proposals. Figure 2.1 shows two single-stage, shared-buffered switch architectures [5] with feedforward and feedback configurations where N and B are the number of input ports and the number of FDLs, respectively. They both contain an FDL pool that is shared among all output ports. In the feedforward configuration, packets/bursts may be delayed only once, whereas the feedback configuration allows them to be delayed multiple times. Since the FDLs are optical fiber themselves, it is possible for them to hold multiple packets/bursts of different wavelengths simultaneously [6]. However, this comes at the expense of increased complexity in scheduling algorithms. Compared to single-stage buffer architectures, multi-stage counterparts [7, 8, 9] are much more complex. They contain several primitive switching elements connected together by FDLs, usually in a feedforward configuration. Multi-stage buffers can achieve buffer depth of several thousands.

Recently, optical buffers based on slow-light delay lines have received considerable interest [10]. In slow-light delay lines, light is slowed down using a variety of techniques such as electromagnetically induced transparency (EIT), population oscillations (POs) and microresonator-based photonic-crystal (PC) filter. In principle, these techniques can make the group velocity approach zero.



(a) Feedback shared-buffered architecture



(b) Feedforward shared-buffered architecture

Fig. 2.1. Single-stage optical buffer architectures. ©[2006] IEEE.

However, very slow group velocity always comes at the cost of very low bandwidth or throughput. Therefore, slow-light delay lines are still not practical in optical switches that have to handle very high data rates.

In summary, despite the considerable research efforts on FDL-based optical buffers, there remain some hurdles that limit their effectiveness. Firstly, by their nature, they can only offer discrete

delays. The use of recirculating delay lines can give finer delay granularity but it also degrades optical signal quality. Secondly, the size of FDL buffers is severely limited not only by signal quality concerns but also by physical space limitations. A delay of 1 ms requires over 200 km of fibre. Due to the size limitations of buffers, optical buffering alone as a means of contention resolution may not be effective under high load or bursty traffic conditions.

2.1.2 Deflection routing

Deflection routing is a contention resolution approach ideally suited for photonic networks that have little buffering capacity at each node. If no buffer is present, deflection routing is also known as hot-potato routing. In this approach, if the intended output port is busy, a burst/packet is routed (or deflected) to another output port instead of being dropped. The next node that receives the deflected burst/packet will try to route it towards the destination. The performance of slotted deflection routing has been extensively evaluated for regular topologies such as ShuffleNet, hypercube and Manhattan Street Network [11, 12, 13]. It is found that deflection routing generally performs poorly compared to store-and-forward routing unless the topology in use is very well connected. Nevertheless, its performance can be significantly improved with a small amount of buffers. Between slotted and unslotted networks, deflection routing usually performs better in the former since the networks can make use of the synchronous arrival of the packets to a router to minimise locally the number of deflections. Nevertheless, such deflection minimisation can also be done to some extent in unslotted networks using heuristics [14]. This brings the performance of deflection routing in unslotted networks close to that in slotted networks.

For an arbitrary topology, the choice of which output links to use for deflected bursts/packets is critical to the performance of the network. The existing deflection routing protocols can be divided into three categories: fixed alternate routing, dynamic traffic aware and random routing. Fixed alternate routing is the most popular approach. In this method, the alternate path is either defined on a

hop-by-hop basis [15] or by storing at each node both the complete primary path and the complete alternate path from itself to every possible destination node in the network [16]. Fixed alternate routing can yield good performance on small topologies. However, selecting a good alternate path becomes difficult on large topologies due to the tight coupling between subsequent burst loss probabilities, traffic matrices and network topology. Traffic aware deflection routing takes into consideration the transient traffic condition in selecting the output links for deflected bursts/packets [17, 18]. It becomes similar to load balancing, which we will address in Chapter 5. Random deflection routing [19] appears to strike the right balance between simplicity of implementation, robustness and performance. In this approach, bursts/packets carry in their header a priority field. Every time a burst/packet is deflected, its priority is decreased by one. Normal bursts/packets on their primary paths can preempt those low priority ones. Thus, the worst-case burst/packet loss probability of this method is upper-bounded by that in standard networks.

To apply deflection routing to OBS networks, the problem of insufficient offset time must be overcome. This problem is caused by a burst traversing more hops than originally intended as a result of being deflected. Since the offset time between the burst and its header decreases after each hop, the burst may overtake the header packet. Various solutions have been proposed [16], such as setting extra offset time or delaying bursts at some nodes on the path. It is found that delaying a burst at the next hop after it is deflected is the most promising option.

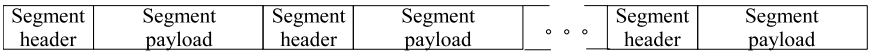
Deflection routing may be regarded as "emergency" or *unplanned* multipath routing. It might cause deflected bursts to follow a longer path than other bursts in the same flow. This leads to various problems such as increased delay, degradation of signal quality, increased network resource consumption and out-of-order burst arrivals. A better method to reduce congestion and burst loss is probably *planned* multipath routing, or load-balancing. The topic of load balancing will be discussed in chapter 5.

2.1.3 Burst segmentation

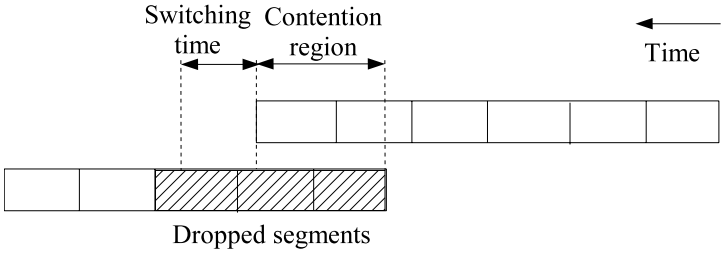
Burst segmentation [20, 21] is a contention resolution approach unique to OBS networks. It takes advantage of the fact that a burst is composed of multiple IP packets, or segments. Therefore, in a contention between two overlapping bursts, only the overlapping segments of a burst need to be dropped instead of the entire burst. Network throughput is improved as a result. Two currently proposed variants of burst segmentation are shown in Figure 2.2. In the head-dropping variant [20], the overlapping segments of the later arriving burst, or the head segments, are dropped. On the other hand, the tail-dropping variant [21] drops the overlapping segments of the preceding burst, or the tail segments. A number of strategies to combine burst segmentation with deflection routing have also been discussed. Comparing the two variants, the tail-dropping approach results in a better chance of in-sequence delivery of packets at the destination. Burst segmentation is later integrated with void-filling scheduling algorithms in [22, 23]. A performance analysis of burst segmentation is presented in [24].

2.1.4 Wavelength conversion

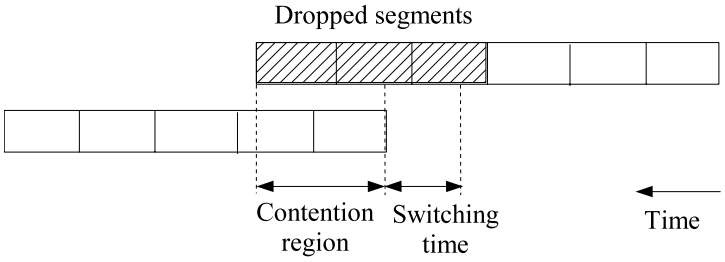
Wavelength conversion is the process of converting the wavelength of an incoming signal to another wavelength for transmission on an outgoing channel. In WDM, each fibre has several wavelengths, each of which functions as a separate transmission channel. When contention for the same output wavelength happens between some bursts, the node equipped with wavelength converters can convert all except one burst to other free wavelengths. Wavelength conversion enables an output wavelength to be used by bursts from several input wavelengths, thereby increasing the degree of statistical multiplexing and the burst loss performance. As the number of wavelengths that can be coupled into a fibre continues to grow, this approach becomes increasingly attractive. For example, with 32 wavelengths per link, the burst loss probability at a loading of 0.8 is about 4×10^{-2}. With 256 wavelengths per link, the burst loss probability drops to less than 10^{-4}.



(a) Segment structure of a burst



(b) Head dropping



(c) Tail dropping

Fig. 2.2. Burst segmentation approaches

Although optical wavelength conversion has been demonstrated in the laboratory environment, the technology remains expensive and immature. Therefore, to be cost-effective, an optical network may be designed with some limitations on its wavelength conversion capability. Following are the different categories of wavelength conversion:

- *Full conversion:* Any incoming wavelength can be converted to any outgoing wavelength at every core node in the network. This is assumed by most current OPS and OBS proposals. It

is the best performing and also the most expensive type of wavelength conversion.

- *Sharing of converters at a node:* Converter sharing [25, 26, 27] is proposed for OPS/OBS networks. It allows savings on the number of converters needed. However, the drawbacks are the enlargement of the switching matrix and additional attenuation of the optical signal.
- *Sparse location of converters in the network:* Only some nodes in the network are equipped with wavelength converters. Although this category is well-studied for wavelength-routed networks, it has not been widely considered for OPS and OBS networks due to the poor loss performance at nodes without wavelength conversion capability.
- *Limited-range conversion:* An incoming wavelength can only be converted to some of the outgoing wavelengths. Various types of limited-range converters for OPS networks have been examined [28, 29, 30]. It is shown that nodes with limited-range wavelength converters can achieve loss performance close to those with full conversion capability.

2.2 Traditional Channel Scheduling Algorithms

Since the large number of wavelengths per link in WDM offers excellent statistical multiplexing performance, wavelength conversion is the primary contention resolution approach in OBS. In this approach, every OBS core node is assumed to have full wavelength conversion capability. When a header packet arrives at a node, the node invokes a channel scheduling algorithm to determine an appropriate outgoing channel to assign to the burst. Channel scheduling plays a crucial role in improving the burst loss performance of an OBS switch. A good scheduling algorithm can achieve several orders of magnitude performance improvement over a first-fit algorithm. Because of its importance, channel scheduling in OBS has been the subject of intense research in the last few years.

In the JET OBS architecture, each burst occupies a fixed time interval, which is characterised by the start time and the end time carried in the header packet. Therefore, channel scheduling can be

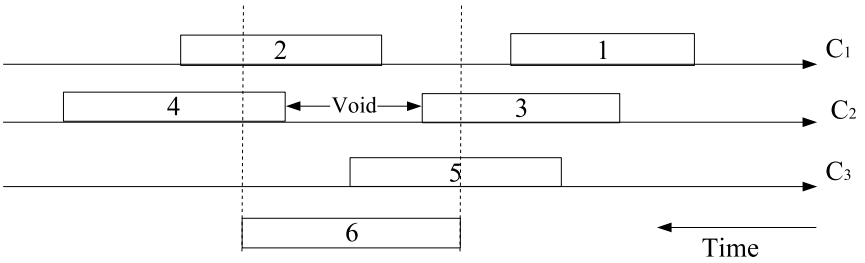


Fig. 2.3. Illustration of the channel fragmentation problem

regarded as a packing problem wherein the primary objective is to pack as many incoming bursts onto the outgoing channels as possible. This problem is complicated by the fact that the order of the header packet arrivals is not the same as the arrival order of the bursts themselves. Thus, bursts with long offset times are able to reserve a channel before those with shorter offset times. Their reservations fragment a channel's free time and produce gaps or *voids* among them that degrade the schedulability of bursts with shorter offset times. This is illustrated in Figure 2.3 where the numbers inside the bursts indicate their header arrival order. Although all six bursts can theoretically be accommodated, burst 6 cannot be scheduled because of the channel fragmentation caused by the other bursts. Many channel scheduling algorithms have been proposed to deal with this problem. In this section, a survey of some traditional channel scheduling algorithms is given.

2.2.1 Non-void filling algorithm

Non-void filling algorithm is the simplest type of channel scheduling algorithms. It is named Horizon [31] and Latest Available Unscheduled Channel (LAUC) [32] by two independent research groups. In order to maximise processing speed, it does not utilise voids caused by previously scheduled bursts to schedule new bursts. Instead, it only keeps track of the *unscheduled* time, which is the end time of the last scheduled burst, for each channel. When a header arrives, it assigns to the burst the channel with the unscheduled time being closest but not exceeding its start time. The idea is to minimise the void produced before it. This is illus-

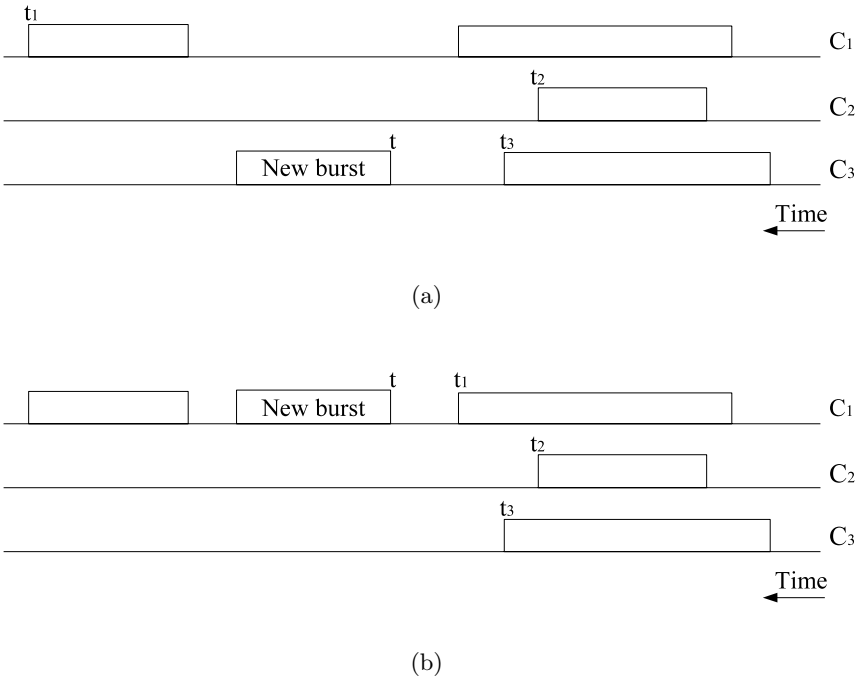


Fig. 2.4. Examples of channel assignment using: (a) non-void filling algorithm (Horizon or LAUC), and (b) void filling algorithm (LAUC-VF)

trated in Figure 2.4(a) where t_1, t_2 and t_3 are the unscheduled times. Channel C_3 is selected to schedule the new burst because $t-t_3 < t-t_2$. By storing the unscheduled times in a binary search tree, the two algorithms can be executed in $O(logW)$ time, where W is the number of wavelengths per link.

2.2.2 Algorithms with void filling

Void-filling algorithms [32, 33] utilise voids to schedule new bursts to improve burst loss performance. They keep track of every void on the outgoing channels and check all of them as well as unscheduled channels when an incoming burst needs to be scheduled. Latest Available Unused Channel with Void Filling (LAUC-VF) [32] is perhaps the most popular OBS channel scheduling algorithm to date. When an incoming burst needs to be scheduled, LAUC-VF

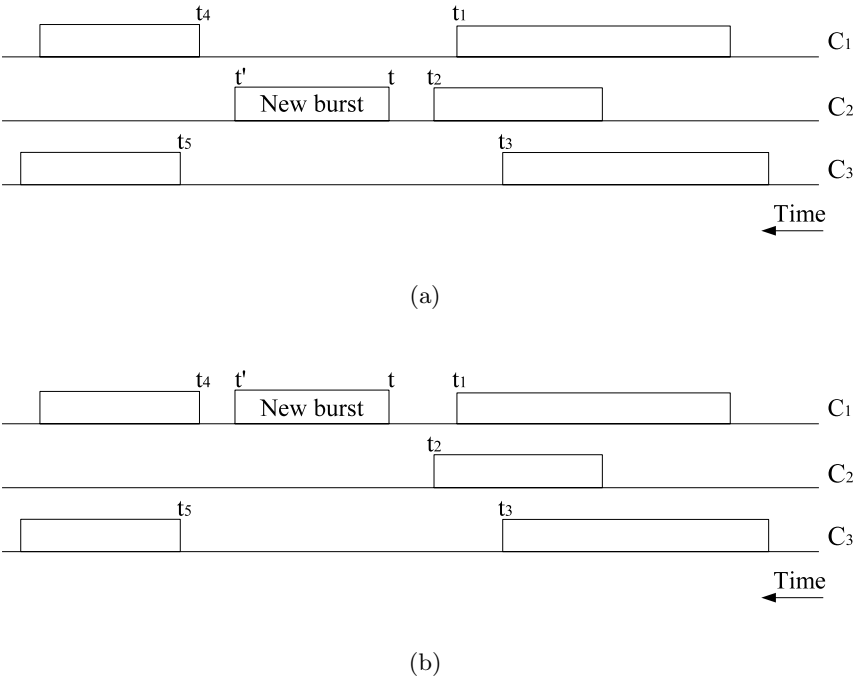


Fig. 2.5. Comparison of LAUC-VF and its variants: (a) LAUC-VF, and (b) LAUC-VF variants

calculates the *unused* time of each available channel, which is the end time of the burst preceding the incoming one. The channel with the unused time closest to the start of the incoming burst is selected. This is illustrated in Figure 2.4(b) where t_1, t_2 and t_3 are the unused times. Channel C_1 is selected to schedule the new burst because t_1 is closest to t. Since the unused times for each burst are different, LAUC-VF has to recalculate all of them for each new burst. Using a binary search tree structure, each unused time calculation takes $O(logN_b)$, where N_b is the average number of scheduled bursts per channel. Thus, LAUC-VF takes $O(WlogN_b)$ to execute.

A drawback of the basic LAUC-VF algorithm is that it may select an unscheduled channel to schedule a new burst even though suitable voids are available because it bases its decision only on the unused times. This problem is illustrated in Figure 2.5(a) where

LAUC-VF selects channel 2 for the new burst because $t - t_2$ is smaller than both $t - t_1$ and $t - t_3$. This creates more voids and degrades performance. An LAUC-VF variant [34] that solves the problem is to give priority to channels with voids, thus minimising the number of voids generated. A further refinement of LAUC-VF [35] is to compare all the voids that would be generated and choose the channel that will give minimum voids. In the illustration in Figure 2.5(b), the first variant will schedule the new burst on either channel 1 or channel 3 while the second variant will only choose channel 1 since $t-t_1 < t-t_3$ and $t_4-t' < t_5-t'$. Parallel processing and associative memory can be used to implement LAUC-VF and its variants to reduce the time complexity [36].

Another implementation of LAUC-VF [37] is called Minimum Starting Void (Min-SV). Min-SV has the same scheduling criteria as LAUC-VF. However, it uses an augmented balanced binary search tree as the data structure to store the scheduled bursts. This enables Min-SV to achieve processing time as low as that of Horizon without requiring special hardware or parallel processing.

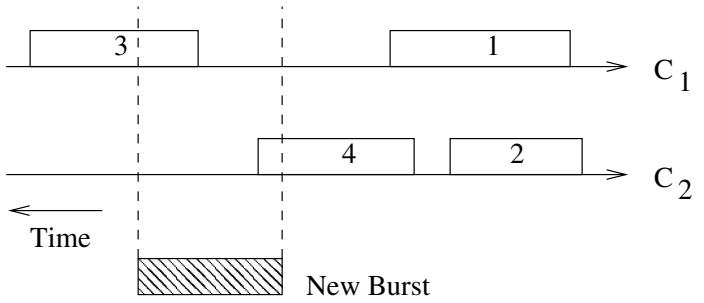
2.3 Burst-Ordered Channel Scheduling Approach

A common characteristic of the traditional channel scheduling algorithms is that they schedule incoming bursts in the order of their header packet arrivals. Therefore, the bursts with long offset times are able to reserve a channel before those with shorter offset times. Their reservations fragment a channel's free time and produce voids among them that degrade the schedulability of bursts with shorter offset times. To address the root of this problem, it is required that a node schedules incoming bursts in the order of their actual arrivals. In that case, the voids will have no negative effect on burst scheduling because by the time a void is generated, all the bursts in that region have already been scheduled.

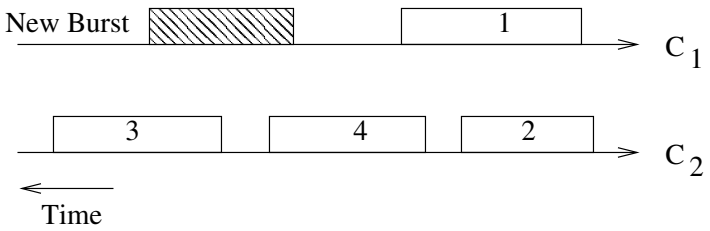
The most popular method to implement the burst-ordered scheduling approach is to delay the processing of headers that arrive early (bursts with long offset times) in order to collect information from the headers that arrive later (bursts with short

offset times). In the batch scheduling variant [38, 39], a node collects multiple burst headers and makes scheduling decisions for all of them at one go. In the delayed scheduling variant [40, 41], the node delays burst headers for a certain period and sorts them according to their burst arrivals. However, in both variants, the inherent conflict between the need to delay headers and the need to forward headers early to give downstream nodes sufficient processing times prevents the burst-ordered scheduling concept from being fully realised.

Dual-header OBS (DOBS) [42] is another method to implement the burst-ordered scheduling concept. In this method, information about each burst is carried in two separate headers: the *service request packet* (SRP) header and the *resources allocated packet* (RAP) header. The SRP header originates from the ingress node and carries service requirement information such as offset time and burst length, which allows intermediate nodes to calculate the time interval that the incoming burst will occupy. Upon receiving an SRP header, a node records the information contained within the header and passes it onto the next downstream node. Later, just before the corresponding data burst arrives, the node makes a scheduling decision for the burst and sends the scheduled channel information to the next node in an RAP header. This method resolves the conflict in the batch scheduling and delayed scheduling methods and fully realises the burst-ordered scheduling concept. However, it has the problem of phantom reservations since a SRP header is sent to the next node before the scheduling decision is made. Therefore, it is possible that the burst is dropped at that node while the SRP header continues to make reservations at downstream nodes. Those phantom reservations caused by bursts dropped at an upstream node lock up resources at downstream nodes, which could be used for other bursts.



(a) LAUC-VF fails to schedule the new burst



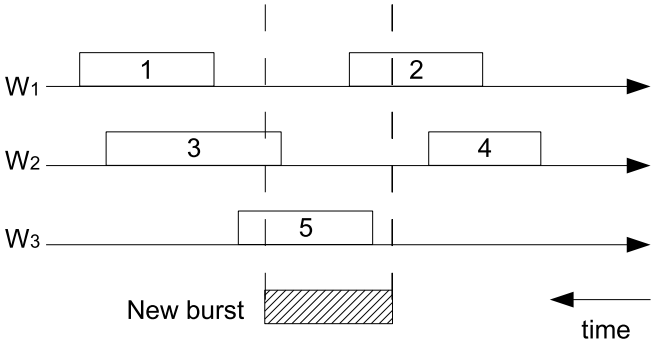
(b) Burst 3 is rescheduled to accommodate the new burst

Fig. 2.6. Illustration of the benefits of burst rescheduling

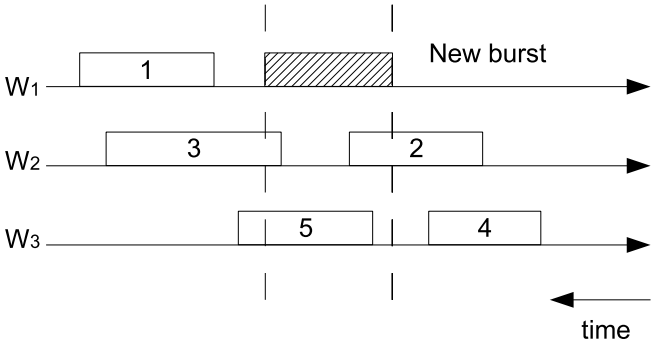
2.4 Burst Rescheduling

Burst rescheduling[1] [43, 44] attempts to approximate the burst-ordered scheduling concept. It helps to improve the burst loss performance and at the same time achieve low computational complexity. The key idea of burst rescheduling is to reschedule an existing scheduled burst to another wavelength to accommodate an incoming burst. This is possible as requests arrive dynamically and a header packet reserves wavelengths well before the arrival of its corresponding data burst. The benefit of burst rescheduling is illustrated in Figure 2.6. In this scenario, the new burst, which cannot be scheduled by LAUC-VF, can be scheduled after

[1] Reprinted from (S. K. Tan, G. Mohan, and K. C. Chua, "Algorithms for Burst Rescheduling in WDM Optical Burst Switching Networks," *Computer Networks*, vol. 41, no. 1, pp. 41–55), ©[2003], with permission from Elsevier.



(a) No wavelength is available for new burst.



(b) Multi-level rescheduling to accommodate new burst.

Fig. 2.7. Illustration of multi-level rescheduling.

burst 3 is moved to another free wavelength. It is to be noted that rescheduling does not affect any ongoing traffic. Rescheduling a burst on a link requires changes in the control setting in both the end nodes of the link. Therefore, whenever rescheduling is successful, a special "NOTIFY" packet is sent to the next node to notify it about the changes, for e.g., wavelength for that burst so that the receiving node will do the necessary settings.

Rescheduling algorithms can be developed based on two approaches. These are *single-level rescheduling* and *multi-level rescheduling*. Single-level rescheduling involves the rescheduling of only one

burst to another available wavelength to accommodate a new burst. The example illustrated in Figure 2.6(b) falls under this category. In multi-level burst rescheduling, several bursts are rescheduled one by one in sequence to other available wavelengths in order to accommodate a new burst. As shown in Figure 2.7(a), no wavelength is available for the new burst if single-level rescheduling is used. Multi-level rescheduling can reschedule burst 4 from W_2 to W_3 followed by rescheduling burst 2 from W_1 to W_2 to free wavelength W_1 to accommodate the new burst, as shown in Figure 2.7(b). Multi-level rescheduling is expected to provide better performance than single-level rescheduling. However, from the computational complexity point of view, multi-level rescheduling is more complex than single-level rescheduling. This is because a multi-level rescheduling algorithm needs to determine an appropriate order (among several possibilities) in which different bursts are to be rescheduled in sequence. Since the objective is to achieve low computational complexity, single-level rescheduling is generally preferred.

2.4.1 Burst rescheduling algorithms

On-Demand Burst Rescheduling (ODBR) algorithm

As the name suggests On-Demand Burst Rescheduling (ODBR) algorithm considers rescheduling of an existing burst only when a burst fails to be scheduled to any of the wavelengths. The algorithm works in two phases. When a new burst arrives, phase 1 is executed to select a suitable free wavelength using LAUC. If no wavelength is available, phase 2 is called to check if any of the existing bursts can be moved to a new wavelength to enable scheduling of the new burst. The algorithm examines the wavelengths one by one. For a given wavelength, it checks if the last burst can be moved to any other wavelength and determines the void created. After examining all the wavelengths, it chooses the one which possibly creates the smallest void after migration.

The pseudo-code of the algorithm is given in Table 2.1. The new burst is assumed to arrive at time t. The latest available time of wavelength channel W_i is denoted by t_i.

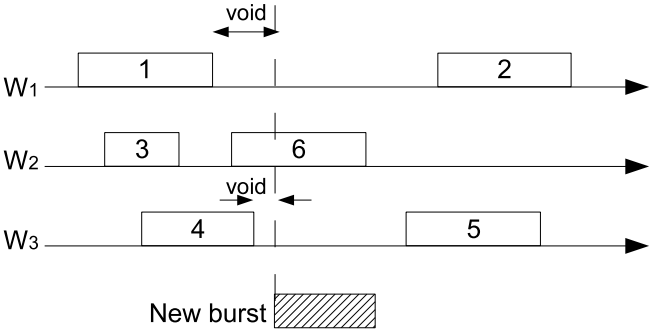
Table 2.1. ODBR algorithm

Phase 1 Use LAUC to find a suitable wavelength for the incoming burst. If no such wavelength is found, call phase 2. Otherwise, exit.
Phase 2 Step 1: For every wavelength W_i and out-wavelength V_i, determine if V_i is valid. Out-wavelength V_i is said to be valid if the last burst on W_i can be moved to V_i and the new burst can be scheduled to W_i. Step 2: If no valid out-wavelength V_i exists, the new burst is dropped. Otherwise, choose wavelength W_p that has a valid out-wavelength V_p and is the latest available wavelength after rescheduling among all the valid out-wavelengths. Step 3: Reschedule last burst on W_p to V_p. Assign new burst to W_p. Step 4: Send a special NOTIFY header packet to notify the next node about the change in wavelength of the rescheduled burst.

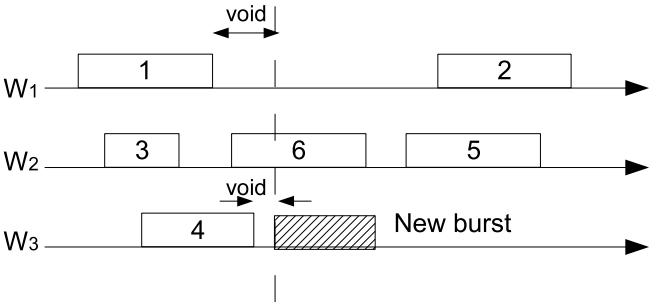
A simple example shown in Figure 2.8(a) and (b) helps to illustrate how ODBR works. Phase 1 fails to assign any wavelength to the new burst as shown in Figure 2.8(a) and phase 2 will therefore be invoked. Wavelength W_1 and W_3 both have a valid out-wavelength W_2. Therefore, rescheduling of the last burst from W_1 or W_3 to W_2 would make a wavelength available for scheduling the new burst. In order to optimize the performance, ODBR chooses the best wavelength which has the latest available time. In this case, the void formed by the new burst on W_3 by rescheduling the burst from W_3 to W_2 is the smallest compared to the void formed at W_1 by rescheduling the burst from W_1 to W_2. Therefore, W_3 is the best wavelength. The last burst on W_3 is rescheduled to W_2 and the new burst can be scheduled to W_3 as shown in Figure 2.8(b).

The complexity of ODBR is as follows

- *Phase 1 - Scheduling:* ODBR examines the information of one burst on each wavelength. Phase 1 runs in $O(W)$ time in the worst case.
- *Phase 2 - Rescheduling:* Phase 2 has the worst case complexity of $O(W^2)$ time since it examines the last burst on each wavelength for rescheduling to one of the other wavelengths.



(a) The new burst cannot be scheduled.



(b) The last burst on W_3 is moved to W_2 to accommodate the new burst on W_3.

Fig. 2.8. Illustration of multi-level rescheduling.

Since the complexity of LAUC-VF is $O(KW)$ with K being the average number of scheduled bursts per wavelength, the complexity of ODBR will be more than LAUC-VF if $W > K$. However, since ODBR is called only when a burst is dropped (usually less than 10%), the overall processing complexity remains better than LAUC-VF. It therefore has the advantage of low complexity similar to LAUC.

Aggressive Burst Rescheduling (ABR) algorithm

As shown in Table 2.2, the ABR algorithm also has two phases. However, it is different from the ODBR algorithm in that phase 2

Table 2.2. ABR algorithm

Phase 1 Use LAUC to find a suitable wavelength W_p for the incoming burst. If no such wavelength is found, drop the burst. Otherwise, assign wavelength W_p to the new burst and call phase 2. **Phase 2** Step 1: For every wavelength W_i other than W_p determine if the last burst can be rescheduled to W_p and also the void created at W_p after rescheduling. If such rescheduling is possible for a wavelength, it is said to be a valid in-wavelength for W_p. If no valid in-wavelength exists, exit phase 2. Step 2: Choose a valid in-wavelength W_j which has the smallest void. Step 3: Reschedule the last burst from W_j to W_p. Step 4: Send a special NOTIFY header packet to notify the next node about the change of wavelength of the rescheduled burst.

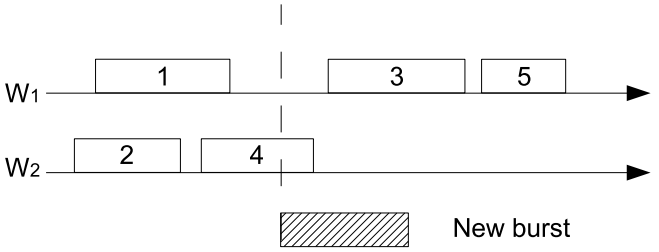
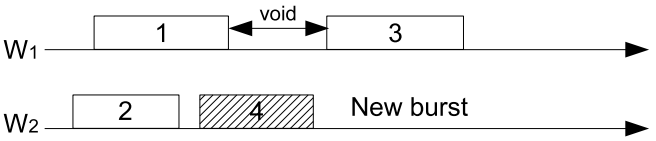


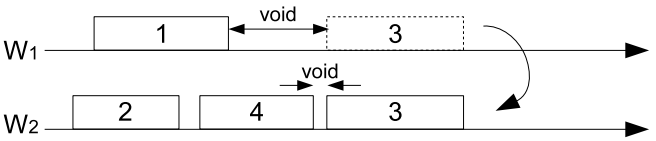
Fig. 2.9. LAUC, ODBR and LAUC-VF fail to schedule new burst 6.

is not invoked when phase 1 fails but when phase 1 is successful. This algorithm is intended to prevent future data burst dropping by invoking rescheduling every time a burst has been scheduled successfully. In ABR, upon successful scheduling of a burst at W_p in phase 1, rescheduling of one latest burst from some other wavelength W_i to W_p takes place in phase 2 if such a burst exists. Rescheduling is governed by the rule that the void formed when the burst is rescheduled from W_i to W_p is minimum among all possible wavelengths. By doing so, the probability of dropping data bursts that arrive later could be decreased.

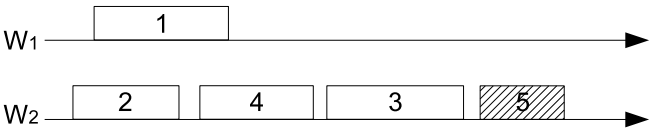
Examples as shown in Figure 2.9 and Figure 2.10 are used to illustrate this algorithm. Figure 2.9 shows two wavelengths and bursts that are being considered. Burst 1, 2, 3, 4, and 5 arrive at a node one by one in that order and are scheduled to W_1 and W_2 at phase 1. When burst 6 arrives, it cannot be scheduled to



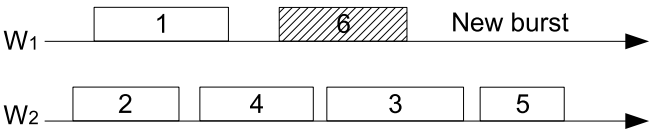
(a) New burst 4 is assigned to W_2.



(b) Last burst from W_1 is rescheduled to W_2.



(c) Burst 5 is assigned to W_2.



(d) Burst 6 will be able to be scheduled to W_1.

Fig. 2.10. Illustration of working of ABR algorithm.

any wavelength by LAUC, LAUC-VF, or ODBR. For the same burst arriving pattern, Figure 2.10(a) to (d) show that with ABR, the new burst 6 which would otherwise have been dropped, can be scheduled successfully. This demonstrates that prevention of burst dropping is achieved by using ABR. As shown in Figure 2.10(a), when burst 4 is scheduled to wavelength W_2, consideration for rescheduling of one last burst from other wavelength to

current wavelength W_2 takes place. Here, the last burst on wavelength W_1 is scheduled to wavelength W_2 as it conforms to the rule that the void formed after rescheduling is shorter, as shown in Figure 2.10(b). Figure 2.10(c) shows that when burst 5 arrives, it is scheduled to wavelength W_2 in phase 1 as it is the latest available wavelength. Finally, burst 6 will be scheduled to W_1 at time t of its arrival as shown in Figure 2.10(d). If there are more than one burst on different wavelengths that could be rescheduled to W_p, the burst with the smallest void formed after being rescheduled to W_p would be chosen. This is to make sure that the smallest void would be formed every time a rescheduling takes place.

The complexity of ABR is as follows.

- Phase 1 - Scheduling : ABR examines the information of one burst on each wavelength. Phase 1 runs in $O(W)$ time in the worst case.
- Phase 2 - Rescheduling : Phase 2 has the worst case complexity of $O(W)$ time as well since it examines only the last burst on each wavelength for rescheduling.

The complexity of ABR is approximately two times that of LAUC since it examines only the last burst on each wavelength for rescheduling. Therefore, even for values of $K \geq 3$, ABR is expected to run faster than LAUC-VF whose complexity is $O(KW)$. It therefore has the advantage of low complexity similar to LAUC.

2.4.2 Signalling overhead

Additional signalling is needed when rescheduling is successful by using ODBR or ABR. This is to notify the next node about the change of wavelength by sending a NOTIFY packet. However, rescheduling does not incur significant signalling overhead for both ODBR and ABR algorithms. The NOTIFY packet is much smaller than the header packet as it needs to carry only the wavelength change information. Also, no complex algorithm is executed upon receiving the NOTIFY packet. Further, a successful reschedule requires a NOTIFY packet to be sent on only one link. Alternatively, without sending extra signalling packet, the information

can be piggybacked to the header packet. It therefore does not incur significant processing time and does not consume significant control channel bandwidth when compared to the computational complexity gain achieved over existing algorithms such as LAUC-VF.

2.4.3 Performance study

The performance of the burst rescheduling techniques is studied via simulation in [43]. A random topology with 32 nodes and 60 bidirectional links is considered. Each link has 8 wavelengths for carrying data traffic. The transmission capacity of each wavelength is approximately 10 Gbps. No FDL buffer is assumed in the network. Network traffic consists of two classes, namely class 1 and class 2. Class 2 traffic is given a higher priority over class 1 traffic by assigning an extra offset time. The performance of ODBR and ABR is evaluated under different traffic loading conditions. The burst arrival rate is measured as the number of bursts arrived per node per microsecond. The range for traffic load is chosen to be from 0.3 to 0.6 so that the burst dropping probability is below 15%.

Figures 2.11, 2.12, and 2.13, indicate that ODBR and ABR have better performance in terms of the overall burst dropping probability and that for class 1, and class 2 traffic, respectively, than LAUC. The dropping probability increases with increasing traffic load as most of the wavelengths are heavily used at high traffic load, therefore, it is less probable for a burst to find an available wavelength. However, as shown in Figures 2.11 and 2.12, the performances of ODBR and ABR are always in between LAUC and LAUC-VF. Particularly, ODBR and ABR perform closer to LAUC-VF at low arrival rates than at high arrival rates. This is because more voids are created at high arrival rates and the rescheduling algorithms consider only the last burst for rescheduling and do not utilize the voids in between the burst as in LAUC-VF.

Figure 2.13 shows that all algorithms have similar dropping performance for class 2 high priority traffic. This is because class

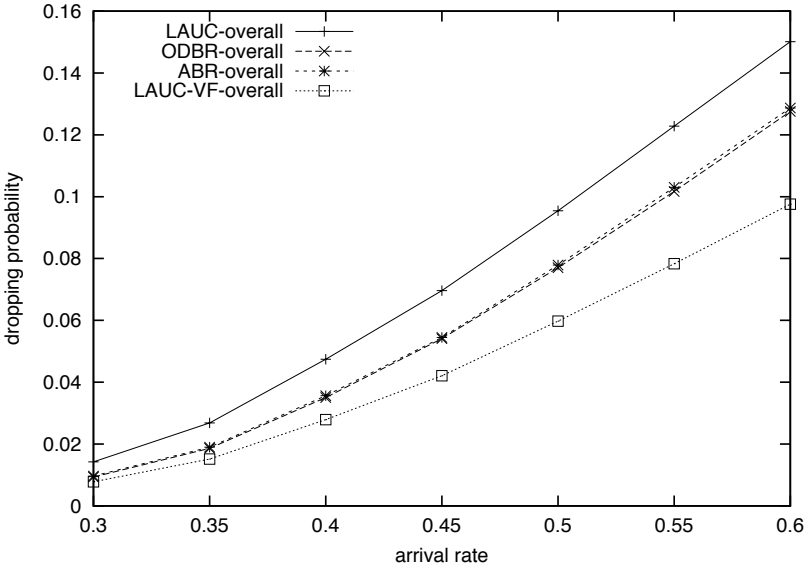


Fig. 2.11. Performance of overall traffic under different traffic loading.

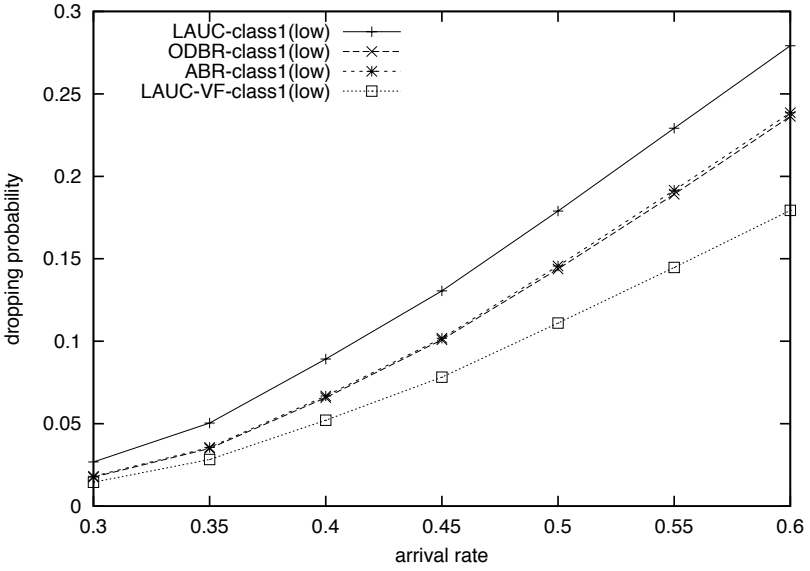


Fig. 2.12. Performance of class 1 traffic under different traffic loading.

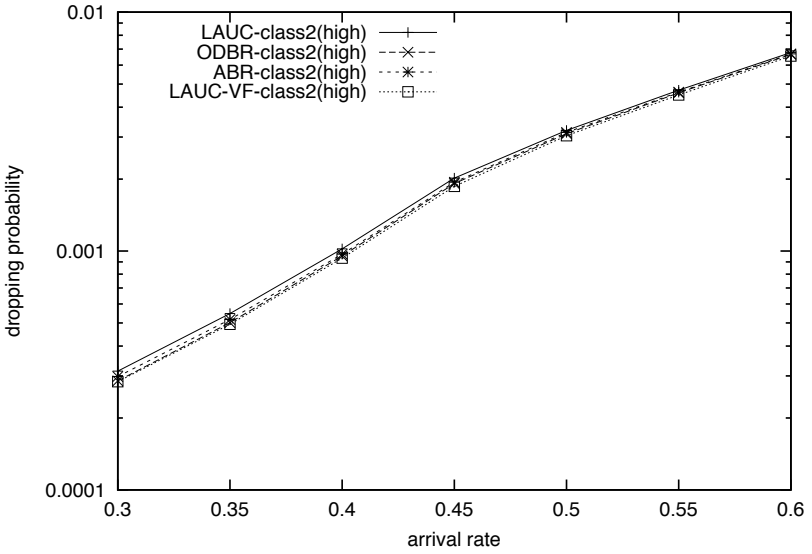


Fig. 2.13. Performance of class 2 traffic under different traffic loading.

2 traffic have large initial offset time as compared to class 1 traffic, which makes a class 2 burst highly likely to reserve wavelength at the far end on the time line. Therefore, a high priority burst is highly likely to be the last burst and hence not much improvement is achieved by LAUC-VF and the rescheduling algorithms over LAUC.

The simulation study in [43] has also observed the number of header packets that correspond to successful bursts and the number of NOTIFY packets that correspond to successful rescheduling on each of the links. The results show that about 2% and 20% of signalling overhead is incurred by ODBR and ABR, respectively.

2.5 Ordered Scheduling

2.5.1 High-level description

Unlike other channel scheduling algorithms that are heuristic in nature, Ordered Scheduling[2] [45] is designed to optimise burst scheduling in OBS. 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 packet is passed on to the next node. The buffer is in the form of a priority queue[3] 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 Figure 2.14 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 packets 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. Since the priority queue sorts the reservations according their burst arrival times, all unscheduled reservations will be to the left of the newly dequeued reservation. Therefore, any void it produces on the right has no effect on the schedulability of non-scheduled

[2] Reprinted from (M. H. Phung, K. C. Chua, G. Mohan, M. Motani, T. C. Wong, and P. Y. Kong, "On Ordered Scheduling for Optical Burst Switching," *Computer Networks*, vol. 48, no. 6, pp. 891–909), ©(2005), with permission from Elsevier.

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

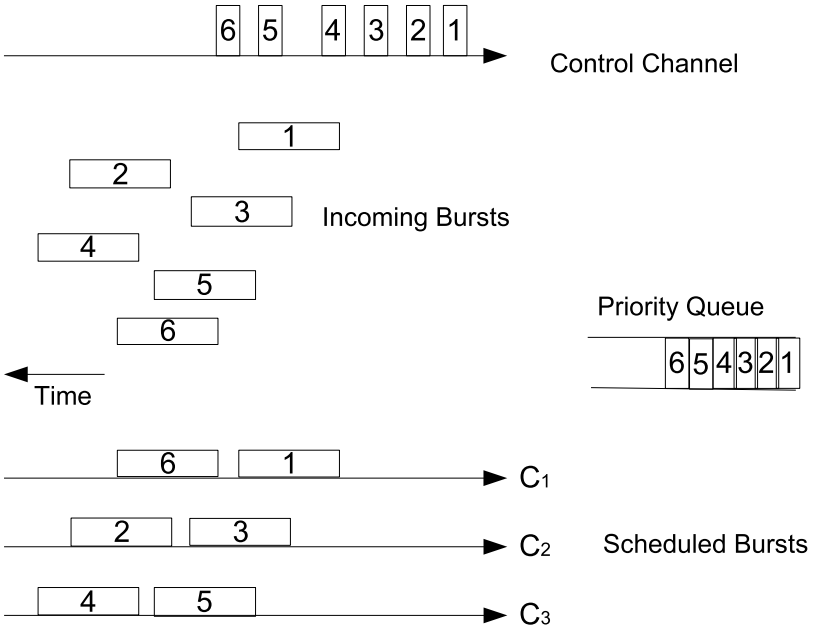


Fig. 2.14. The main concept of Ordered Scheduling

reservations. Thus, any free wavelength can be assigned to the newly dequeued reservation. In the example, a round robin assignment is used 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 $[t_0 + D_{FDL}, t_1 + D_{FDL}]$. 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 using the priority queue.

It can be seen that Ordered Scheduling is similar to Dual-header OBS described in section 2.3. The primary difference between the two schemes is the admisison control test. The admission control test is important because it prevents resource wastage due to over-admitting bursts, also known as phantom reservations in Dual-header OBS. In both schemes, 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 packet 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.

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.

2.5.2 Admission control test realisation

The admission control test in section 2.5.1 is presented in continuous form, which may not be practical or feasible to realise. 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 $N_{occupied} < M$ for all the slots it will occupy. This version is referred to in [45] as Basic Ordered Scheduling. It is suitable if the optical switches in the network also operate in a slotted fashion as mentioned in [32]. 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.

An enhanced version of the above slotted approach is called Enhanced Ordered Scheduling in [45]. Instead of a single number to indicate the number of bursts occupying a time slot, the admission control routine keeps the following three data entities for each time slot:

1. N_{total} is the total number of bursts that occupy the slot, whether wholly or partly;

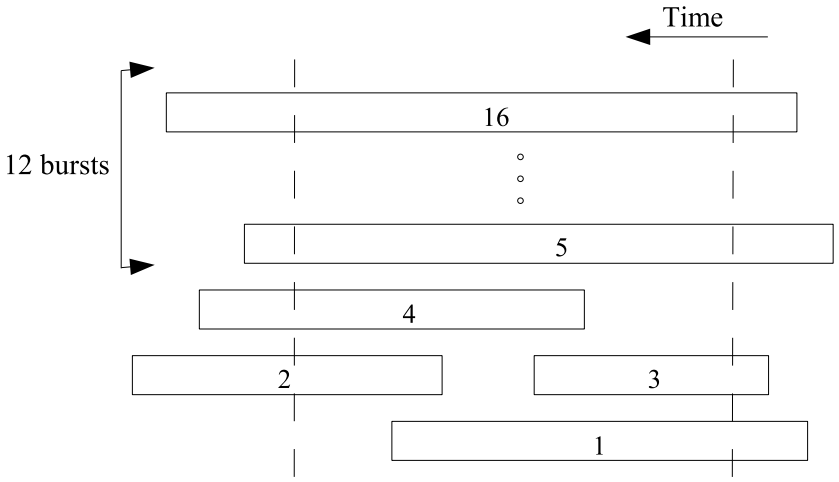


Fig. 2.15. Example of the bookkeeping for a typical time slot.

2. *heads* is the list of the start times of the bursts that have the start of their reservation periods fall within the slot, sorted in increasing order; and
3. *ends* is the list of the end times of the bursts that have the end of their reservation periods fall within the slot, sorted in increasing order.

The bookkeeping of the two versions is illustrated in Figure 2.15. In the figure, the slot sizes are exaggerated and the bursts not shown occupy the entire slot. For the basic version, $N_{occupied} = 16$. For the enhanced version, $N_{total} = 16$ and there are two entries in each of *heads* and *ends*.

When a header packet arrives at a node to request a 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 packet. 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 packet is 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 $heads[i] \leq ends[j]$ 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 Table 2.3. 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 $insert(t_0, t_1)$.

In terms of loss performance, Enhanced Ordered Scheduling is optimal since it fully implements the test in continuous form. Its outperformance compared to the basic version is illustrated in Figure 2.15 where the basic version reports that 16 wavelengths are occupied for the 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.

Table 2.3. Admission control test

```
MAIN()
accept ← true
for slot = s_0 to s_1
    slot.insert(t_0, t_1)
    if slot.N_total − slot.match() > M
        accept ← false
if accept = false
    for slot = s_0 to s_1
        slot.remove(t_0, t_1)


INSERT(t_0, t_1)
N_total ← N_total + 1
if slot contains t_0 then insert_head(t_0)
if slot contains t_1 then insert_head(t_1)


MATCH()
i ← 0
j ← 0
matched ← 0
while Neither i nor j have passed the end of heads and ends, respectively
    if heads[i] ≤ ends[j]
        matched ← matched + 1
        i ← i + 1
    j ← j + 1
return matched
```

2.5.3 Complexity analysis

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 header packet 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 $O(logM)$. Therefore, the overall worst case time complexity is $O(1)+O(M)+O(logM) = O(M)$. 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 $O(1) + O(M/K) + O(logM) = O(M/K + logM)$. As an example, let $M = 256$ 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 burst 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 comparison, it is possible to perform a parallel search across the wavelengths to find all the unused wavelengths for LAUC-VF. Then the search results are compared to each other to find the latest available one. These operations can be performed in $O(logM)$ 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 that for Ordered Scheduling.

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 [46, 47]. They have $O(1)$ time complexity with regard to queue size. Implemented on conservative technologies such as 0.35-micron and 0.18-micron 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. Note that the above priority queue implementations can accommodate any queue size of practical interest. The queue size only affects the amount of required memory.

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 packet and passes it to the priority queue, it can handle the next header packet while the first header packet 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 for Enhanced Ordered Scheduling are $O(M)$ and $O(M/K + logM)$, respectively.

2.5.4 Performance study

In [45], a simulation model is used to evaluate the performance of the Ordered Scheduling algorithms. 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. The reason for the difference in the chosen slot sizes is because the performance of Basic Ordered Scheduling critically depends on the slot size while the performance of Enhanced Ordered Scheduling does not. LAUC-VF operating under the same condition is used for comparison.

The burst assembly algorithm is a simple time-based algorithm with a time limit T_{limit}. There are separate assembly queues for each ingress 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, T_{limit} is set 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.

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 an entire network to investigate the effect of network topology on the performance trend among the algorithms.

The simulation topology for the first two sets of experiments is shown in Figure 2.16. 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.

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 8 data wavelengths. The core node has an FDL buffer with 6 FDLs of lengths 5 μs, 10 μs,..., 30 μs.

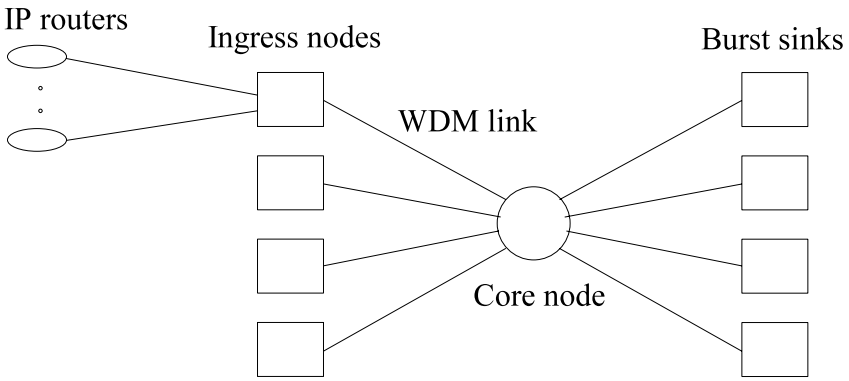


Fig. 2.16. Topology for simulation study of a single core node

Firstly, the effect of varying the offered load to the core node is examined. For this experiment, two offset-based QoS classes 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 Figure 2.17. They show that the burst dropping probabilities increase with increasing offered load, which is expected. Among the algorithms, Enhanced Ordered Scheduling has the best burst dropping performance followed by Basic Ordered Scheduling and then LAUC-VF. This order of burst dropping performance among the algorithms is as expected based on the discussion in the previous sections. The order of performance is the same in virtually all of the following experiments. 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 as-

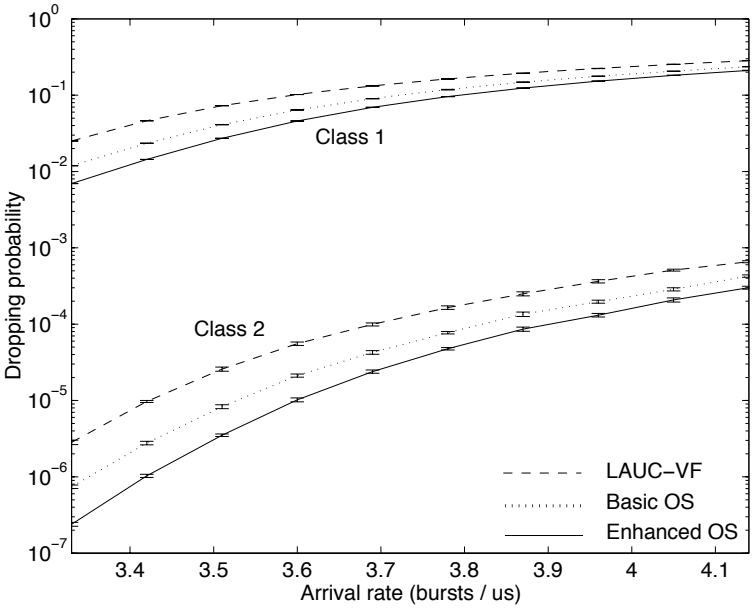


Fig. 2.17. Burst loss probability versus traffic loading

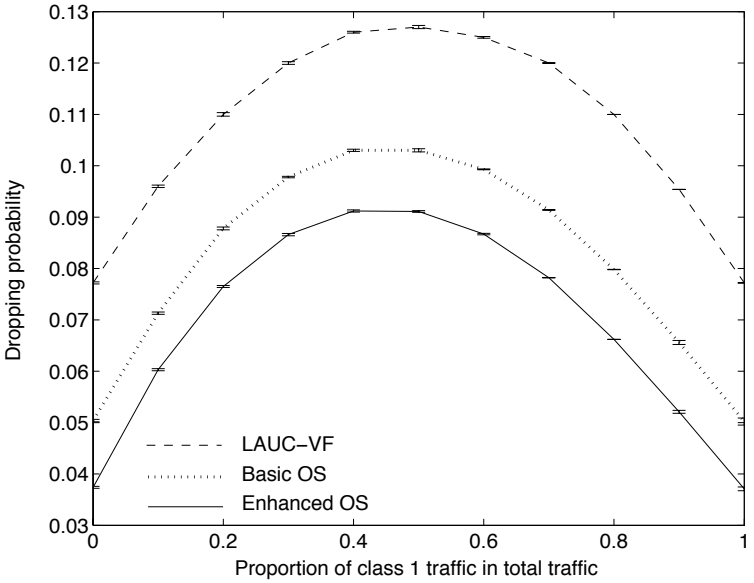
signment decisions due to incomplete knowledge of other burst reservations. Between the two classes, it is observed that the performance improvement of Ordered Scheduling over LAUC-VF is greater for class 2 than it is for class 1. The reason for this is also related to loading. Since full isolation is 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. The same traffic parameters as above are used 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, Figure 2.18(a) shows that the overall traffic loss rate follows a bell-shaped curve, which is slightly tilted towards the left. The burst 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

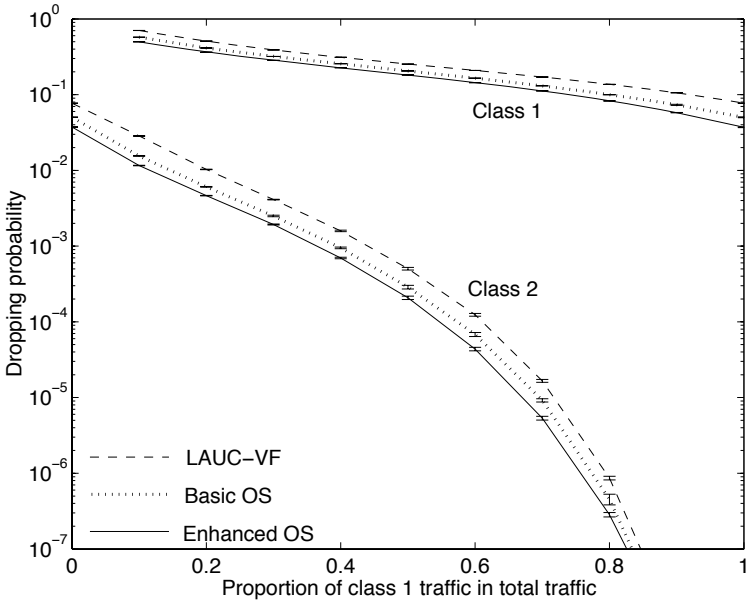
becomes more balanced, more class 1 bursts are preempted by those from class 2. When burst B_1 from class 1 is preempted by burst B_2 from class 2, the effective size of B_2 is its actual size plus the portion already served of B_1. If the burst size distribution is not exponential, that will increase 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. 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.

The loss rates of individual classes are shown in Figure 2.18(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.

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. The overall burst dropping probabilities are plotted in Figure 2.19. It shows that the overall burst 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.



(a) Overall performance



(b) Performance of individual traffic classes

Fig. 2.18. Effect of traffic composition on burst loss probability

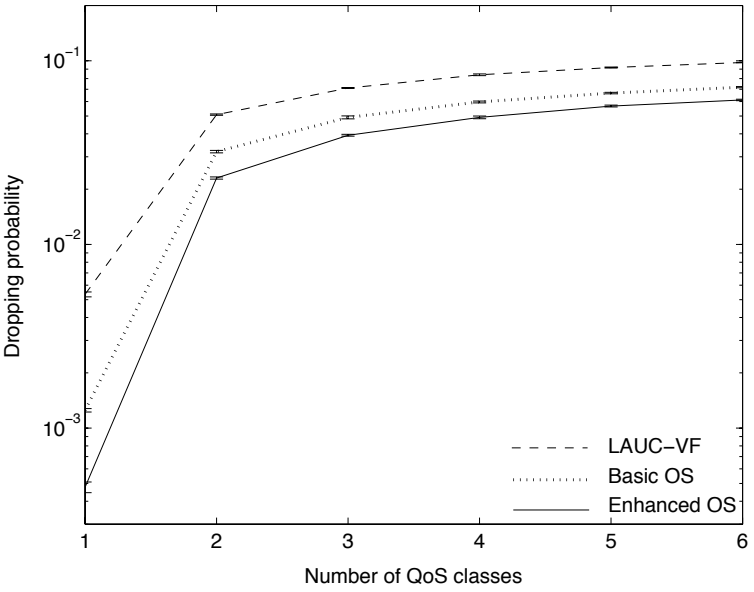


Fig. 2.19. Overall burst loss probability versus number of traffic classes

Effects of hardware configuration

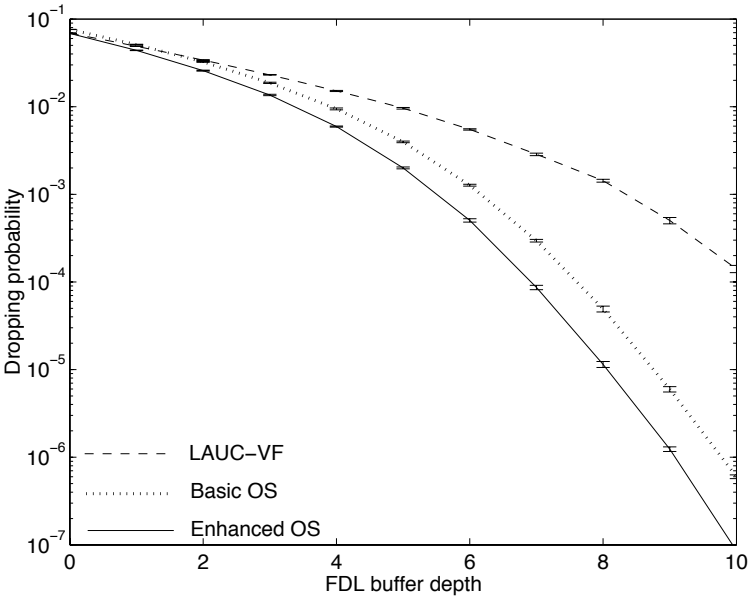
In the first experiment, the impact of FDL buffer depth on the performance of the algorithms is studied. Two kinds of data traffic are considered: one with a single QoS class and the other with two offset-based QoS classes. This is because the effects on burst dropping performance are slightly different 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.

Figure 2.20 shows 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 burst dropping performance. Note also 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

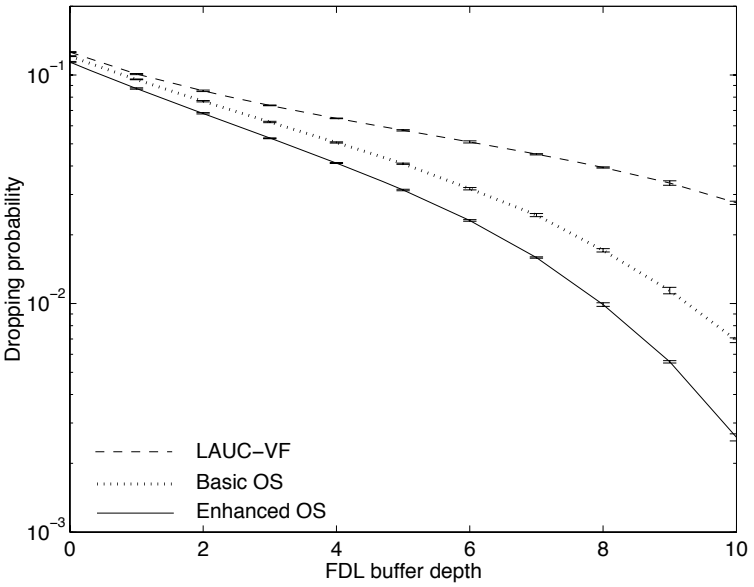
more FDLs are introduced. For LAUC-VF, this negative effect opposes and neutralises the beneficial effect of having more FDLs, which explains the levelling off of its curve. Ordered Scheduling, on the other hand, is not affected due to its deferment of scheduling decisions. The above effect is more pronounced in Figure 2.20(a) than it is in Figure 2.20(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 Figure 2.21. 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 burst 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 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. The performance results of a non-void filling scheduling scheme as described in section 2.2.1 is included. The purpose is to see how its performance compares to those of other void filling algorithms at different numbers of wavelengths. The burst dropping probabilities are measured at an overall offered load of 0.8 and different numbers of wavelengths per link and plotted in Figure 2.22. 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. Observe



(a) Single traffic class



(b) Two traffic classes

Fig. 2.20. Burst loss probability versus buffer depth

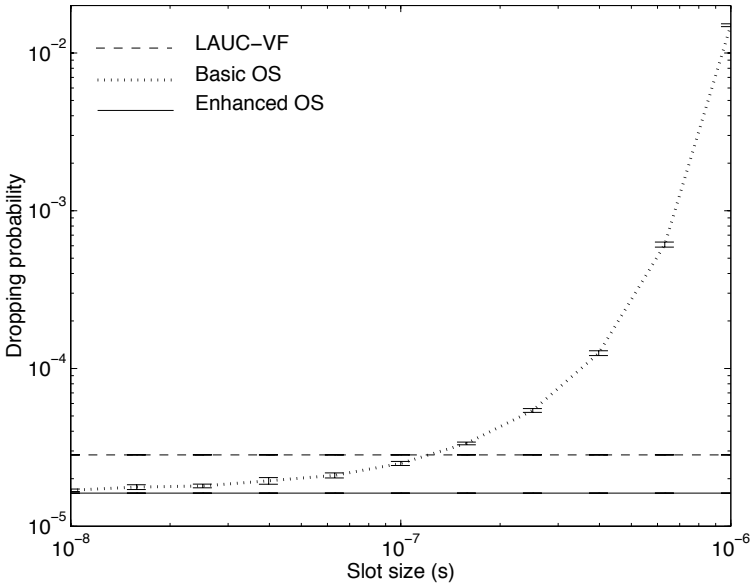


Fig. 2.21. Performance of Basic Ordered Scheduling with different slot size

also 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, 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. As a slot handles more and more bursts, the chance that Basic Ordered Scheduling over-reports the number of occupied wavelengths as illustrated in Figure 2.15 increases. From this experiment and the previous one, it is observed that the performance of Ordered Scheduling depends on the ratio between the number of slots per burst and the number of wavelengths per link.

Simulation study for an entire network

The three scheduling algorithms are next simulated in a realistic network setting to see if the network topology affects the performance trend among the algorithms. For this experiment, the

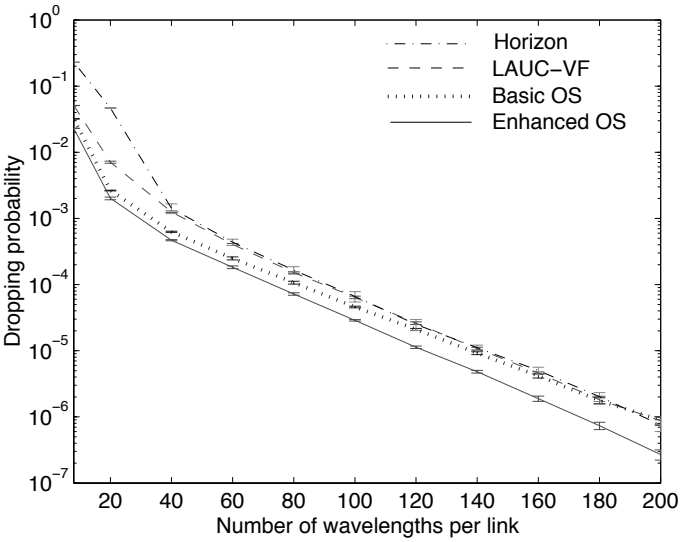


Fig. 2.22. Burst loss probability versus number of wavelengths per link

topology in Figure 2.23, which is a simplified topology of 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. The IP packet arrival rates are the same for every node. The header packet 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.

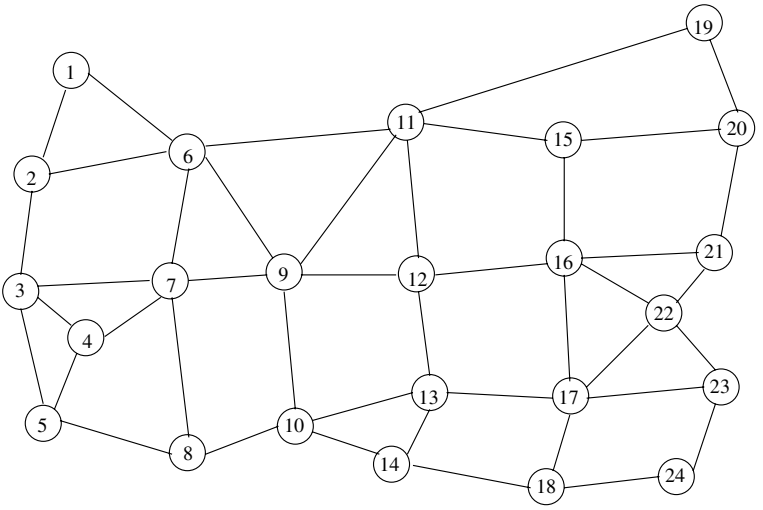


Fig. 2.23. 24-node NSF network topology

The burst dropping probabilities for the algorithms are plotted 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 Figure 2.24. It is observed that the performance trend is similar to that in Figure 2.17. 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. Note that the arrival rates used in this experiment are much smaller than those used in Figure 2.17 but the ranges of the burst 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.

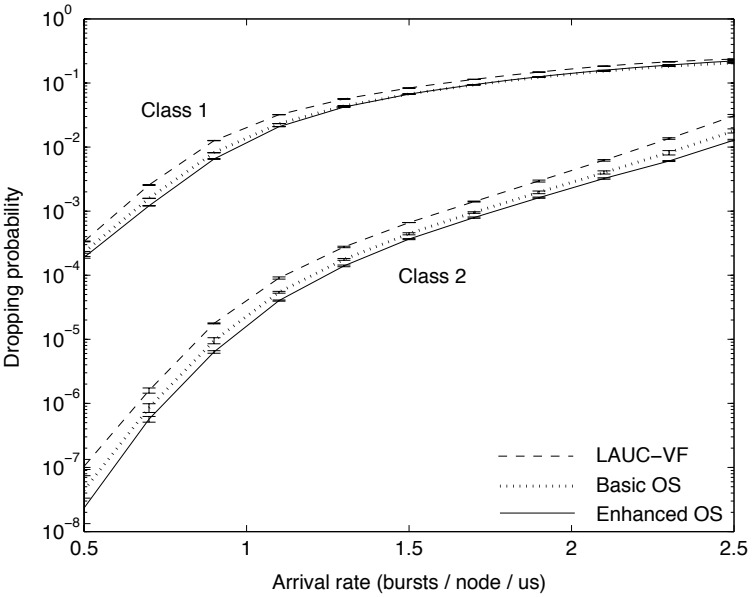


Fig. 2.24. Average burst loss probability for NSFNET versus average network load

References

1. D. B. Sarrazin, H. F. Jordan, and V. P. Heuring, "Fiber Optic Delay Line Memory," *Applied Optics*, vol. 29, no. 5, pp. 627–637, 1990.
2. I. Chlamtac *et al.*, "CORD: Contention Resolution by Delay Lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014–1029, 1996.
3. D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in Optical Packet Switches," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2081–2094, 1998.
4. W. D. Zhong and R. S. Tucker, "A New Wavelength-Routed Photonic Packet Buffer Combining Traveling Delay Lines with Delay Line Loops," *IEEE/OSA Journal of Lightwave Technology*, vol. 19, no. 8, pp. 1085–1092, 2001.
5. T. Zhang, K. Lu, and J. P. Jue, "Shared Fiber Delay Line Buffers in Asynchronous Optical Packet Switches," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 118–127, 2006.
6. K. K. Merchant *et al.*, "Analysis of an Optical Burst Switching Router With Tunable Multiwavelength Recirculating Buffers," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3302–3312, 2005.
7. D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: A Switch with Large Optical Buffers for Packet Switching," *IEEE/OSA Journal of Lightwave Technology*, vol. 16, no. 10, pp. 1725–1736, 1998.
8. I. Chlamtac, A. Fumagalli, and C. J. Shu, "Multibuffer Delay Line Architectures for Efficient Contention Resolution in Optical Switching Nodes," *IEEE Transactions on Communications*, vol. 48, no. 12, pp. 2089–2098, 2000.
9. N. Ogashiwa, H. Harai, N. Wada, F. Kubota, and Y. Shinoda, "Multi-Stage Fiber Delay Line Buffer in Photonic Packet Switch for Asynchronously Arriving Variable-Length Packets," *IEICE Transactions on Communications*, vol. E88-B, no. 1, pp. 258–265, 2005.
10. R. S. Tucker, P. C. Ku, and C. J. Chang-Hasnain, "Slow-Light Optical Buffers: Capabilities and Fundamental Limitations," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 12, pp. 4046–4066, 2005.
11. A. S. Acampora and S. 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, 1992.
12. A. G. Greenberg and B. Hajek, "Deflection Routing in Hypercube Networks," *IEEE Transactions on Communications*, vol. 40, no. 6, pp. 1070–1081, 1992.

13. F. Forghieri, A. Boroni, 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, 1995.
14. T. Chich, J. Cohen, and P. Fraigniaud, "Unslotted Deflection Routing: A Practical and Efficient Protocol for Multihop Optical Networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 47–59, 2001.
15. S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "A Unified Study of Contention-Resolution Schemes in Optical Packet-Switched Networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 21, no. 3, pp. 672–683, 2003.
16. C.-F. Hsu, T.-L. Liu, and N.-F. Huang, "Performance Analysis of Deflection Routing in Optical Burst-Switched Networks," in *Proc. IEEE Infocom*, 2002, pp. 66–73.
17. S. Lee, K. Sriram, H. Kim, and J. Song, "Contention-Based Limited Deflection Routing Protocol in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 8, pp. 1596–1611, 2005.
18. N. Ogino, and H. Tanaka, "Deflection Routing for Optical Bursts Considering Possibility of Contention at Downstream Nodes," *IEICE Transactions on Communications*, vol. E88-B, no. 9, pp. 3660–3667, 2005.
19. C. Cameron, A. Zalesky, and M. Zukerman, "Prioritized Deflection Routing in Optical Burst Switching," *IEICE Transactions on Communications*, vol. E88-B, no. 5, pp. 1861–1867, 2005.
20. A. Detti, V. Eramo, and M. Listanti, "Performance Evaluation of a New Technique for IP Support in a WDM Optical Network: Optical Composite Burst Switching (OCBS)," *IEEE/OSA Journal of Lightwave Technology*, vol. 20, no. 2, pp. 154–165, 2002.
21. V. M. Vokkarane and J. P. Jue, "Burst Segmentation: An Approach for Reducing Packet Loss in Optical Burst Switched Networks," *SPIE/Kluwer Optical Networks*, vol. 4, no. 6, pp. 81–89, 2003.
22. V. M. Vokkarane, , G. P. Thodime, V. U. Challagulla, and J. P. Jue, "Channel Scheduling Algorithms using Burst Segmentation and FDLs for Optical Burst-Switched Networks," in *Proc. IEEE International Conference on Communications*, 2003, pp. 1443–1447.
23. W. Tan, S. Wang, and L. Li, "Burst Segmentation for Void-Filling Scheduling and Its Performance Evaluation in Optical Burst Switching," *Optics Express*, vol. 12, no. 26, pp. 6615–6623, 2004.
24. Z. Rosberg, H. L. Vu, M. Zukerman, and J. White, "Performance Analyses of Optical Burst Switching Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1187–1197, 2003.
25. V. Eramo and M. Listanti, "Packet Loss in a Bufferless Optical WDM Switch Employing Shared Tunable Wavelength Converters," *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 1818–1833, 2000.
26. V. Eramo, M. Listanti, and P. Pacifici, "A Comparison Study on the Number of Wavelength Converters Needed in Synchronous and Asynchronous All-Optical Switching Architectures," *IEEE/OSA Journal of Lightwave Technology*, vol. 21, no. 2, pp. 340–355, 2003.
27. M. Yao, Z. Liu, and A. Wen, "Accurate and Approximate Evaluations of Asynchronous Tunable Wavelength Converter Sharing Schemes in Optical Burst Switched Networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 10, pp. 2807–2815, 2005.

28. G. Shen, S. K. Bose, T. H. Cheng, C. Lu, and T. Y. Chai, "Performance Study on a WDM Packet Switch with Limited-Range Wavelength Converters," *IEEE Communications Letters*, vol. 5, no. 10, pp. 432–434, 2001.
29. Z. Zhang and Y. Yang, "Performance Modeling of Bufferless WDM Packet Switching Networks with Wavelength Conversion," in *Proc. IEEE Globecom*, 2003, pp. 2498–2502.
30. V. Eramo, M. Listanti, and M. Spaziani, "Resources Sharing in Optical Packet Switches with Limited-Range Wavelength Converters," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 2, pp. 671–687, 2005.
31. J. S. Turner, "Terabit Burst Switching," *Journal of High Speed Network*, vol. 8, no. 1, pp. 3–16, 1999.
32. Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control Architecture in Optical Burst-Switched WDM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838–1851, 2000.
33. L. Tančevski, S. Yegnanarayanan, G. Castanon, L. Tamil, F. Masetti, and T. McDermott, "Optical Routing of Asynchronous, Variable Length Packets," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2084–2093, 2000.
34. M. Iizuka, M. Sakuta, Y. Nishino, and I. Sasase, "A Scheduling Algorithm Minimizing Voids Generated by Arriving Bursts in Optical Burst Switched WDM Network," in *Proc. IEEE Globecom*, 2002, pp. 2736–2740.
35. M. Ljolje, R. Inkret, and B. Mikac, "A Comparative Analysis of Data Scheduling Algorithms in Optical Burst Switching Networks," in *Proc. Conference on Optical Network Design and Modeling*, 2005, pp. 493–500.
36. S. Q. Zheng, Y. Xiong, and H. C. Cankaya, "Hardware Design of a Channel Scheduling Algorithm for Optical Burst Switching Routers," in *Proc. SPIE*, vol. 4872, 2002, pp. 199–209.
37. J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient Burst Scheduling Algorithms in Optical Burst-Switched Networks Using Geometric Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 9, pp. 1796–1811, 2004.
38. F. Farahmand, and J. P. Jue, "Look-Ahead Window Contention Resolution in Optical Burst Switched Networks," in *Proc. IEEE Workshop on High Performance Switching and Routing*, 2003, pp. 147–151.
39. S. Charcranoon, T. S. El-Bawab, J. D. Shin, and H. C. Cankaya, "Group Scheduling for Multi-Service Optical Burst Switching (OBS) Networks," *Photonic Network Communications*, vol. 11, no. 1, pp. 99-110, 2006.
40. H. Li, H. Neo, and L. J. I. Thng, "Performance of the Implementation of a PipeLine Buffering System in Optical Burst Switching Networks," in *Proc. IEEE Globecom*, 2003, pp. 2503–2507.
41. J. Li, C. Qiao, and Y. Chen, "Maximizing Throughput for Optical Burst Switching Networks," in *Proc. IEEE Infocom*, 2004, pp. 1853–1863.
42. N. Barakat, and E. H. Sargent, "Separating Resource Reservations from Service Requests to Improve the Performance of Optical Burst Switching Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 95–107, 2006.
43. S. K. Tan, G. Mohan, and K. C. Chua, "Algorithms for Burst Rescheduling in WDM Optical Burst Switching Networks," *Computer Networks*, vol. 41, no. 1, pp. 41–55, 2003.

44. S. K. Tan, G. Mohan, and K. C. Chua, "Burst Rescheduling with Wavelength and Last-Hop FDL Reassignment in WDM Optical Burst Switching Networks," in *Proc. IEEE International Conference on Communications*, 2003, pp. 1448–1452.
45. M. H. Phung, K. C. Chua, G. Mohan, M. Motani, T. C. Wong, and P. Y. Kong, "On Ordered Scheduling for Optical Burst Switching," *Computer Networks*, vol. 48, no. 6, pp. 891–909, 2005.
46. R. Bhagwan and B. Lin, "Fast and Scalable Priority Queue Architecture for High-Speed Network Switches," in *Proc. IEEE Infocom*, 2000, pp. 538–547.
47. A. Ioannou and M. Katevenis, "Pipelined Heap (Priority Queue) Management for Advanced Scheduling in High-Speed Networks," in *Proc. IEEE International Conference on Communications*, 2001, pp. 2043–2047.