Journal of Engineering and Applied Sciences 6 (1): 71-78, 2011
ISSN: 1816-949X


# A Novel Scheduling Algorithm in OBS Networks

[1]Dhaneesh Chandran and [2]Janahanlal Stephen
[1]Department of Communication Engineering, IES College of Engineering, Thrissur, Kerala, India
[2]Department of Computer Science, Illahia College of Engineering, Ettumanoor, Kerala, India

**Abstract:** Optical Burst Switching (OBS) is a promising paradigm for the next-generation internet. In OBS, a key problem is to schedule bursts on wavelength channels whose bandwidth may become fragmented with the so-called void (or idle) intervals when attempting to reduce the burst loss with both fast and bandwidth efficient algorithms. Till date, only two scheduling algorithms called Horizon and LAUC-VF have been proposed which trade off bandwidth efficiency with fast running time and vice versa. In this study, we propose several novel algorithms for scheduling bursts in OBS networks with and without Fiber Delay Lines (FDLs). In networks without FDLs, the proposed Min-SV algorithm can schedule a burst successfully in O (log m) time where m is the total number of void intervals as long as there are suitable void intervals. Simulation results suggest that the algorithm achieves a loss rate which is as low as that provided previously known algorithm LAUC-VF but can run much faster. In fact, its speed can be almost the same as Horizon (which has a much higher loss rate). In networks with FDL's, the proposed Batching FDL algorithm considers a batch of FDL's simultaneously to find a suitable FDL to delay a burst which would otherwise be discarded due to contention, instead of considering the FDL's one by one. The average search time of this algorithm is therefore significantly reduced from that of the existing sequential search algorithms.

**Key words:** Optical barst switehing, schedule bursts, fragmented, horizon, networks, algorithm

## INTRODUCTION

To meet the increasing bandwidth demands and reduce costs, several optical network paradigms have been under intensive research. Of all these paradigms, optical circuit switching (e.g., wavelength routing) is relatively easy to implement but lacks efficiency to cope with the fluctuating traffic and the changing link state; Optical Packet Switching (OPS) is a natural choice but the required optical technologies such as optical buffer and optical logic are too immature for it to happen anytime soon. A new approach called Optical Burst Switching (OBS) that combines the best of optical circuit switching and optical packet switching was proposed by Yoo and Qiao (1997) and Qiao and Yoo (1999) and has received an increasing amount of attention from both academia and industry worldwide (Xiong *et al*., 2000; Xu *et al*., 2001; Detti and Listanti, 2002; Hsu *et al*., 2002).

In an OBS network, an ingress OBS node assembles data (e.g., IP packets) into (data) bursts and sends out a corresponding control packet for each burst. This control packet is delivered out-of-band and leads the burst by an offset time, o. The control packet carries among other information, the offset time at the next hop and the burst length l. At each intermediate node along the way from the ingress node to the egress node, the control packet reserves necessary resources (e.g., bandwidth on a desired output channel) for the following burst which will be disassembled at the egress node.

A prevailing reservation protocol in OBS networks is called Just-Enough-Time (JET) whereby a control packet reserves an output wavelength channel for a period of time equal to the burst length l, starting at the expected burst arrival time r (which can be determined based on the offset time value and the amount of processing time the control packet has encountered at the node up to this point in time). If the reservation is successful, the control packet adjusts the offset time for the next hop and is forwarded to the next hop. Otherwise, the burst is blocked and will be discarded if there is no Fiber Delay Lines (FDL's).

If a FDL providing say d units of delay is available for use by the burst and the channel will be available for at least l units of time starting at time r+d, the control packet will reserve both the FDL and the channel for the burst which will not be dropped at this node.

Because bursts do not arrive one right after another, the bandwidth on each channel may be fragmented with the so called void (or idle) intervals (Fig. 1). These void intervals may be utilized by a scheduling

**Corresponding Author:** Dhaneesh Chandran, Department of Communication Engineering, IES College of Engineering, Thrissur, Kerala, India

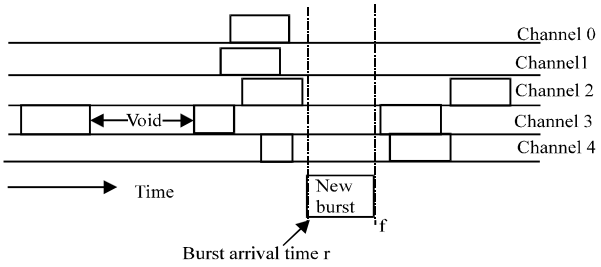


Fig. 1: Channels are fragmented into voids before scheduling a new burst

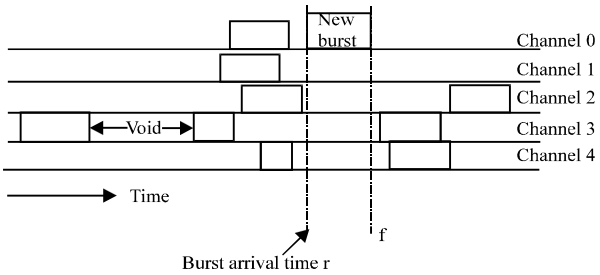


Fig. 2: Horizon schedules the new burst in Fig. 1 to channel 0

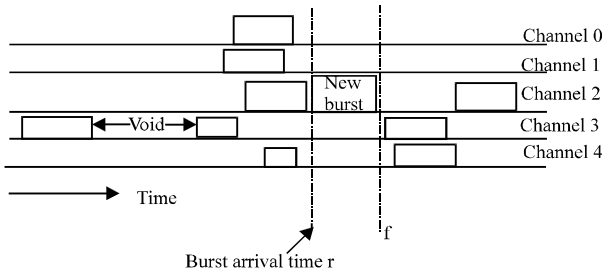


Fig. 3: LAUC-VF schedules the new burst in Fig. 1 to channel 2

algorithm to make the reservation for some bursts whose corresponding control packets arrive after the void intervals have been created (which is possible when the JET protocol is used and the bursts have a variable, non-zero offset time). However to keep the information on all existing void intervals and to search for a suitable one upon receiving a control packet (or equivalently a reservation request) could be a daunting task. In OBS networks, a key problem is thus to design efficient algorithms for scheduling bursts (or more precisely their bandwidth reservation). An ideal scheduling algorithm should be able to process a control packet fast enough before the burst arrives and yet be able to find a suitable void interval (or a suitable combination of a FDL and an void interval) for the burst as long as there exists one. Otherwise, a burst may be unnecessarily discarded either because a reservation cannot be completed before the burst arrives or simply because the scheduling algorithm is not smart enough to make the reservation.

Given the fact that OBS uses one-way reservation protocols such as JET and that a burst cannot be buffered at any intermediate node due to the lack of optical RAM (a FDL if available at all can only provide a limited delay and contention resolution capability), burst loss performance is a major concern in OBS networks. Hence, an efficient scheduling algorithm that can reduce burst loss by scheduling bursts fast and in a bandwidth efficient way is of paramount concern in OBS network design.

So far, two well known scheduling algorithms have been proposed. Horizon (Turner, 1998) does not utilize any void intervals and thus is fast but not bandwidth efficient (Fig. 2). On the other hand, LAUC-VF (Xiong *et al*., 2000) can schedule a burst as long as it is possible but has a slow running time (Fig. 3).

In this study, we propose an efficient way to organize the void intervals and as well as algorithms to schedule a burst as long as it is possible. The algorithms can schedule bursts at least as efficiently as any existing scheduling algorithms (including LAUC-VF) and can handle the case with FDL's efficiently as well. In addition, the scheduling algorithms take as little as O (log m) time where m is the total number of void intervals which improves over the LAUC-VF algorithm by k times where, k is the number of wavelengths on each link. Since, we can easily reduce the binary search problem to this channel scheduling problem, the lower bound of this problem is O (log m), meaning the algorithm is theoretically optimal.

In fact, the simulations show that on average, it can run as fast as Horizon (which has a much higher burst loss rate). Also in case, there are up to B different delays via FDL's available, the algorithm can batch process multiple possibilities and achieve an average running time which is much faster than that of existing approaches that sequentially checks one FDL at a time.

## LITERATURE REVIEW

**Problem description:** In an OBS network, it is possible that a control packet may arrive o units of time (which is called the offset time) before the corresponding burst b arrives.

In such a case, the reservation for the burst will not start at the current time (t) but at r = o+t (i.e., when the burst actually arrives). If the burst's length is l, the reservation will be made until f = r+l. Because bursts may not arrive one after another without any interval in between each channel is likely to be fragmented with

several reservation periods, separated by idle (also called void) intervals. More specifically, each of the k channels initially corresponds to a void interval from time 0 to positive infinite.

Let each void interval Ij be modeled as an ordered pair (sj, ej) where sj and ej are the starting and ending time of the void interval Ij, respectively with ej>sj. We say a void interval Ij is feasible to a data burst b = (r, f) if and only if sj r and ej f. Once the reservation is made using a feasible interval Ij up to two new void intervals may be created which are (sj, r) and (f, ej), respectively.

An efficient scheduling algorithm should be able to fit a new reservation period into an existing void interval whenever possible to increase the bandwidth utilization and decrease the data loss rate.

The availability of FDL's at each node further complicates the design of scheduling algorithms. More specifically assume that there are B different delays (d1 <d2< • • • <dB) that a burst can obtain via FD'Ls at a node. Then, the possible offset time values are o, o1, o2, • • •, oB where oj = o+dj for 1 j B. This in turns leads to B+1 different starting times r, r1, r2, • • •, rB and finishing times f, f1, f2, • • •, fB of the reservation period for the burst. An efficient scheduling algorithm thus needs to examine up to B+1 possible ways to satisfy a reservation request for each burst.

In addition to be efficient in terms of bandwidth utilization and loss rate, a scheduling algorithm also needs to be fast as mentioned earlier. Assume that the minimum burst length is 1 unit time (e.g., a millisecond). For an OBS switching fabric having N input links, each multiplexed with k channels, the maximal number of control packets (or reservation requests) that may need to be processed is kN per unit time. For N = 64, k = 100 and a unit time of 1 millisecond, this translates to a required processing speed of 6.4 million requests per second.

**Prior solutions and their limits:** Several algorithms have previously been studied for solving the channel scheduling problem. Turner designed the Horizon scheduling algorithm (Turner, 1998).

In this algorithm, a scheduler only keeps track of the so called horizon for each channel which is the time after which no reservation has been made on that channel. The scheduler assigns each arriving data burst to the channel with the latest horizon as long as it is still earlier than the arrival time of the data burst (this is to minimize the void interval between the current horizon and the starting time of the new reservation period; Fig. 2 for an example). For a link with k channels, the best implementation of the horizon scheduling algorithm takes O (log k) time to schedule a burst. Accordingly, the horizon algorithm is relatively simple and has a reasonably good performance in terms of its execution time. However, the horizon scheduling algorithm results in a low bandwidth utilization and a high loss rate.

This is due to the fact that the horizon algorithm simply discards all the void intervals. Xiong *et al.* (2000) proposed a channel scheduling algorithm, called LAUC-VF (Latest Available Unused Channel with Void Filling). LAUC-VF keeps track of all void intervals (including the interval between the horizon and positive infinity) and assigns a burst arriving at time r a large enough void interval whose starting time si is the latest but still earlier than r.

This yields a better bandwidth utilization and loss rate than the Horizon algorithm. However even the best known implementation of LAUC-VF has a much longer execution time than the Horizon scheduling algorithm, especially when the number of voids m is significantly larger than k (which in general is the case).

For example, a straightforward implementation of the LAUC-VF algorithm, described in (Xiong *et al.*, 2000) takes O (m) time to schedule a burst. The time complexity becomes O (B m) when there are B different delays a burst can obtain via the use of FDL's. Searching for a suitable void interval in this way, might take a longer time than that is allowed by the offset time of a burst, thus resulting in a failed reservation.

## PROPOSED SCHEDULING ALGORITHMS

In this study, we discuss several efficient algorithms for selecting channels for incoming data bursts using different criteria. The algorithms are based on interesting techniques from computational geometry.

**Modeling the problem geometrically:** We view each void interval Ij as a point with coordinates (sj, ej) on a 2-dimensional plane whose x and y axes are the starting and the ending time, respectively (Fig. 4). In addition, without considering the use of FDL's, the reservation period for each data burst b is mapped to a fixed point (r, f). When there is no ambiguity, we will also use Ij and b to denote the points (sj, ej) and (r, f), respectively.

Since in each void interval, the ending time is always larger than the starting time, all the void intervals are mapped to points above the 45 line y = x (Fig. 4). Also the set Fb of void intervals feasible to b lies in the unbounded region R which is to the left of the line x = r and above the line y = f (Fig. 5) (since each channel can have at most one void interval feasible to b the total number of points inside R is at most k).

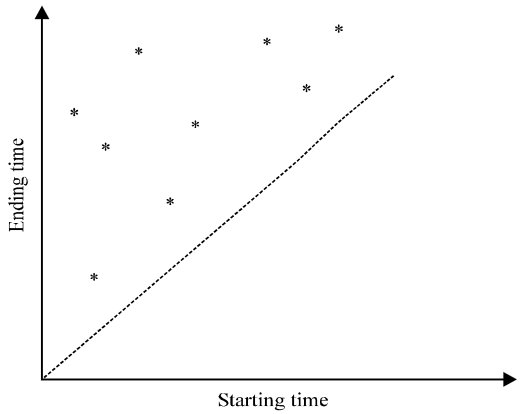


Fig. 4: Void intervals map to points

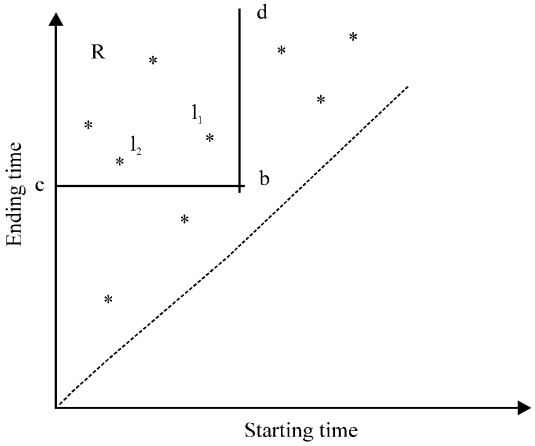


Fig. 5: Feasible region for data burst b

If there is no point inside R, it means that no channel is available to the burst b with its current offset time o. In this case, if FDL's are available, one can consider using a FDL to effectively increase the offset time by a fixed value and map the requested reservation period to a new point, say b1 in the plane. In case, there are B different delay times the burst can obtain, the desirable reservation periods correspond to a set of points b, b1, b2, ... and bB. All those points are on a straight line y = x+l (where, l is the duration of burst b). The feasible region for a data burst with the set of (B+1) offset times is bounded by a staircase curve from below (Fig. 6).

**Representative criteria for selecting a channel:** We have proposed scheduling algorithms that can apply several different criteria to select a channel for an arriving burst b. The first criterion is that for a given offset time to find a void interval Ij which minimizes the difference between r and sj among all feasible void intervals, i.e., r-sj = miniFb (r -si). We call the feasible interval meeting this criterion as the minimum starting void or Min-SV fit which aims to

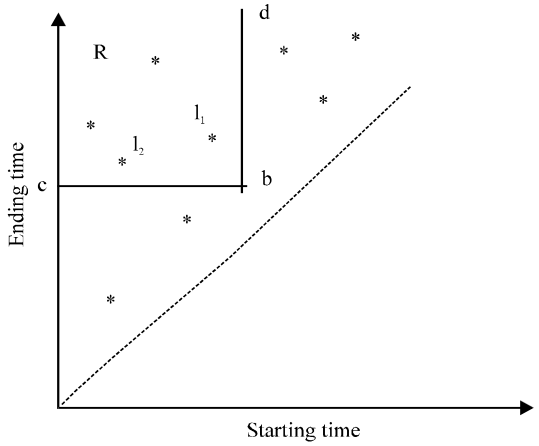


Fig. 6: Feasible region of a data burst b with multiple offset times

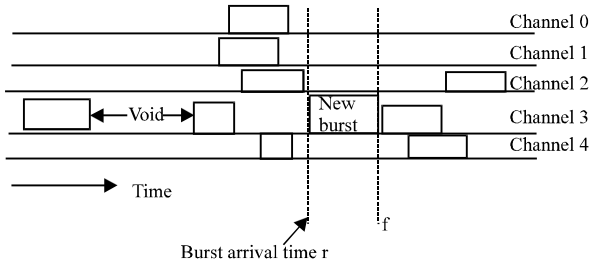


Fig. 7: Min-EV schedules the new burst in Fig. 1 to channel 3

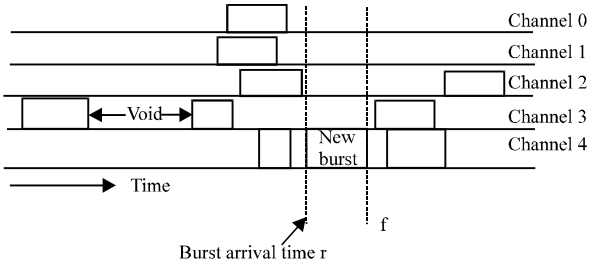


Fig. 8: Best-fit schedules the new burst in Fig. 1 to channel 4

achieve the same objective as LAUC-VF (Fig. 3). Figure 5 shows an example where I1 is the Min-SV fit for b. Similarly, we can define the minimum ending void or Min-EV fit which minimizes the difference between ej and f (Fig. 5 for an example where I2 is the Min-EV fit for b Fig. 7 for another example) as well as their opposites, Max-SV fit and Max-EV fit, respectively (Fig. 8). Another criterion is called the best fit which finds a void interval Ij which minimizes the following (over all feasible void intervals Ii) miniFb{(r-si)+(ei-f)}. The weighted best fit finds a void interval to minimize the following weighted sum miniFb{(r - si) + (1 - )(ei - f)} for 0<<1.

**Algorithm and data structure for the Min-SV fit:** Below, we will describe the method to schedule a burst by finding a Min-SV fit which is the point in the feasible region R (Fig. 2) that is closest to the vertical line x = r. This method can easily be modified to find a Min-EV fit, Max- SV fit or Max-EV fit by rotating the coordinate system and/or reversing the order of searching for points in the coordinate system.

The data burst query takes a burst b as input and outputs. The Min-SV interval if such an interval exists. The interval insertion adds a new interval into the data structure and the interval deletion removes a interval away from the data structure. To assign an available channel to a data burst b, one needs to first perform a burst query operation to find the Min-SV fit Ij then a deletion operation to remove Ij from the data structure and finally up to two insertion operations to insert up to two sub-intervals of Ij (called a starting void and an ending void, respectively) resulting from the channel assignment. Obviously, all three operations must be performed quickly.

To build the data structure DSMin-SV, we first sort the set I of all void intervals by their starting times (this is done off-line first and incrementally thereafter) and then build a balanced binary search tree Tstart based on the sorted starting times. This is accomplished by finding the interval Im (called median interval) whose starting time sm is the median of all the starting times in the considered set I of intervals. The median interval Im is then used to partition I into two approximately equal-sized smaller sets and each smaller set is partitioned recursively. All intervals in I will be the leaves of the search tree Tstart. A non-leaf (or internal) node v in the tree corresponds to a vertical strip Sv in the coordinate system. More specifically, the root of the tree corresponds to the whole plane and its two children correspond to the two half planes induced by the separating line x = sm crossing the median interval and each half plane is recursively partitioned.

Without loss of generality, the median interval is assumed to be in the left subtree. Each internal node v is associated with the following information SV m, pv ymax and pv ymin where, SV m is the median starting time among all intervals in Sv, pv ymax is a pointer to the interval in Sv with the maximum ending time and pv ymin is a pointer to the interval with the smallest ending time (the last value is used for clean-up operation to be described later). To perform a burst query on Tstart for a burst b = (r, f), we can use the information stored in the internal nodes of T start and perform the following procedure to search for the Min-SV fit.

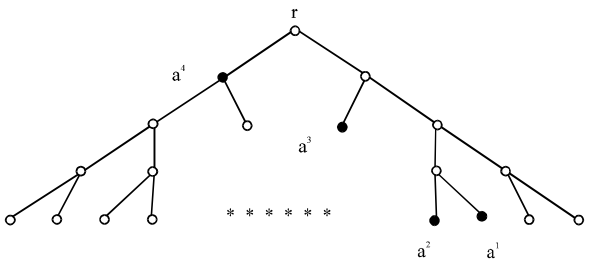


Fig. 9: Allocation nodes and search for Min-SV fit

We first search the tree Tstart to find all points to the left of the vertical line x = r (some of which may be below the feasible region). To do this, we start at the root of the tree and compare r with the median starting time associated with the root. If r is larger, we mark the left child and proceed to the right. Otherwise, we simply proceed to the left.

We call the set A of marked nodes allocation nodes which correspond to vertical regions that contain intervals or the intervals themselves whose starting times are no greater than r. The path P from the root of Tstart to a1 is called the allocation path. Figure 9 for an example where P is the dark curve and A is the set of solid nodes. All the intervals whose starting time that is ≤r are to the left of the leaf node reached in the end (i.e., $a^1$ in Fig. 4). Thus to find the Min-SV fit for b, it is sufficient to check the marked leaf node and if needed, then search in the vertical strips corresponding to the marked non-leaf nodes. Finally, one can find a random-fit by searching the region or responding to a randomly selected marked node (and then if necessary, the regions corresponding to other marked nodes in a random order).

**Scheduling algorithm for the case with FDL's:** Scheduling a data burst with only one fixed offset time (i.e., without FDLs) may fail when there is no feasible void interval. One way to alleviate this problem is to use FDL's to effectively increase the offset time (and in turn, shift the reservation period). Several approaches could be used to handle the case where there are B different delays that a burst may obtain via FDL's at a given node: Run the Min-SV fit algorithm (or its variation) to schedule the data burst, starting with the minimum possible delay, until either the reservation succeeds or fails even with the maximum possible delay. This approach takes O (B logm) for scheduling a data burst in the worst case.

Select a set of p(1<p B) different delays and schedule the burst with any one of these delays in the associated feasible regions corresponding to the p possible reservation periods. This approach as to be discussed next, essentially processes p possible delays in one batch and has a shorter worst-case (and average case ) running time than any previous approach with an appropriate value of p. The main idea of the second approach which

we call the Batching FDL algorithm is as follows. To find a void interval for an incoming data burst b, the batching approach first uses either the Min-SV fit or the Min-EV algorithm to schedule a channel for b based on the original offset time (i.e., the offset time without the use of any FDL's). If such an interval exists then it stops. Otherwise, a set D = {d1, d2, • • •, dp} of p delays are selected and a feasible void interval inside the union RD of all the feasible regions for the corresponding reservation periods is sought. To speed up the search, the scheduling algorithm no longer looks for the Min-SV fit or the Min-EV fit. Instead, it reports the first fit I1 found inside RD by the search algorithm. This first fit may not require a minimum delay (i.e., the burst may be scheduled with a smaller delay than that is required by using this first fit). As long as the delay is within an upper boun dfor a burst, introducing a non-minimum delay to the burst may lead to bad overall performance.

**Algorithm and data structure for finding the best fit:** The objective of finding a best fit (or weighted best fit) is to minimize the total length (or weighted length) of the two void intervals (called starting void and ending void and denoted by SV and EV, respectively) so as to further improve bandwidth utility and reservation success ratio. Unlike the Min-SV fit in which the search criterion minimizes a distance in one dimension of the coordinate system (e.g., the starting time), the best fit criterion considers the L1 distance (or Manhattan distance) which is the sum of a distance in the x dimension and a distance in the y dimension. Figure 10 where, b = (r, f) is the data burst and I = (s, e); Below, we show that we can solve the best fit query in O (log2m) time. Since, the weighted best fit can be solved similarly by scaling up the y dimension by a factor of (1 - )/, we will only focus on the best fit. The data structure DSB for computing the best fit for a data burst b makes use of the dynamic version of range tree (Preparata and Shamos, 1985) data s tructure. The basic idea is to construct a randomized balanced binary search tree (Martinez and Roura, 1997) tend based on the ending time of the void intervals and for each internal node v of this tree, build another balanced binary search tree Tv start based on the starting time for the intervals in the strip Sv associated with v only (i.e., not all the intervals). One can also use the starting time as the primary dimension.

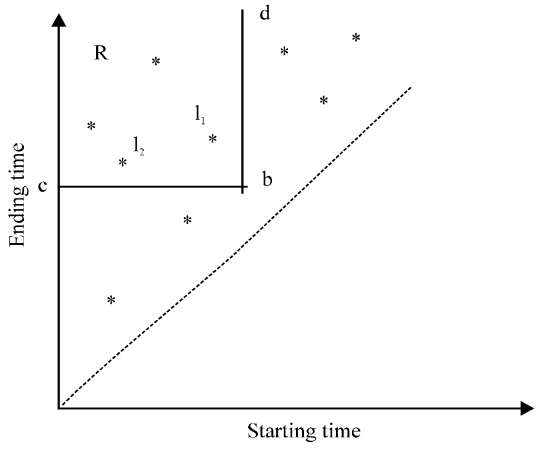


Fig. 10: Converting the bent distance to one dimension

Different from the normal range-tree, the data structure does not apply the fractional cascading technique. This is because in OBS networks, the intervals are frequently inserted and deleted from the data structure and the dynamic fractional cascading technique (Mehlhorn and Naher, 1990) although, theoretically has a factor of O (logm log logm) improvement is much more complicated and practically less efficient than the relatively simpler data structure. To facilitate the search of the best fit in this data structure in each node v of the tree tend, we store the median interval Iv ym (based on the ending time) and the minimum starting (and an optional minimum ending time) of the intervals in the strp Sv and in each node u of Tv start, we store the median interval Iu xm (based on the starting time) and a pointer pu pmin to the interval whose projection distance is the minimum among all intervals in this subtree of Tv start rooted at u. To locate the best fit of b in this data structure, we first search on the tree tend. Similar to the Min-SV fit case, we compute the set A end of allocation nodes of b in tend.

## COMPUTATIONAL COMPLEXITY ANALYSIS

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

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 Noccupied per burst. Therefore, the required processing time is constant and takes <1 NS assuming a processing speed in the order of 109 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 2 slots at the 2 ends of a

burst takes O (log M). Therefore, the overall worst case time complexity is O (1)+O (M)+O (log M) = O (M). In a normal case, however the size of heads and ends is about M/K where K is the average number of slots per burst. Hence, the average complexity is O (M/K) per matching operation.

The overall average complexity is O (1)+O (M/K)+O (log M) = O (M/K+log M). Let us consider an example with M = 256 and K = 16, leads 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 (t0, t1). The average processing time is much smaller as match () and remove (t0, t1) are only executed in heavy loading conditions. The required number of processing elements is inversely proportional to the slot size or proportional to the average number of slots per burst K. Therefore, although basic Ordered scheduling has the advantage of fast processing compared to the enhanced version, its drawback is that it requires a much larger number of processing elements to ensure good dropping performance. For enhanced Ordered scheduling, there is a tradeoff between processing speed and hardware complexity.

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

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

**Timing in Ordered scheduling:** The timing in an operationcycle of Ordered scheduling is shown in Fig. 11.

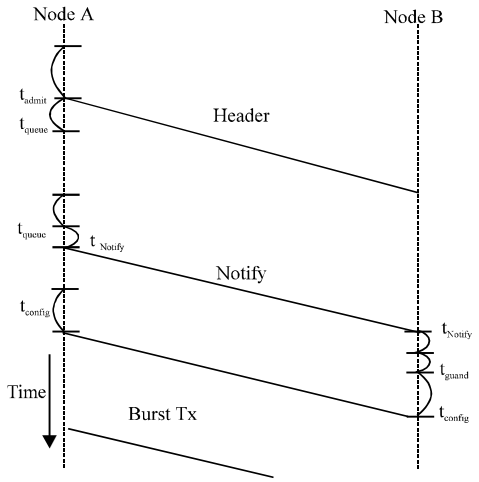


Fig. 11: Timing diagram for scheduling a burst

At the beginning of an operation when a header packet arrives at node A, the admission control test takes tadmit. If the burst reservation is successfully admitted, its header is sent to the next downstream node B while the reservation object is placed in the priority queue which takes tqueue.

At a suitable time, the object is removed from the queue which also takes tqueue since for most implementations of the priority queue, enqueue and dequeue operations take approximately the same amount of time. It takes tNOTIFY to transmit the notify packet at node A and to receive it at node B. At both nodes, the optical switching matrices are configured tconfig before the burst arrival. There is a guard time tguard between the receipt of the NOTIFY packet and the start of the optical switch configuration.

This is to ensure that timing variations in various operations will not cause the NOTIFY packet to arrive late for the optical switch configuration. One of those timing variations may result from a large number of burst arrivals in a short interval.

Without tguard, this could overload the scheduler and render it unable to make scheduling decisions in time for optical switch configuration at the downstream node. From the timing diagram, we can calculate the minimum time required to schedule a burst under Ordered scheduling .

We attempt an estimate for T0 here. From the previous section, we have tadmit = 50 NS in the worst case and tqueue = 5 NS. For tconfig according to (Martinez and Roura, 1997), the Semiconductor Optical Amplifier (SOA) technology can achieve a switching time tconfig of 1 NS or less. For tNOTIFY, assuming 8-byte NOTIFY packets as in study 3.5, it will take

tNOTIFY = 6.4 NS to transmit or receive a NOTIFY packet at 10 Gb/s. Assuming a 10% guard time, the estimate for T0 is 80 NS in the worst case. With rapid advances in electronics, we expect the figure to go down significantly in the near future.

## CONCLUSION

In this study, we have presented several novel channel scheduling algorithms (including Min-SV, Min-EV, Max-SV, Max-EV, Batching FDL and Best Fit) in OBS networks using different channel selection criteria. Unlike existing channel scheduling algorithms, such as LAUC-VF and Horizon which primarily aim at optimizing either the running time or loss rate but not both the algorithms take both performance metrics into consideration and can perform well in networks with or without FDL. Most of the algorithms have a very shorter scheduling time for each incoming burst (e.g., worst case O (logm) time) and maintain a low loss rate. We have implemented three of the algorithms, Min-SV, Min-EV and Batching FDL and conducted a comprehensive experimental study on them under different network settings (e.g., different channel number, different offered link load and different offset time range).

## REFERENCES

Detti, A. and M. Listanti, 2002. Impact of segments aggregation on TCP Reno flows in optical burst switching networks. Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, June 23-27, New York, USA., pp: 1803-1812.

Hsu, C.F., T.L. Liu and N.F. Huang, 2002. Performance analysis of deflection routing in optical burst-switching networks. Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, June 23-27, New York, USA., pp: 66-73.

Martinez, C. and S. Roura, 1997. Randomized binary search trees. Research Report of Universitat Politcnica de Catalunya, LSI-97-8-R, 1997.

Mehlhorn, K. and S. Naher, 1990. Dynamic fractional cascading. Algorithmica, 5: 215-241.

Preparata, F. and M.I. Shamos, 1985. Computational Geometry: An Introduction. Springer-Verlag, New York.

Qiao, C. and M. Yoo, 1999. Optical Burst Switching (OBS): A new paradigm for an optical internet. J. High Speed Networks, 8: 69-84.

Turner, J., 1998. Terabit burst switching progress report (9/98-12/98). Washington University at St. Louis Technical Report, 1998.

Xiong, Y., M. Vandenhoute and H. Cankaya, 2000. Control architecture in optical burst-switched WDM networks. IEEE J. Selected Areas Commun., 18: 1838-1851.

Xu, L., H.G. Perros and G. Rouskas, 2001. Techniques for optical packet switching and optical burst switching. IEEE Commun. Magaz., 39: 136-142.

Yoo, M. and C. Qiao, 1997. Just Enough Time (JET): A high speed protocol for bursty traffic in optical networks. Proceeding of IEEE/LEOS Conference on Technologies For a Global Information Infrastructure, August 1997, IEEE Xplore, pp: 26-27.