Available online at www.sciencedirect.com
SCIENCE DIRECT®

Computer Communications 28 (2005) 754–764

computer communications

www.elsevier.com/locate/comcom

Efficient techniques for improved QoS performance in WDM optical burst switched networks

G. Mohan[*], K. Akash, M. Ashish

Department of Electrical and Computer Engineering, National University of Singapore, Engineering Drive 4, Level 5, Singapore, Singapore 117576

Received 11 November 2003; revised 7 October 2004; accepted 26 October 2004
Available online 10 November 2004

Abstract

In this paper, we address the problems of *burst scheduling*, *network fairness*, and *service differentiation* in *wavelength division multiplexing* (WDM) *optical burst switched* (OBS) networks. We propose two approaches—*time-slotting* and *burst fragmentation*—to provide efficient solutions for the above problems. We also propose a path selection algorithm to distribute label switched paths (LSP) based on load balancing. Time slotting refers to the quantization of time into slots of fixed sizes. Time slotting aids fast implementation of scheduling algorithms. Further, the slotting is local to the nodes and they need not be synchronized across the network. In burst fragmentation, a burst is fragmented instead of being dropped if it cannot be accommodated on any wavelength as a whole. These fragments are then sent on different wavelengths on the same path. Only in the event, that a fragment cannot be scheduled on any wavelength of the link, is the burst dropped. We develop a new scheduling algorithm based on time-slotting and fragmentation. This algorithm has several attractive features such as fast and simple implementation and improved burst dropping performance. Using the hop-based shortest path algorithm to route a burst without accounting for the link load results in poor burst dropping performance. This is because, some links could be heavily loaded forming a bottleneck for network performance. Load balancing approach aims at reducing the bottleneck of data traffic in the network. We develop a path selection algorithm based on load balancing with the objectives of improving network performance in terms of burst dropping probability and improve fairness among bursts traversing paths of different lengths. We then develop an offset-based algorithm to provide inter-class service differentiation and intra-class fairness for the bursts belonging to classes with different levels of priority. An attractive feature of this algorithm is that it works with a constraint of maximum permissible initial offset time in order to reduce the burst transfer delay and also to reduce the buffer requirements at the ingress edge routers. It also ensures that shorter-hop bursts of a low-priority class does not perform better than the longer-hop bursts belonging to a high-priority class. Extensive simulation results have been used to demonstrate the effectiveness of the proposed approaches and algorithms for various *identical* and *non-identical* traffic demand scenarios.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Wavelength division multiplexing (WDM); Optical burst switching (OBS); Network fairness; Service differentiation; Time-slotting; Fragmentation; Load balancing

1. Introduction

Potential bottlenecks of electronic processing to carry Internet protocol (IP) traffic over wavelength division multiplexing (WDM) optical networks can be overcome by aggregating multiple data packets into a super packet, called burst which is assembled at the ingress router. Bursts have the same network properties like ingress and egress routers and QoS requirements. In burst switching, resources are reserved in a one-way process and the burst cuts through the intermediate nodes, without any need for buffering, thus reducing nodal complexity and alleviating synchronization problems as opposed to packet switching [1–4].

A burst consists of a *header* and a payload called *data burst*. The burst header is also called the *control packet*. The payload and the header are sent separately on different wavelengths/channels. The burst header contains all the necessary information to be used by the switch control unit (SCU) at each hop to schedule the data burst, and configure the optical switching matrix to switch the data burst optically [5]. The difference between the times at which a control packet arrives and leaves a node is called

[*] Corresponding author. Tel.: +65 6874 4688; fax: +65 6779 1103.
E-mail address: elegm@nus.edu.sg (G. Mohan).

the *control processing time*, denoted by Δ. In order to avoid buffering of the burst at the intermediate nodes, the control header precedes the payload by a minimum duration of $h\Delta$ where h is the number of hops along the path. The separate transmission and switching of data bursts and their headers help to facilitate the electronic processing of headers and lower the opto-electronic processing capacity required at core routers. Further, it provides ingress to egress transparent optical paths for transporting data bursts.

In the literature, several burst switching techniques have been proposed. These include close-ended resource reservation techniques like Reserve-a-fixed-duration (RFD) and open-ended reservation techniques like Tell-and-go (TAG) and In-band-terminator (IBT). Among the protocols available, Just-enough-time (JET), an RFD based OBS protocol is attractive [2]. The JET protocol works as follows. After the burst assembly is complete, the ingress node sends out a control packet followed by the data burst after an initial offset time. The control packet carries information such as burst duration and offset time. Because the burst is buffered at the source for a sufficient amount of time, no fiber delay lines (FDL's) are necessary at intermediate nodes to delay the burst. When the control packet arrives at a node, a wavelength is reserved on the relevant outgoing link at the node from the time when the burst is expected to arrive for the duration of the burst. If no wavelength is available, the burst is said to be blocked and will be dropped.

As bursts arrive dynamically and multiple bursts may attempt to traverse a link simultaneously, resource contention may occur. Therefore, an efficient scheduling algorithm is required to allocate wavelengths to various contending bursts. In the literature, a few burst scheduling algorithms have been proposed [1,5]. They include First Fit Unscheduled Channel (FFUC), latest available unscheduled channel (LAUC) and, latest available unused channel with void filling (LAUC-VF) [5]. The *scheduling horizon* algorithm developed in [1] is similar to LAUC scheduling algorithm.

Both FFUC and LAUC [5] do not examine and fill voids formed on the wavelength channels. Instead, they use only the last burst information on each wavelength. By doing so, they become computationally simple but the scheduling efficiency is compromised. The worst case time complexity is $O(W)$ where W is the number of wavelengths per link. Among all the eligible wavelengths FFUC chooses the first free wavelength whereas LAUC chooses the one whose latest available time is closest to the arrival time of the new burst. By doing so, LAUC attempts to pack the bursts as tightly as possible. This improves the chances of successfully scheduling future bursts. LAUC-VF performs significantly better than LAUC by allowing a new data burst to fill the voids between the existing bursts. However, it is computationally complex. Its worst case time complexity is $O(NW)$, where N is the number of voids or bursts currently scheduled.

Internet traffic has different service requirements in terms of delay and loss. Therefore, several classes of bursts with different grades of QoS requirements such as burst loss performance need to be handled. In the literature, different kinds of approaches have been proposed to deal with service differentiation. In the first approach, class isolation is ensured by assigning extra offset time of $3L$ to high-priority class bursts, where L is the mean burst duration [6]. This approach might over-penalize the low-priority class in order to ensure class isolation. In the second approach proportional QoS is ensured between different classes. In [7], proportional QoS is provided by intentionally dropping low-priority bursts when relative performance between classes is violated. This method ensures a fair level of performance by low-priority bursts. However, it could lead to poor utilization of bandwidth or wavelength resources due to the intentional dropping.

An important issue in OBS networks is fairness. In most networks, the dropping probability of a burst increases with the number of hops in its path. This implies that for any given ingress–egress pair the burst dropping probability is not uniform but depends on the path length from the ingress to the egress, which is undesirable. In the literature, a solution has been suggested to achieve fairness in OBS networks. In this, the initial offset is made a function of the number of hops the burst has to traverse, typically $5hL$ or $10hL$, where h is the number of hops and L is the mean burst duration [2]. Increasing offset time improves the probability of the burst getting transmitted through the network and hence helps in reducing the dropping probability for the longer-hop bursts. This, however, increases the time a burst spends in the network and can become quite undesirable. Further, due to the increased offset time, large buffers are required at the ingress edge routers. We note that the fairness problem can also be caused due to other reasons such as length of burst duration. In our work, we consider the network fairness problem caused by differing hop lengths of the paths traversed by bursts.

We propose two new approaches—*time slotting* and *burst fragmentation* and a new path selection algorithm based on *load balancing*—as the effective ways of improving burst dropping performance, network fairness, inter-class service differentiation, and intra-class fairness. Based on the time-slotting and fragmentation approaches, we develop a new scheduling algorithm called Best fit void filling with fragmentation (BFVFF). This algorithm has advantages of computational simplicity and improved burst dropping performance. Time slotting refers to the quantization of time into slots of fixed size. The burst duration is specified as the number of slots and every node performs scheduling in terms of slots. This helps reduce the number of bits required to encode the burst duration. Further, the maintenance of the status of slots and implementation of scheduling at the core nodes become simpler. Fragmentation allows a burst to be fragmented when it cannot be scheduled as a single entity on any wavelength. It enables allocation of multiple wavelengths on the same path to different fragments of a burst.

With the use of multi-protocol label switching (MPLS) capabilities in OBS networks, label switched paths (LSPs) can be used for transporting the bursts between node pairs [8]. In MPLS, the control packets carry only a small label which identifies the route that needs to be taken by the burst. Further, the LSPs could be setup by signaling which are initiated by ingress edge routers on the specified explicit route. Usually, a hop-based shortest path algorithm is used to route a burst without accounting for the link load. This can result in poor burst dropping performance if some links are heavily loaded. Load balancing aims at reducing the bottleneck of data traffic in the network. We develop a path selection algorithm based on load balancing with the objectives of improving network performance in terms of burst dropping probability and improve fairness among bursts traversing paths of different lengths. We then develop an offset-based algorithm to provide inter-class service differentiation and intra-class fairness for the bursts belonging to classes with different levels of priority. An attractive feature of this algorithm is that it works with a constraint of maximum permissible initial offset time in order to reduce the burst transfer delay and also to reduce the buffer requirements at the ingress edge routers. It also ensures that shorter-hop bursts of a low-priority class does not perform better than the longer-hop bursts belonging to a high-priority class.

We discuss the implementation issues of the proposed approaches in the following sections. We use extensive simulation results to demonstrate the effectiveness of the proposed approaches and algorithms for various *identical* and *non-identical* traffic demand scenarios.

The rest of the paper is organized as follows. In Section 2, the proposed time-slotting and burst fragmentation approaches are discussed. The proposed scheduling algorithm based on the above approaches is also described. Section 3 discusses the load balancing approach and presents a path selection algorithm based on this approach. In Section 4, the proposed method for providing inter-class service differentiation and intra-class fairness in a multi-class environment is described. Section 5 studies the performance of the proposed approaches and algorithms. Finally, concluding remarks are made in Section 6.

2. Proposed scheduling algorithm

A burst scheduling (also referred to as wavelength scheduling or channel scheduling) algorithm has two vital aspects, its computational complexity and its efficiency in scheduling network resources. We develop a scheduling algorithm called best fit void filling with fragmentation (BFVFF) with the objectives of achieving a better network performance than that of LAUC-VF, but at the same time, restricting the worst case time complexity close to that of the best known scheduling algorithm (LAUC-VF). The computational simplicity is achieved by using time-slotting and efficiency in scheduling resources is achieved by using burst fragmentation.

2.1. Time-slotting approach

Time-slotting refers to the quantization of time into slots of a fixed size. Each node in the network has a time-line for each wavelength on each of the outgoing links. This is used to schedule the outgoing bursts on different wavelengths on a specific link. The burst duration is expressed as an integral number of time slots. The time-slotting being referred to here is fundamentally different from the photonic slots used in the photonic slot routing in WDM packet-switched networks [9, 10]. In our time-slotting approach, there is no need for slot synchronization globally among the nodes. Time slots and their usage status are locally maintained by the nodes. Time-slotting helps in improving the computational speed of a void filling algorithm. The maintenance and lookup of the status of time slots could be done faster by using simple data structures such as arrays or bit vectors. It is also possible to implement these operations in hardware resulting in a much faster scheduling computation.

As the burst arrival time at a node is known while scheduling the burst, it can be used to directly reference the time-line of the resource (wavelength) being allocated, and the availability of the resource can be determined. No extensive search needs to be performed on the resource schedule. Thus, if there are W wavelengths on a particular link then the worst case time complexity to schedule a burst is $O(kW)$, where k is the maximum number of time slots a burst can span. The upper bound on the value of k is usually known as the slot size is fixed and the burst assembly techniques are generally based on maximum-burst-size or maximum-assembly-time schemes. The theoretical worst case complexity is therefore close to that of LAUC-VF algorithm whose complexity is $O(NW)$, where N is the number of voids or bursts currently scheduled. Regardless of the theoretical complexity, we note that the actual running time of time-slotting based algorithm is expected to be smaller than that of LAUC-VF algorithm due to the use of simple data structures as stated above.

The size of the time slot can be chosen based on various factors such as mean burst duration, network properties and statistics about the traffic of data. For good resource utilization, the time slot should be small compared to the burst duration. Since an integral number of time slots is allocated to a burst, the start and end of a burst might not be aligned to the beginning and ending of a time slot. Therefore, on an average, half a time slot is wasted on either end of the burst. A possible way to reduce this waste is to keep track of the latest available time within each slot. This allows two bursts to use the same slot (i.e. to use the same wavelength on a link during the same slot) if the beginning of a burst is later than the end of another burst. We note that this method requires constant number of extra operations per slot and hence it does not increase the worst case computational complexity of the scheduling algorithm. The smaller the time slot size, the lesser the wastage and the more efficient is the resource utilization.

However, the time-slot size cannot be reduced to a very great extent. This is because, the smaller the time-slot, the more the number of slots needed to schedule a burst. This adds on to the processing time. Thus, for very small values of time-slot the processing time may become unacceptably large. Thus, a good compromise needs to be achieved.

2.2. Burst fragmentation approach

Burst fragmentation refers to the splitting of a burst at a node if it cannot be accommodated as a whole, on any of the wavelengths of the particular link it is supposed to go on. In the conventional burst scheduling algorithms like LAUC-VF, each wavelength on a fiber is considered as a separate resource and a burst can be scheduled on only one of them at a time. This, however, leads to wastage on each of the wavelengths and the cumulative wastage on all the wavelengths is quite significant. If all the wavelengths on a given fiber can be treated as a single resource then this wastage can be minimized. This is the motivation for fragmenting bursts. As voids are more easily filled by shorter bursts, a greater number of longer bursts are dropped as compared to the shorter ones. Therefore, the conventional burst scheduling algorithms are unfair to longer bursts. This becomes another motivation for fragmentation.

A burst can be scheduled on one or more wavelengths as long as the required slots are available. For example, in Fig. 1, if we use a void-filling algorithm, the new burst B1 cannot be scheduled on either wavelength w_1 or w_2 and hence will be dropped. However, if we allow burst fragmentation, as shown in Fig. 2, then the burst can be accommodated on wavelengths w_1 and w_2 by fragmenting the burst at time t_3 and scheduling the first fragment B1′ on w_1 and the second fragment B1″ on wavelength w_2. When a burst is fragmented at a node each fragment of the burst is treated like an independent burst, thus enabling further fragmentation. The OBS nodes are assumed to have no buffers (i.e. FDLs) to delay bursts or their fragments to resolve contention. Therefore, a burst will be dropped at an OBS node if no wavelengths are available to accommodate the entire burst or all of its fragments.

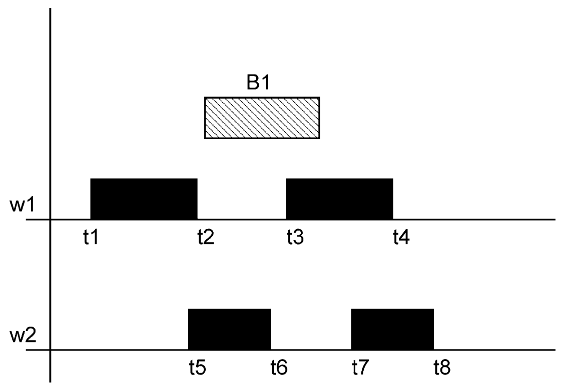

Fig. 1. A situation wherein a void-filling algorithm fails to schedule burst B1.

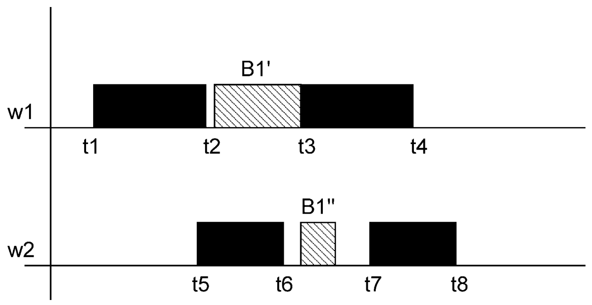

Fig. 2. A situation wherein BFVFF successfully schedules burst B1 by using burst fragmentation.

There are several issues that arise with fragmentation. If the permissible fragment size is an integral multiple of n slots, then at the end of every n slots, guard bits are inserted to account for the delay in switching to a new wavelength. This guard time is a very small percent of the burst duration and hence does not significantly add on to the burst size. We note that even without fragmentation, some time gap is needed when two bursts are scheduled in continuation. Another important issue is the signaling overhead. When a burst is fragmented, the information needs to be carried to the downstream nodes. There are two possible ways to handle this. The first way is to add the fragment information to the original control packet. The second way is to modify the burst duration in the original control packet and transmit a new control packet corresponding to the new fragment through the same path as that of the original burst. In both the cases, the effective control packet size and processing time increase. Therefore, signaling overhead depends on how many times a burst is fragmented. In Section 5, we show that the signaling overhead is very small. Thus, overall, the fragmentation scheme aids effective use of wavelength resources. We note that all the fragments of a burst traverse along the same path but on different wavelengths. Therefore, the start time and end time of the original burst are the same as the start time of the first fragment and end time of the \last fragment of the burst, respectively. They are finally processed at the egress router. From the control packet information, the egress router can infer which fragment comes at what time and on which wavelength.

Our work on burst fragmentation with time-slotting [12,13] is fundamentally different from a parallel work on *burst segmentation* [11]. In our scheme, all the fragments of a burst traverse the same path. Moreover, a burst is fragmented based on its own decision and it cannot be fragmented by other bursts. Further, a burst (with all its constituent fragments) is either successful or dropped as a whole. On the other hand, the segmentation scheme [11] adopts deflection routing of segments. It allows a burst to be segmented by other segments. Also, it is possible to schedule some segments while dropping the remaining segments. This results in increased out-of-order delivery of packets and unnecessary waste of bandwidth.

2.3. BFVFF scheduling algorithm

When a node receives a control packet, it extracts the information about burst arrival time and burst duration. It applies BFVFF scheduling algorithm to determine fragments and assignment of slots and wavelengths to the fragments. Let t_1 be the starting time slot of the burst. Let b be the burst duration expressed in number of slots. Our algorithm requires that the size of a fragment should be at least f_{min} slots. The key idea of the algorithm is to keep the burst (or fragment) size as long as possible. With this meaning we use the term 'best fit'. Therefore, a burst is fragmented only if it is necessary. In an iteration, the algorithm searches each wavelength starting from slot t looking for l contiguous free slots. The initial values of t and l are t_1 and b, respectively. It chooses wavelength w_{max} which has the maximum number (denoted by *max*) of free contiguous time slots starting from time t. If it is sufficient to accommodate the entire burst, the burst is not fragmented and the wavelength is chosen. Otherwise the burst is split into fragments F and F'. If the fragmentation does not meet the minimum size requirement of a fragment, the algorithm fails and the burst is dropped. Fragment F is of size *max* slots starting from slot t and fragment F' is of size $l-max$ starting from slot $t+max$. The algorithm allocates *max* slots on w_{max} to F and continues to search for free slots for F'. The above procedure is repeated until all the fragments are assigned wavelengths. If sufficient free slots are not available for a fragment, the burst is dropped. The maximum number of slots searched per wavelength in each iteration is *max* and the next iteration only starts from time $t+max$, thus at most b slots are searched on each wavelength. Therefore, the worst case time complexity is $O(bW)$. The pseudo-code of the scheduling algorithm is given below.

```
$l=b$; $t=t_1$;
while ($l>0$)
{
  Determine $w_{max}$ with maximum number of free contiguous slots from $t$;
  $max$ =number of free contiguous slots on $w_{max}$;
  if ($max=0$) sufficient slots are not available, fail;
  if ($max<l$)
    if (($l-max$)<$f_{min}$) $max=l-f_{min}$;
  else $max=l$
  if ($max<f_{min}$) fragment is too short, fail;
  Generate fragments $F$ of size $max$ slots starting from slot $t$ and $F'$ of size $l-max$ starting from slot $t+max$;
  Allocate $max$ slots on $w_{max}$ to $F$; Continue to search for free slots for $F'$;
  $l=l-max$; $t=t+max$;
}
```

3. Path selection based on load balancing

Multi-protocol label switching (MPLS) capabilities can be extended to OBS networks. This has several attractive features such as *explicit routing*. Here, label switched paths (LSPs) can be established for transporting the bursts between node pairs. This helps the control packets to carry only a small label. Usually, in OBS networks, shortest-hop paths are assumed to be used ignoring the network state and traffic demands between node pairs. This results in uneven traffic loads on different links resulting in a number of bottleneck links. In an OBS network, bursts are dropped once a resource bottleneck is reached. If the network traffic is concentrated only on a few nodes and links then this resource bottleneck is reached faster and hence a larger number of bursts are dropped. However, if paths are chosen in a way to balance the load on the links, it is less likely to create bottleneck links and hence the network performance increases in general. Thus, this feature is very important for the optimum performance of a network.

Another desired feature in OBS networks is to ensure fairness which means that for all ingress–egress pairs in the network a burst must have equal likelihood of getting through. It should be independent of the path length involved and thus the network should not be biased towards bursts which need to traverse a shorter path. The longer the path that a burst needs to traverse, the greater is the probability that it will find a resource bottleneck and will be dropped. Thus, if paths are selected based on load balancing then the chances of the burst getting dropped even on a longer path is lower, hence improving fairness.

We use load balancing with the dual objectives of improving overall burst dropping performance and network fairness for bursts traversing paths with different hop counts. We develop a path selection algorithm which balances the load on the links. We use the estimated average traffic demand (in terms of bursts per unit time) between node pairs as a measure of the load. As the traffic demand might change with time, the algorithm can be applied at varying time intervals. This makes the load balancing algorithm adaptive to the changing network traffic. Tarffic demands between node pairs could be estimated based on the past history of seasonal variations in traffic. Such traffic demands change on a relatively long time scale. Therefore, the overhead incurred due to the computation and setting up of paths is acceptably small.

An intuitive way of achieving load balancing is to make the path cost for using a network link a function of its length and the amount of traffic being carried on it. This will assign a high cost to a link which is heavily loaded and when the minimum-cost path is being computed for a given ingress–egress pair, this link will be avoided by the path selection algorithm. This is the main idea behind the proposed load balancing algorithm.

Let the traffic load carried by an LSP between node s and node d be t^{sd}. Let the link load information be stored in L, where l_{ij} is the load on the link between nodes i, j. We route LSPs between node pairs one by one in some order and incrementally update the link load. At any instance during the execution of the algorithm, the load on a link (or the cost of a link) is the sum of the load carried by the LSPs traversing that link. Initially the load on each link is a tiny value ϵ. The algorithm considers node pairs in the non-increasing order of their hop counts. First, the longest-hop node pair is considered. The minimum-cost path between the node pair based on the current link cost is selected. The cost of each link traversed by the path is then updated by a value equal to the traffic load between the node pair. Next, based on the new link costs, the minimum-cost path for the second longest-hop pair is chosen. This procedure continues until paths are chosen for all the node pairs. The pseudo-code for the algorithm is as follows:

```
Algorithm:

Initialize $L$:
  $l_{ij}=\epsilon$, if $i$ and $j$ are directly connected.
  $l_{ij}=\infty$, otherwise.
For every ingress--egress pair $\langle s,d\rangle$
compute hop count $w_{sd}$.
Sort the node pairs in the non-increasing
order of $w_{sd}$ into set $W_S$.
For every element in $W_S$, $\langle i,j,w_{sd}\rangle$,
  Find the minimum-cost path from $i$ to $j$
  based on the current link costs. Update
  $L$ by adding $t^{sd}$ to the cost of each of the
  links $l_{ij}$ traversed by the chosen path.
```

In the above algorithm, minimum-cost paths are computed for the ingress–egress pair having the maximum hop count prior to the ones having the smallest. This is expected to improve the dropping performance of longer-hop paths. The load balancing algorithm results in significant improvement in the burst acceptance ratio and a significant improvement in the fairness as verified through simulation in Section 5.

4. Inter-class service differentiation and intra-class fairness

Internet traffic carries different categories of data with different levels of priority. A high-priority class burst requires better QoS in terms of loss and delay than a low-priority class burst. We propose an offset-time based method to ensure service differentiation between classes and to ensure fairness among bursts with different hop counts within a class. We consider burst dropping probability as the QoS metric. Minimizing the burst loss will minimize the retransmissions and hence the burst or packet delivery time. Since the burst is switched in a pure optical domain from the ingress to the egress router, a small initial offset time at the ingress router will lead to reduced end-to-end delay.

The algorithm proposed for service differentiation uses the idea of varying the initial offset time for the burst to achieve service differentiation. When the initial offset for a burst is increased the chances of it being successfully transferred through the network increases. This is because resources required for the burst can be scheduled in advance. As the higher priority bursts need to have a higher acceptance ratio, they are assigned a high initial offset.

In order to reduce the burst transfer delay and buffer requirements at the ingress routers, we set an upper limit on the initial offset time. The maximum permissible offset range (which is predefined by a maximum offset factor) is equally divided into two and the higher part of this offset range is assigned to the higher priority service, and the lower to the normal priority service. The reason for dividing the possible initial offset region into two non-overlapping ranges is to isolate the performance of the two services. This idea can also be extended to any number of service types. If there are n service categories, then the offset range can be equally divided into n non-overlapping regions, such that the higher priority services get the higher offset. Although the increase in the maximum offset factor improves network performance and degree of class isolation in terms of acceptance ratio, the overall time spent by the burst in the network (which is defined by the initial offset time) increases. In real life implementation, the maximum offset factor should be chosen in such a way that it strikes a good compromise between improved performance in terms of burst dropping performance, degree of class isolation, and the end-to-end delay of the bursts.

When all the bursts of a particular class are allocated the same initial offset, cluttering takes place. Because of this the level of improvement in the performance of the algorithm is mitigated. In order to overcome this problem, in the proposed algorithm a further differentiation is done in terms of the initial offset, depending on the path length for each of the service categories. This in turn also improves the intra-class fairness in the network. This improvement is because of the fact that the longer paths are assigned a greater initial offset as compared to smaller paths. This in turn reduces the difference in the dropping probability for the different path lengths. For instance, if the maximum offset factor allowed is $O_{\max}$, then the permissible offset range is given by [0, $T_{\max}$], where $T_{\max}$ is given by $[h_{\max}\times\Delta+O_{\max}\times L]$. Here, L is the mean burst duration, Δ is the control processing time, and $h_{\max}$ is the maximum number of hops for any ingress–egress pair. Let n be the number of service categories with priorities p_s, 1,2,…,n. In the algorithm the Offset range is equally divided into $n\times h_{\max}$ segments. These offset segments are of size B. The bursts of hop length h and belonging to class p_s are then assigned the initial offset time O_{init} as determined below.

$$B = T_{\max}/(n\times h_{\max});$$

$$O_{\text{init}} = ((p_s-1)\times h_{\max}+(h-1))\times B+\Delta\times h;$$

We note that the above way of choosing the offset time values not only improves fairness within each class, but also ensures that the shorter-hop bursts belonging to a low-priority class do not perform better than the longer-hop bursts belonging to a high-priority class.

Unlike the service differentiation scheme discussed in [2], the method proposed in this paper, puts a ceiling to the maximum time a burst can spend in the network and also provides a way to increase fairness among bursts for the same data class. As the proposed method does a logical distribution of initial offsets among the different data classes, it is more scalable and can cater to multiple traffic classes without increasing the offset time beyond the limits specified.

5. Performance analysis

In this section, we verify the effectiveness of the BFVFF scheduling and path selection algorithms based on the proposed approaches of time-slotting and fragmentation through simulation. The simulation network is randomly generated with 32-nodes and 104 links. It is generated by creating a link between a node pair with a ceratin probability (in this case it is 0.2). Each link is bidirectional comprising two unidirectional fibers in opposite directions. Each fiber is assumed to carry eight data wavelengths and one control wavelength. Bursts arrive randomly according to Poisson process, with exponentially distributed duration with a mean of 24 μs. The Poisson model has been widely used by the research community since better and more accurate models for burst traffic are yet to be developed. We believe that our algorithms should exhibit similar behavior for other traffic models also. The destination nodes for the bursts are generated using a uniform distribution. The control processing time is assumed to be 3 μs. The relationship between the ratio of the number of data wavelengths and control wavelengths and the ratio between the mean burst length and control processing time [5] holds to ensure that the control packets are processed before the data bursts arrive at a node even under full link utilization condition. The time slot size is chosen to be 1 μs for most of the simulations; however, we do analyze the effect of varying slot size. The simulation experiments were run for a sufficiently long time and were repeated several times. The 95% confidence interval range is within 3% of the values plotted.

We consider identical as well as non-identical traffic demand scenarios to demonstrate that our algorithms can perform well for changing traffic patterns. In an identical traffic case, the traffic demands between all the node pairs are same. On the other hand, in a non-identical case, the traffic demands need not be same, meaning that the burst arrival rate between a node pair can be different from that of another node pair. Apart from the *mean burst dropping probability*, we use a new metric called *count of control packets per burst* to measure the signaling overhead that is caused by fragmentation.

5.1. Effect of traffic load and fragment size

We begin with comparing the dropping probability for different arrival rates for various fragment sizes. The arrival rate used here is the mean number of bursts generated per node, per mean burst duration. As can be seen from Fig. 3, when the burst is allowed to fragment, the dropping probability improves. For instance, for the arrival rate of 19, 8% bursts which are dropped in the 'no-fragmentation' case are accepted when completely fragmented. Since a WDM network carries a huge amount of traffic, an improvement of 8% is attractive in terms of extra data that can be transmitted through the network and in terms of reduction in retransmission overhead at the higher layer. Similar behavior can be observed for other arrival rates as well. A compromise between the two scenarios can also be seen when the burst is allowed to fragment only at the boundaries of 10 time slots. It can also be observed that the burst dropping probability decreases when the permissible fragment size decreases. This is because, shorter fragments are more likely to find free wavelengths when compared to longer fragments. The 'no fragmentation' algorithm exhibits the behavior of the LAUC-VF void filling algorithm.

5.2. Signaling overhead due to fragmentation

Fig. 4 depicts the signaling overhead that is caused by burst framentation for varying fragment sizes. We recall that, whenever a burst is fragmented additional information or control packets need to be sent to the downstream nodes. The mean number of fragments transmitted per burst on a link reflects the signaling overhead. For arrival rate 20, for example, an average of 1.003 control headers are sent for each link traversed. That is an increase of only 0.3% in

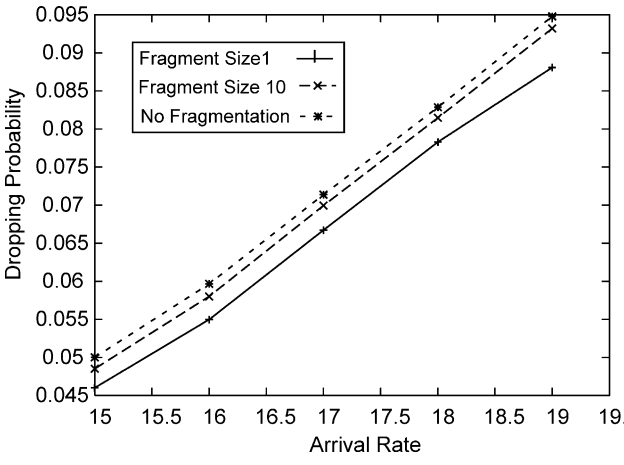

Fig. 3. Dropping probability vs arrival rate for different levels of fragmentation.

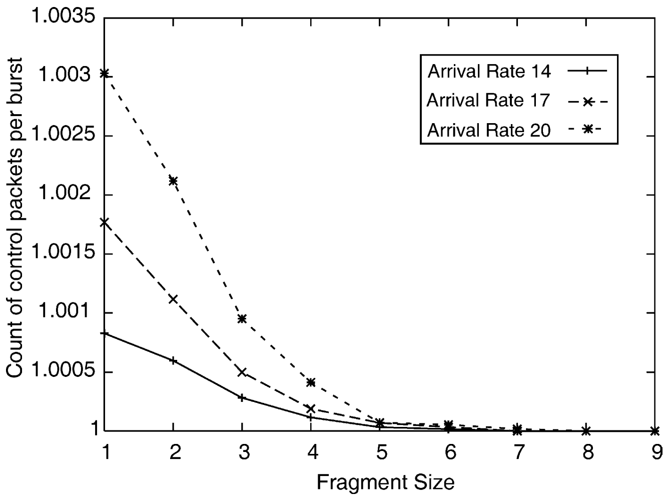

Fig. 4. Signaling overhead due to fragmentation.

control header traffic when the fragment size is 1. The ratio decreases with increasing fragment size as the chances of fragmentation drops. When the arrival rate increases the signaling overhead also increases. The reason is that at higher load, the chance of finding a single free wavelength for the whole burst is reduced resulting in fragmentation.

5.3. *Effect of time-slot size*

The effect of time-slot size is studied on the performance of the BFVFF algorithm as well as the signaling overhead. Fig. 5 shows the plot of dropping probability against time-slot size for different arrival rates. From Fig. 5 it can be seen that the dropping probability increases with increasing slot size. This is because of the fact that a large slot size results in resource wastage at the end slots of bursts. With the increase in the arrival rate per mean burst duration, the amount of increase in dropping probability for every time-slot size change increases. This is because, with the increasing arrival rate the amount of total resource wasted increases.

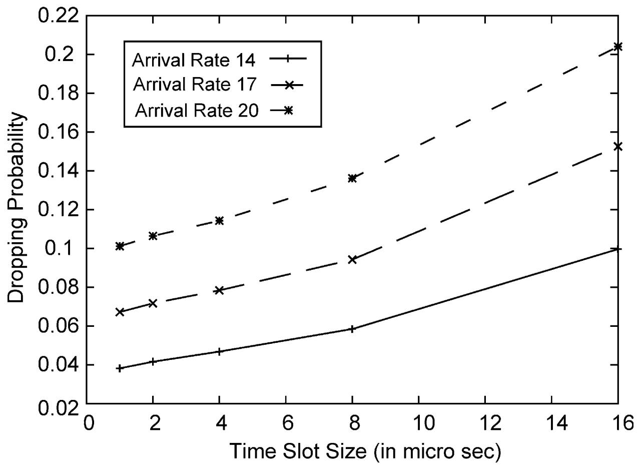

Fig. 5. Burst dropping probability vs time-slot size.

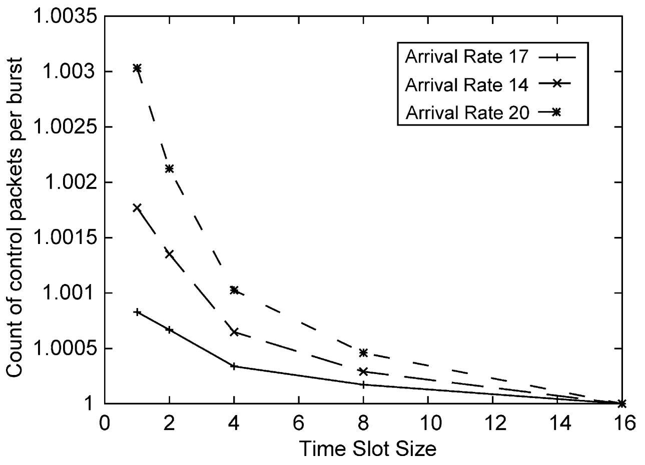

Fig. 6. Signaling overhead vs time-slot size.

Fig. 6 shows how the signaling overhead varies with the time-slot size. As the time-slot size increases, the chance of finding a free wavelength for the duration of a time slot decreases, and hence the possibility of fragmentation drops, thereby reducing the per burst control packet count.

5.4. *Effectiveness of load balancing*

(1) *Improvement in network performance*. The effect of using load balancing on both identical and non-identical traffic load was studied with the BFVFF algorithm. We recall that the identical load refers to the case where all the node pairs have the same mean burst arrival rate per mean burst duration whereas in the case of non-identical load, node pairs use different mean burst arrival rates. In both the cases the algorithm shows significant improvement after load balancing.

Fig. 7 shows the performance of the algorithm with and without load balancing for identical load. From the graph, it is apparent that the network performance increases significantly after load balancing. This is because the average resource utilization in the network increases,

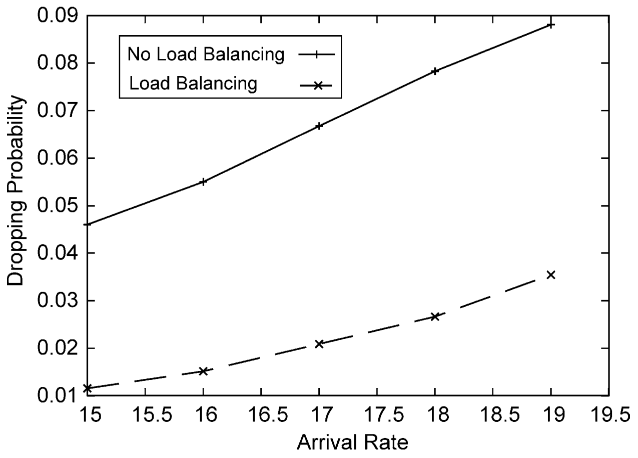

Fig. 7. Burst dropping probability with and without load balancing.

Table 1
Dropping probability for three different non-identical traffic loading patterns with and without load balancing

Loading pattern	Load balancing	No load balancing
*s*1	0.011067	0.028788
*s*2	0.012293	0.028069
*s*3	0.011641	0.027976

hence the throughput increases. From Fig. 7 it can be seen that after using load balancing the dropping probability decreases by about 60% for arrival rate of 19. In other words, the network can support a far higher arrival rate for a given dropping probability.

A similar observation can also be made from Table 1, which compares the burst dropping probability in a network for three different non-identical loading patterns *s*1, *s*2, and *s*3, with and without load balancing. The loading patterns are generated randomly with different seeds to obtain a network-wide mean arrival rate of 15 per node pair. Table 1 shows the dropping probability for the three different loading patterns. From Table 1, it can be seen that after load balancing the burst dropping probability decreases significantly for all the loading patterns.

(2) *Improvement in network fairness*. The impact of load balancing on network fairness has also been studied using the BFVFF algorithm on both identical and non-identical loads. Fig. 8 shows the burst dropping probability against the hop count, with and without load balancing for a uniform load. From the graph it is clear that load balancing not only prevents the dropping probability from increasing with hop count, but also reduces the magnitude for the dropping probability for each hop count quite significantly. For instance, for the hop count of 2, the burst dropping probability decreases by about 60% of the initial dropping probability after load balancing. Thus, load balancing has a major impact on network fairness.

Figs. 9–11 show the dropping probability against hop count for the non-identical traffic loading patterns *s*1, *s*2 and *s*3, respectively. The trend is similar to that seen in Fig. 8.

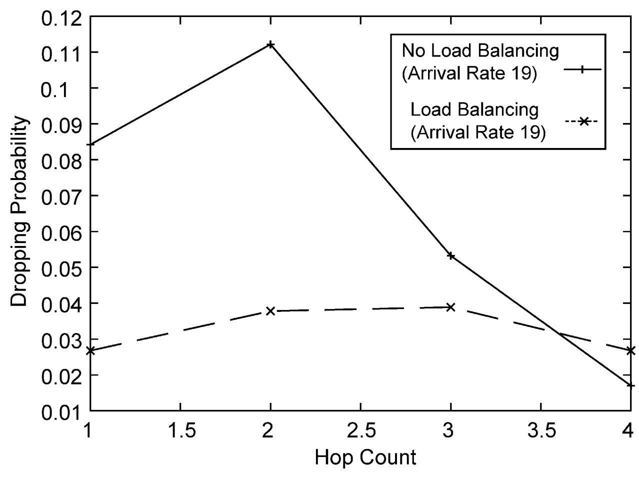

Fig. 8. Impact of load balancing on network fairness for uniform load.

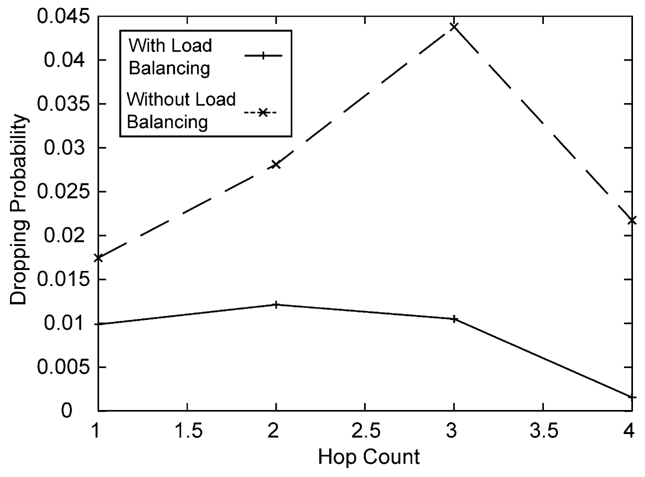

Fig. 9. Impact of load balancing on network fairness for non-identical traffic load (*s*1).

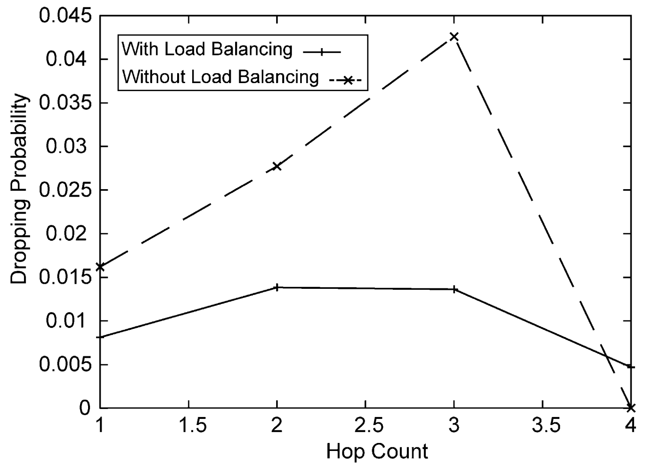

Fig. 10. Impact of load balancing on network fairness for non-identical traffic load (*s*2).

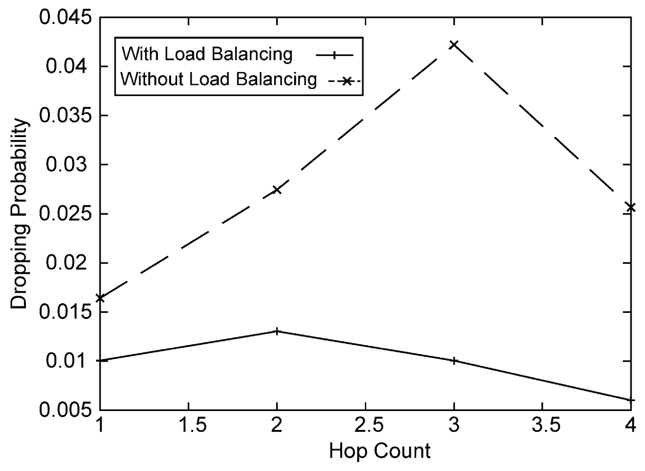

Fig. 11. Impact of load balancing on network fairness for non-identical traffic load (*s*3).

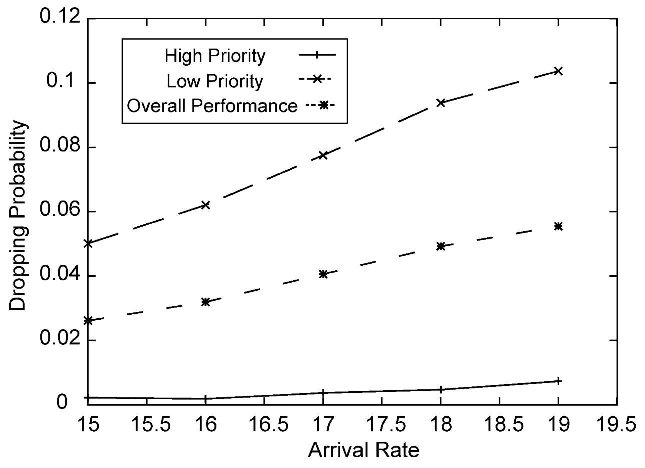

Fig. 12. Burst dropping performance for different classes without load balancing.

5.5. Inter-class service differentiation

The burst dropping performance for the high and low-priority data classes has been studied using the BFVFF algorithm with and without load balancing.

Fig. 12 shows the burst dropping probability vs arrival rate for an offset factor 2 without load balancing. It can be seen that the dropping probability for the high-priority bursts is nearly zero. This is because the high-priority bursts are given a higher initial offset. The dropping probability for the low-priority bursts is higher than that without service differentiation and the high-priority bursts.

A similar trend can be seen in Fig. 13, which shows the burst dropping probability vs arrival rate for service differentiation with load balancing for an offset factor 2. The difference between Figs. 12 and 13 lies in the fact that the overall magnitude of the burst dropping probability is lower in the case with load balancing. In both the cases the impact of service differentiation on both the classes of network traffic is quite significant.

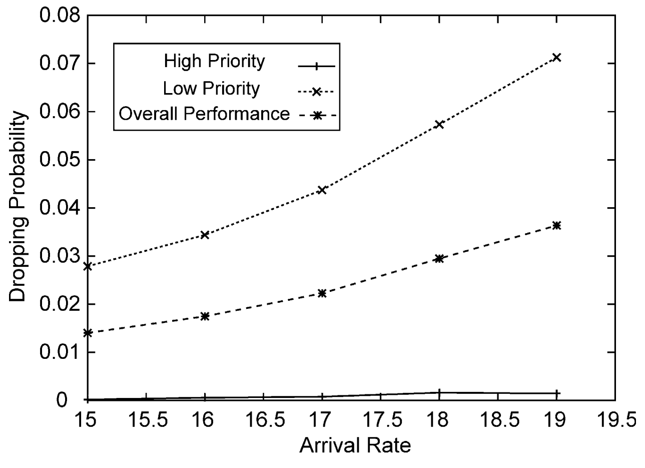

Fig. 13. Burst dropping performance for different classes with load balancing.

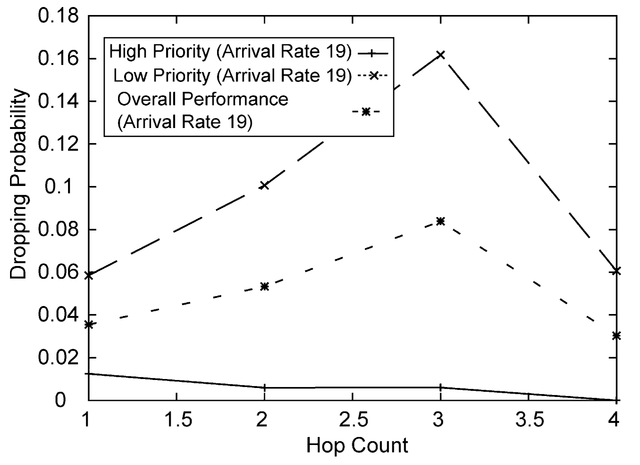

Fig. 14. Intra-class fairness without load balancing.

5.6. Intra-class fairness

The fairness among the bursts traversing different hop counts within the high and low-priority classes has been studied using the BFVFF algorithm with and without load balancing.

Fig. 14 shows the plot of burst dropping probability against hop count for an arrival rate of 19 on using service differentiation without load balancing. The offset factor used in the graph is 2. From the graph it can be seen that with service differentiation the overall burst dropping probability becomes more uniform with different hop counts for both the high and low-priority classes. Another observation that can be made from Fig. 14 is that after service differentiation the dropping probability of the higher priority service is less than that for the low-priority service for all the hop counts.

Fig. 15 studies network fairness for service differentiation with load balancing. From the graph similar trends can be seen as that for without load balancing. However, it can be observed that service differentiation with load

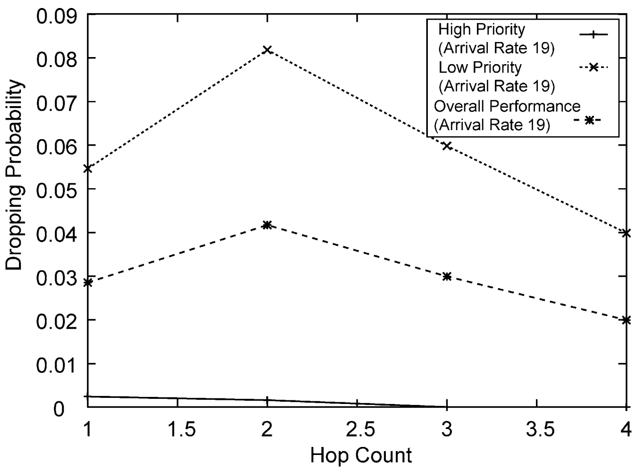

Fig. 15. Intra-class fairness with load balancing.

balancing leads to a better network performance. This is due to better resource utilization because of load balancing.

6. Conclusions

In this paper, we proposed new approaches namely, time-slotting and burst fragmentation. We also proposed a path selection scheme to distribute LSPs based on load balancing. We developed new algorithms for burst scheduling and path selection which achieve high performance with low computational complexity, while improving network fairness and providing service differentiation. The reduction in computational complexity was achieved using time-slotting. The improvement in burst dropping performance was achieved by burst fragmentation and load balancing. Further, the path selection algorithm proposed for load balancing also helps to improve fairness among bursts with different hop counts. We also proposed an offset-time based service differentiation algorithm which is not only useful to differentiate between different priority classes with the constraint of limited permissible maximum offset time, but also to improve fairness within each class. We discussed the implemenation and signaling issues of the proposed approaches. Through extensive simulation experiments we showed the effectiveness of the proposed approaches and algorithms.

References

[1] J.S. Turner, Terabit burst switching, Journal of High Speed Networks 8 (1) (1999) 3–16.

[2] C. Qiao, M. Yoo, Optical burst switching—a new paradigm for an optical Internet, Journal of High Speed Networks 1999; 69–84.

[3] L. Xu, H.G. Perros, G. Rouskas, Techniques for optical packet switching and optical burst switching, IEEE Communication Magazine January (2001) 136–142.

[4] C. Siva Ram Murthy, G. Mohan, WDM Optical Networks: Concepts, Design, and Algorithms, Prentice-Hall PTR, NJ, USA, November 2001.

[5] Y. Xiong, M. Vandenhoute, H.C. Cankaya, Control architecture in optical burst-switched WDM networks, IEEE Journal on Selected Areas in Communications October (2000) 1838–1851.

[6] M. Yoo, C. Qiao, QoS performance in IP over WDM networks, IEEE Journal on Selected Areas in Communications 18 (10) (2000) 2062–2071.

[7] Y. Chen, M. Hamdi, D.H.K. Tsang, Proportional QoS over OBS networks, in: Proceedings of IEEE Globecom 2001, 2001, pp. 1510–1514.

[8] C. Qiao, Labeled optical burst switching for IP-over-WDM integration, IEEE Communication Magazine September (2000) 104–114.

[9] H. Zhang, J.P. Zue, B. Mukherjee, Capacity allocation and contention resolution in a photonic slot routing all-optical WDM mesh network, IEEE/OSA Journal of Lightwave Technology 18 (12) (2000) 1728–1741.

[10] I. Chlamtac, V. Elek, A. Fumagalli, C. Szabo, Scalable WDM access network architecture based on photonic slot routing, IEEE/ACM Transactions on Networking February (1999) 1–9.

[11] V.M. Vokkarane, J.P. Jue, S. Sitaraman, Burst segmentation: an approach for reducing packet loss in optical burst switched networks, in: Proceedings of IEEE ICC 2002, 2002, pp. 2673–2677.

[12] K. Akash, Wavelength Channel Scheduling Using Fragmentation Approach in Optical Burst Switching Networks, Technical Project Report, Computer Communication Networks Laboratory, National University of Singapore, April 2002.

[13] M. Ashish, New Wavelength Scheduling Techniques for Improved QoS Performance in Optical Burst Switched Networks, Technical Project Report, Computer Communication Networks Laboratory, National University of Singapore, April 2002.