PI-OBS: a Parallel Iterative Optical Burst Scheduler for OBS Networks

P. Pavon-Mariño[1], J. Veiga-Gontan[1], A. Ortuño-Manzanera[1], W. Cerroni[2], J. Garcia-Haro[1]
[1]*Universidad Politécnica de Cartagena (UPCT),Cartagena, Spain*
{pablo.pavon, juan.veiga}@upct, aom@alu.upct.es, joang.haro@upct.es
[2]*University of Bologna, DEIS Department, Italy*
walter.cerroni@unibo.it

Abstract

This paper presents the PI-OBS algorithm, a parallel-iterative scheduler for OBS nodes. Conventional schemes are greedy in the sense that they process headers one by one. In PI-OBS, all the headers received during a given time window are jointly processed to optimize the delay and output wavelength allocation, applying void filling techniques, and allowing traffic differentiation. Results show a similar or better performance than the LAUC-VF algorithm, commonly used as a performance bound for OBS schedulers. The PI-OBS scheduler has been designed to allow parallel electronic implementation similar to the ones in VOQ schedulers, with a deterministic response time.

1. Introduction

In the Optical Burst Switching (OBS) paradigm [1], electronic traffic is assembled into variable length optical bursts, which are injected into the OBS network by network edge nodes, and transparently routed across the OBS network. The transmission of the burst payload in a fiber is preceded by a burst header packet (BHP), which is usually transmitted in a dedicated control wavelength. BHPs include control information: the burst payload transmission wavelength, the payload duration, the offset time between the burst control header and burst payload, the destination address and the class *of service.*

Fig. 1 illustrates the generic architecture of an OBS switching node with N input and output fibers, 1 control wavelength (λ_0) and n data wavelengths ($\lambda_1...\lambda_n$) per fiber. The optical switching fabric (OSF) transparently switches optical bursts from input ports to output ports. In this paper, we consider OSFs able to emulate output buffering, with full wavelength conversion (e.g. like [2] or [3]), and D fiber delay lines (FDLs) of duration d=0,G,...,(D-1)G, where G denotes the FDL granularity.

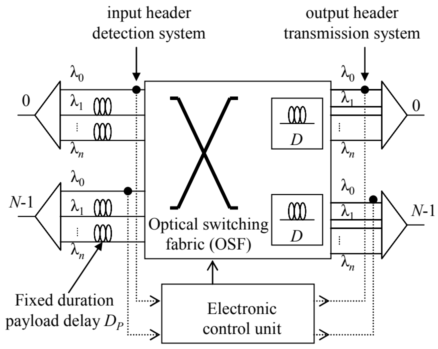
input header detection system
output header transmission system
λ0
λ1
λn
0
N-1
D
Optical switching fabric (OSF)
Fixed duration payload delay D_P
Electronic control unit

Fig.1. Scheme of an OBS node

The scheduling algorithm is responsible for allocating a contention-free FDL and output wavelength to switch every burst payload to its target output fiber. Because of the variable length of the bursts, this process may imply the creation of significant unused time gaps (*voids*) between two consecutive bursts in an output wavelength. The scheduling algorithms intended to

The work described in this paper was carried out with the support of the BONE project ("Building the Future Optical Network in Europe"); a Network of Excellence funded by the European Commission through the 7th ICT-Framework Program.

This research has been partially supported by the MEC projects TEC2007-67966-01/TCM CON-PARTE-1, and TEC2008-02552-E, and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la R. de Murcia, F. Séneca".

Pablo Pavon Marino, Juan Veiga-Gontan, Alejandro Ortuño-Manzanera and Joan Garcia Haro are with Universidad Politécnica de Cartagena (UPCT), Plaza Hospital 1, 30202, Cartagena, Spain (phone: +34 968325952; fax: +34 968325973; e-mail: {pablo.pavon, juan.veiga}@upct, aom@alu.upct.es, joang.haro@upct.es.

Walter Cerroni is with the University of Bologna, DEIS Department, Via Venezia 52, 47023 Cesena (FC), Italy (phone: +39 0547 339209; fax: +39 0547 339208; e-mail: walter.cerroni@unibo.it).

allocate shorter bursts to fill the voids in the output wavelengths are called *void filling* algorithms.

Due to the strict timings of the JET [4] signaling protocol used for delayed reservation, OBS scheduling algorithms are affected by an unavoidable response time constraint: the internal path in the OSF must be ready at the moment of payload arrival. Sequential approaches, processing input headers one by one, suffer from a worst case response time which increases with the port count. Therefore, they are not suitable for medium-to-large scale OSFs. Furthermore, sequential approaches are inherently greedy. As a general rule, it is preferred to make joint resource allocations involving more than one burst, seeking a better average performance.

This paper proposes a parallel scheduling algorithm for OBS nodes, named PI-OBS (Parallel Matching Optical Burst Scheduler). PI-OBS is suitable for a fast parallel iterative implementation, with an algorithm response time almost independent of the switch size. It applies scheduling concepts present in the iSLIP-like [5] parallel-iterative scheduling algorithms designed for Virtual Output Queuing (VOQ) architectures. PI-OBS jointly processes the burst headers arriving in a given time window, using void filling techniques. It has burst loss differentiation capabilities according to burst header information.

The rest of the paper is organized as follows. Section 2 reviews related proposals found in the open literature. The PI-OBS algorithm is described in section 3, and a comparative performance study in included in section 4. Finally, section 5 concludes the paper.

2. Related work

Many scheduling algorithms have been proposed so far in the literature, where the optimal target is to provide efficient resource utilization and minimum burst loss [6].

The very first one [7] defines a time horizon for each output wavelength, as the instant after which no burst occupies the channel. Any channel with a horizon smaller than the arrival time of the burst payload (or of one of its copies delayed by the FDL buffer, if present) is available to accommodate the incoming burst and the algorithm selects the channel with the latest horizon in order to minimize the bandwidth wasted due to voids. For this reason the same scheduling technique is also called Latest Available Unscheduled Channel (LAUC) [8].

A major improvement in terms of performance is achieved by adding void filling capabilities to the Horizon policy, resulting in the Latest Available Unused Channel with Void Filling (LAUC-VF) algorithm [8]. Already scheduled channels are now included in the search, given that they are unused for a period (i.e. void) starting before the (possibly delayed) payload arrival time and large enough to accommodate the entire incoming burst. The unscheduled channels are considered as a particular case of voids with infinite length. The algorithm selects the suitable void with the latest starting time, minimizing the gap left in front of the incoming burst so that the achieved throughput can be considered as an upper bound of the OBS node performance.

In [9], a LAUC-VF variation was proposed which considers burst QoS differentiation. After a header processing is completed, the next header to be processed is chosen giving precedence to that ones associated to higher priority traffic.

Since LAUC-VF must keep track of all the voids in the output channels, the algorithm is computationally more complex and time-consuming than LAUC. However, other variants of void filling scheduling policies, such as MinSV, MinEV, BestFit [10] or HVF [11] achieve the same loss ratios as LAUC-VF with a significantly reduced complexity [1], thanks to the use of efficient data structures and smart search techniques. The most efficient scheduler compared in [1] has a complexity of $O(\log K)$, where K is the number of scheduled bursts.

Recently, the hardware implementation of an OBS scheduler based on burst resequencing, which is able to achieve optimal scheduling in $O(1)$ complexity, has been proposed [12]. This scheduler does not process burst headers immediately as they arrive. Instead, it delays and reorders them according to the respective payload arrival time. Then, by applying a simple Horizon policy, it is possible to schedule bursts that would have required a void filling algorithm in sequential header processing. However, this scheduler is applicable to the bufferless case only, since it is able to merely exploit the voids created by different offset times and not by burst payloads delayed by FDLs. Furthermore, due to the delayed header processing, data bursts need an additional latency.

In order to implement a scalable OBS scheduler, the dependence of the algorithm execution time from the switch size should be as low as possible. This can be achieved adopting a parallel processing scheme, as the one described in a recent paper [13] which presents a specific formulation of the scheduling problem and a simulation of a viable hardware implementation with the resulting response time.

Nevertheless, *all* approaches previously described address the scheduling problem by searching fast and/or parallel algorithms, for processing *one single burst header*. However, headers are still processed sequentially, which brings two persistent drawbacks: (1) a sequential approach is greedy, (2) the system has to be dimensioned for a worst case situation, with a high number of headers to be processed in a short time period. Therefore, a completely novel different approach to a parallel scheduler is the one described in this paper.

3. PI-OBS: Algorithm description

3.1. General view and time constraints

The PI-OBS algorithm is designed as a parallel iterative algorithm, which is able to guarantee an upper bound to the response time. Let us denote this response time upper bound as T_A µs. The algorithm is executed periodically, every T_I µs. The constraint $T_I \geq T_A$ ensures that an algorithm execution starts strictly after the previous execution is finished. The algorithm execution starting at time $t=t_0$, is responsible for jointly processing the burst headers asynchronously received during the time interval $[t_0-T_I, t_0]$. We call this interval the *header arrival time window* of the algorithm execution. After algorithm execution, the scheduling decisions made for all the headers processed are stored in the system so that the correct reconfiguration of the OSF is applied when each payload arrives to the OSF.

We denote T_{WC} as the worst case time (µs) spanning between the instant of header reception, and the moment in which a path is ready for the payload. T_{WC} is the sum of three time parameters: (i) T_I, as the worst case time from header arrival to algorithm execution (corresponding to a header received just at the start of a header arrival window), (ii) the algorithm response time T_A, and (iii) the reconfiguration time T_O of the optical components of the OSF:

$$T_{WC} = T_I+T_A+T_O \qquad (1)$$

Two system parameters can be tuned to fulfill this constraint: (i) a minimum offset time δ_m between the burst header and the burst payload seen by S_E node, and (ii) an extra delay D_P added in the payload path, implemented by fixed duration FDLs in the data input ports (see Fig. 1). Note that although this approach is more commonly used in Optical Packet Switching (OPS) nodes, it is also suitable in this case. Consequently, the following must hold:

$$\delta_m + D_P \geq T_{WC} = T_I + T_A + T_O \qquad (2)$$

3.2. Scheduler architecture

Fig. 2 depicts the main building blocks of the proposed scheduler architecture. It is based on the electronic interconnection of *nNH* input modules (left hand side), and *nN* output modules (right hand side), connected by means of a crossbar interconnection for inter-module signaling.

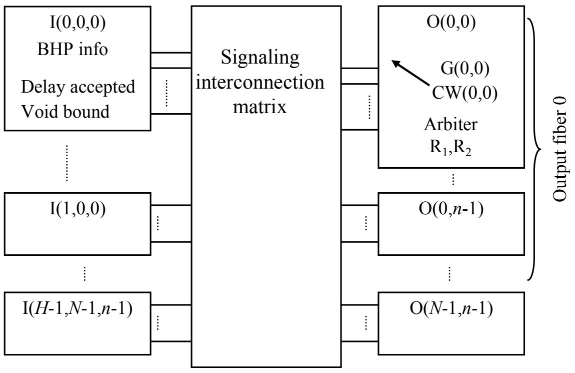

Fig.2. Scheduler architecture.

3.2.1.Input modules description. One input module I_{hfw} exists per each horizon time block (h=0,...,H-1), each input fiber (f=0,..,N-1), and each input wavelength (w=0,...,n-1). Horizons in this context are consecutive intervals of duration T_I, in which we organize the future payload arrivals. During the algorithm execution starting at time $t=t_0$, an input module I_{hfw} contains the information about a *payload* whose first bit will arrive to the OSF through input fiber f, input wavelength w, within the time interval $[t_0-T_I+T_{WC}+hT_I,\ t_0-T_I+T_{WC}+(h+1)T_I)$. The closest horizon in time corresponds to the payloads arriving to the OSF in the range $[t_0-T_I+T_{WC},\ t_0+T_{WC})$. The time $t_0-T_I+T_{WC}$ is the earliest time of arrival of a payload whose header was received at the start of the current header arrival time window, $t=t_0-T_I$. The number of time horizons to consider H, depends on the difference between the maximum and minimum offset time allowed in the system.

It must be guaranteed that at most one payload is associated to each input module in an algorithm execution. The reason is that each input module of the scheduler is able to handle at most one payload arrival. This implies that the minimum allowed payload length (L_{min}) plus the IBG must be greater than the period of the algorithm execution T_I.

$$L_{min} + \text{IBG} > T_I \qquad (3)$$

The information stored in each input module is (i) the associated burst target output fiber, burst offset, burst length and QoS class, (ii) the information about burst allocation, to be updated during the algorithm iterations: FDL and output wavelength assigned, and an upper bound to the length of the void created from burst head to preceding burst tail if the allocation took place. The nature of this void length calculation is described later in this section.

3.2.2.Output modules description. One output module O_{fw} exists for each output fiber f=0,...,N-1, and output wavelength w=0,...,n-1. Each output module contains four control registers: (i and ii) two registers storing the occupation of the output wavelength w along time: $R_1(f,w)$ and $R_2(f,w)$, (iii) a register $G(f,w)$ storing a *grant pointer*, of length $\log_2(nNH)$ bits, and (iv) one bit register $CW(f,w)$ setting the scanning direction of the grant pointer. The utilization of these registers will be made clear below.

3.3. Scheduler algorithm description

As mentioned before, one algorithm execution starts periodically every T_I µs. Let us suppose that current algorithm execution starts at time $t=t_0$. At this moment, the input modules I_{hfw}, h=0,..,H-1, f=0,...,N-1, w=0,...,n-1, contain the control information of the burst headers which arrived in the header arrival time window [t_0-T_I, t_0). In each output module, the R_1 and R_2 registers contain the same information: the occupation of the output wavelengths by the scheduled payloads in previous algorithm executions, which can still overlap with arriving payloads. R_1 will act as a backup copy of R_2 during algorithm execution.

Every algorithm execution is composed of a sequence of C_I *algorithm iterations*. Each iteration is composed of a sequence of *D delay cycles*, one for each FDL in the OSF. Each delay cycle *i* is devoted to find a wavelength allocation for the arriving payloads if they were delayed by the corresponding FDL *i*, *i*=0,...,*D*-1. Each delay cycle consists of a sequence of 4 steps: (i) request, (ii) grant, (iii) accept, and (iv) update.

The operations involved in the delay cycles of the *first iteration* of the algorithm are:

***(i) Request step (delay cycle 0,...,D-1)*:** Executed in parallel, in each of the *nNH* input modules. For input module *i* with a payload destined to output fiber *f*, a request signal is sent to the output modules associated with all output wavelengths of the target output fiber: O_{fw}, w=0,...,n-1. After the request signal, information about the payload arrival time, payload duration, and payload QoS is also transmitted through the signaling interconnection matrix.

(ii) Grant step (delay cycle 0,...,D-1). It is executed in parallel, in each of the *nN* output modules. The request signals received from the input modules are scanned, starting by the input module associated to input horizon *h*, input fiber *f* and input wavelength *w* pointed by the grant pointer $G(f,w)$ of the output module. Internally, the scanning order of the rest of (h,f,w) 3-tuples continues lexicographically: a (h_1,f_1,w_1) input module is scanned before (h_2,f_2,w_2) module if h_1 is closer to h than h_2 in the clockwise or counter-clockwise direction, depending on the state of the $CW(f,w)$ bit. If h_1=h_2, 3-tuples are ordered according to the same type of distance from f_1 and f_2 to f. If f_1 and f_2 are also equal, the input modules are ordered according to the distance from w_1 and w_2 to w. Although arduous to describe, these operations can be performed by arbiters implemented as fast combinational circuits [14]. A grant is sent to *the first* input module found whose burst does not overlap either with existing bursts scheduled in previous algorithm executions, nor allocated bursts in *previous delay cycles of the same algorithm iteration*. The arbiter gives precedence to input modules with higher QoS class, before the input module position. From the arbiter point of view, this approach is similar to the precedence between *strong* and *weak* requests in the VOQ algorithm described in [16]. The information for checking allocation overlap is stored in the R_2 register. Different arrangements of the information in R_1 and R_2 registers may lead to a trade-off between response time and electronic implementation complexity. For instance, if R_1 and R_2 registers are implemented as bit masks, each bit representing occupation during a small interval of time, the overlap check is simplified into fast and parallel bit AND operations. Furthermore, overlapping check in different delay cycles is easily performed by bit-shifting the R_2 register before the check. The grant signal (if any) is transmitted through the interconnection matrix. After it, information about the void created by this grant is also transmitted: the time distance between the head of the payload granted and the tail of the preceding payload according to current allocation. This is easily calculated as a by-product of overlapping check operation. Note that the true void can be decreased if in subsequent delay cycles a burst is allocated to fill the gap between the current granted burst head and the preceding burst tail.

(iii) Accept step (delay cycle 0,...,D-1). Executed in parallel, in each of the *nNH* input modules. From the grant signals received, the one with a smaller void is selected. If that holds for more than one grant, the one with the lowest wavelength index is preferred (first-fit). Then, an accept signal is sent to the associated output module. Note that (i) only the input modules which sent a request can receive a grant, (ii) after sending an accept signal, the input module does not enter into play for future delay cycles *in the same iteration*.

***(iv) Update step (delay cycle 0,...,D-2)*:** Executed in parallel, in each of the *nN* output modules. The internal register R_2 storing the occupation of the output wavelength along time is updated with the new accept signals information, so that future assignments in different delays of the same iteration do not overlap with the accepted allocation.

Once an iteration is finished, all the assignments performed are erased. The occupation of the R_2 register is set to be R_1 again: the system is reset to the state previous to the first iteration in this algorithm execution. The only information that remains in the scheduler from one iteration to the next, is stored at the input modules: each input module remembers the delay and void associated to the allocation accepted in the previous iteration, if any. This information will be used in the request and grant steps of the next iteration.

The actions taken during request steps in iterations 2,..., C_I are modified as follows. Let us suppose an input module (h,f,w) which accepted an allocation for delay D_1 in the previous iteration, with a void bound of V_1. In the next iteration delay cycles d=0,...,D_1-1, normal operation is performed. If an accept signal is sent, the input module refuses to send more request signals in the next delay cycles. During delay cycle D_1 (if the module has not received a grant yet), the request signal is accompanied with the size V_1 of the void bound information stored at the input module, which is sent to the requested output modules.

The grant step is modified as follows. Each output module grants the first input module following the scanning order, for which (i) a request is received, and (ii) the void generated by this allocation (checked from the R_2 register) is strictly lower than the void V_1 published by that input module. Again, this functionality can be implemented by fast binary comparisons performed in parallel in all the output modules. Note that the void comparison implies that grants sent in iteration i to an input module, could be sent to other modules in iteration i+1.

At the end of the last iteration, the allocations accepted by the input modules are considered final. In the output modules, the R_2 register in each output module contains the updated occupation. This information is copied into R_1. Before next algorithm execution starts, R_I registers are modified to reflect a packet propagation of T_I µs. If R_1 and R_2 are bit mask registers, this can be performed by fast bit shift operations.

3.4. Grant pointers operation and system initialization

As it happens in the iSLIP scheduler for VOQ switches [5], the operation of grant pointers strongly affects the performance of the system. If an input module enters into a request step, it simultaneously sends a request signal to n output modules, one per each output wavelength of the target output fiber. It is of interest to reduce the number of simultaneous grants an input module receives, as at most one grant can be accepted. The non-accepted grants correspond to delay assignments not granted to other modules. Those candidate allocations will not enter into play until next iteration. Therefore, the grant pointers of output modules corresponding to the same output fiber should be *desynchronized*, in the sense that they point to input modules as separated as possible one from the other in the lexicographical ordering. Then, we increase the chances that the grants are more uniformly spread among the input modules. Similarly to algorithm [16] for VOQ switches, and to algorithm [17] for slotted OPS switches, this can be obtained by: (i) a grant pointer initialization during system start-up which maximizes the minimum lexicographical distance between pointers, (ii) the *CW* bit is changed after every algorithm execution, switching the scanning direction of the input ports pointed by the grant pointers. This action aims to improve system fairness when packet arrivals are not uniform across input fibers, in the same way as in [17]. (iii) Every two algorithm executions (with opposite scanning directions) all the pointers increase the value by one, modulo nNH.

3.5. Algorithm convergence

We define algorithm convergence time as the number of iterations needed for the system to achieve a *stable allocation* in all the processed bursts, which would not change if more iterations were performed.

Property 1: Worst case convergence time is limited by a finite bound.

Proof: Let I_{hfw} be an input module which has received a grant for delay D_1 in iteration i. In iteration i+1, the same input module can receive a different delay-wavelength assignment, only when (i) the allocations in delay cycles 0,.., D_1-1 have changed from previous iteration, *or* (ii) the void bounds V_1 announced in delay cycle D_1 change in any input module.

For delay cycle 0, only condition (ii) can hold, and a variation in (ii) can only occur when an input module has improved its void estimation. Therefore, this can only happen during a finite number of iterations. After that, allocations do not change in delay cycle 0. At this moment, applying the same principle to delay cycle 1, then 2, etc. convergence is guaranteed in a finite number of iterations.

The PI-OBS algorithm addresses the multi-objective optimization problem of allocating delays and output wavelengths to arriving bursts so that: (i) the number of bursts receiving a delay is maximized for each QoS class, prioritizing higher class traffic, (ii) the average delay of the allocation is minimized, (iii) the average size of the voids generated is minimized.

Property 2: The PI-OBS stable allocations are distance-1 local optimum solutions to the previous problem.

Proof: We provide an intuitive proof. A distance-1 local minimum solution means that the allocation is not improved by neighboring solutions which differ in at most one assignment. Let us assume a solution in which (i) one more burst could receive a delay-wavelength instead of being dropped, or (ii) receive a better delay, (iii) or receive an assignment with the same delay but implying a smaller void. Clearly, this solution would not be a stable allocation, convergence has not been reached, and the algorithm would change the solution in a further iteration.

4. Results

In the testing scenario, the switch under evaluation S_E receives traffic from N input neighbor nodes (I_0,...,I_{N-1}), and is responsible for switching it to N output target nodes (T_0,...,T_{N-1}). Connecting fibers have n data wavelengths λ_1,...,λ_n and one separated control wavelength λ_0.

Three different scheduling algorithms will be evaluated in the S_E node: LAUC, LAUC-VF and PI-OBS. LAUC and LAUC-VF are sequential algorithms which are commonly used as performance bound in comparisons.

The reconfiguration time of the optical equipment and the IBG time are assumed to be equal to 0.03 μs (T_O=IBG=0.03 μs). Both the input nodes and the S_E node under test respect this IBG time in their assignments. Scheduling algorithms can easily do that by artificially adding the value IBG to the payload duration. Then, the scheduler guarantees that every payload is followed by an idle time of IBG μs in the output wavelength.

Each source node assembles bursts of payload duration given by a truncated normal distribution. Minimum burst length is set to 10 μs, and the maximum burst length to 100 μs. Average burst length is set to 55 μs. The time between the assembling of two bursts is exponentially distributed. Its average is calculated to match the desired load value. After a burst is assembled, the transmission wavelength and injection time in the connecting fiber are selected as if the input node was a LAUC-VF node, with an infinite number of FDLs. Using LAUC-VF source nodes is considered a more realistic scenario, which intends to reproduce the correlations in burst arrivals that appear in different wavelengths of the same fiber in an OBS network. The granularity of the FDLs in the source nodes and the S_E node is made equal to 55.03 μs, the average burst length plus the IBG time. We denote this as the *perceived average burst length*, as it is the average burst length observed by the scheduler. Previous works have shown that a FDL granularity close to the average payload duration optimizes system performance [18].

The time between two algorithm executions is set to T_I=10 μs. The algorithm response time is assumed to be also T_A=10 μs (so that the constraint $T_I \geq T_A$ is tight). The minimum offset time of the bursts generated by the edge node is calculated by assuming that the switch under test has D_P=0 μs extra payload delay. Then δ_m=20.03 μs. Bursts are generated with a random offset uniformly distributed in the range [20.03, 80.03] μs. This implies that 7 horizons of 10 μs each have to be used in the PI-OBS algorithm. For the algorithm execution starting at time t=t_0, the first horizon contains the payloads arriving to the OSF at time interval [t_0+10.03, t_0+20.03], and last horizon for payloads arriving at the time interval [t_0+70.03, t_0+80.03]. Note that in OBS networks designed to have a constant offset time, the number of PI-OBS horizons would be reduced to 1, resulting in a relevant saving in implementation complexity.

Target output node of the bursts is selected randomly with uniform distribution. Two classes of service have been defined in all the tests: 10% of the bursts are of high priority traffic (*Hi*), and 90% of low priority or best-effort traffic (*BE*). Only the PI-OBS algorithm performs traffic differentiation.

Performance evaluation has been conducted by means of discrete event simulation. The simulation tool has been built on top of the OMNeT++ platform [19]. All the tests performed consist of 5 independent samples, with 10^7 generated bursts each. Confidence intervals are calculated for a 95% quality, using the *t*-Student method. Confidence intervals obtained validate the results, but are not shown in the figures for the sake of clarity.

The first step in our study addresses the dimensioning of the buffering in the S_E switch, required for guaranteeing an average *bit loss probability* below 10^{-5} for an 80% input load. Measuring the bit loss probability means that the loss of a burst is weighted by its duration. The variance of the payload duration is made equal to the perceived updated average burst length 55.03 μs. This corresponds to a coefficient of variation of the payload distribution equal to 1, *CV*=1.

Results are shown in Table I. Note that PI-OBS and LAUC-VF algorithms have a similar buffering performance. In the DWDM scenario (n=64), 2 FDLs are enough to guarantee the loss target. Also, results confirm that LAUC algorithm strongly increases the buffering requirements, because of its inefficient use of resources. The same buffering requirements, not shown in the table, have been obtained for CV=0.5 and CV=1.5, with the only difference for the case n=16, CV=0.5, where both LAUC-VF and PI-OBS schedulers required one extra FDL.

Table I
FDL buffering {PI-OBS/LAUC-VF/LAUC}

N \ λ	n= 16	n = 32	n = 64
N=2	4 / 4 / >10	3 / 2 / >10	2 / 2 / 4
N=4	4 / 4 / >10	3 / 3 / >10	2 / 2 / 4
N=8	4 / 4 / >10	3 / 3 / >10	2 / 2 / 4

The subsequent tests included in this paper intend to provide a deeper understanding of how LAUC-VF and PI-OBS schedulers react during overload intervals. The LAUC scheduler is removed from the picture because of its well-known worse performance.

Fig. 3 and Fig. 4 show the burst loss probability (BLP) of a switch with N=4 input and output fibers, and n=16 wavelengths per fiber, under an input load of 95%, but with the buffering dimensioning of D=4 FDLs (calculated in Table I for a bit loss probability of 10^{-5} under an 80% load). Naturally, that overload scenario is supposed to be transient in an actual system, as an unacceptable loss probability is obtained. We have conducted similar tests for switch sizes N={2,4,8}, n={16,32,64}, not shown in the paper, which yield to the same conclusions as the ones exposed below.

Fig. 3 shows the burst loss probability distribution, depending on the payload size. Results yield to the following conclusions: (i) short bursts have a better chance to be allocated resources. All bursts larger than approximately one half of the average payload length are treated equally by the system. These effects appear in both LAUC-VF and PI-OBS algorithms. (ii) PI-OBS effectively differentiates the burst losses for the two tested service classes: high class (*Hi*), and best effort (*BE*). Nevertheless, there is room for investigating scheduler variants that obtain a stronger traffic differentiation (iii) the losses of the best-effort traffic in PI-OBS algorithm are similar to the ones in the LAUC-VF algorithm. The performance of high class traffic in PI-OBS scheduler improves LAUC-VF results.

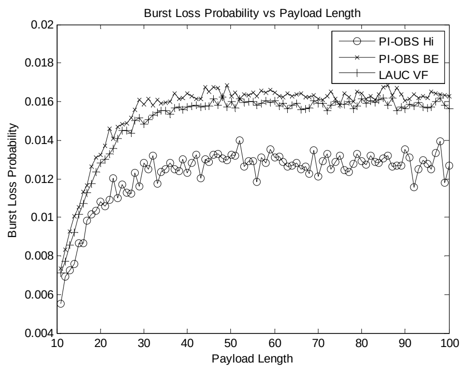

Fig. 3. BLP distribution (payload length).

Fig. 4 displays the burst loss probability distribution as a function of the offset of the burst. Again, LAUC-VF and PI-OBS are similar in the impact of the burst offset on the loss probability for the overloaded scenario, increasing the dropping probability for bursts which were announced with a shorter offset.

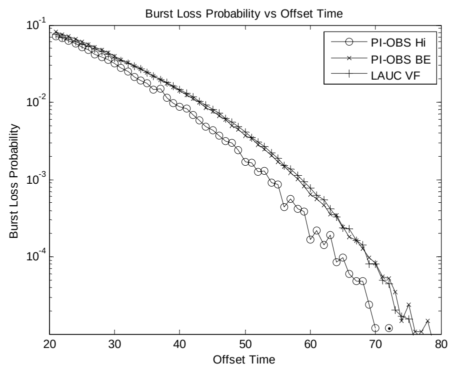

Fig. 4. BLP distribution (offset time).

The observed LAUC-VF results are comparable to the ones obtained in other works which have studied that scheduler performance, although under different testing scenarios (e.g. see [19]). The most relevant conclusions from our study are that PI-OBS, which is designed to minimize the same objective function without a greedy approach, results in a similar behavior under overloading conditions.

While PI-OBS convergence has been proved in a finite number of iterations, it is clear that shorter convergence times may lead to hardware simplification. Table II summarizes the algorithm convergence information for PI-OBS in the same tests shown in Table I. For each N={2,4,8}, n={16,32,64}, we include the number of iterations in the algorithm such that an optimal solution is obtained in the 99% of the time slots, for three CV factors, CV={0.5, 1, 1.5}. We can conclude that the convergence time does not depend either on the number of fibers, nor the *CV* factor in payload length distribution, but is slightly larger for a larger number of wavelengths per fiber. Nevertheless, the number of iterations for convergence can be considered reasonably low.

Table II
PI-OBS Convergence

λ / N	n= 16	n = 32	n = 64
N =2	6 / 6 / 6	8 / 8 / 8	10 / 10 / 10
N =4	6 / 6 / 6	8 / 8 / 8	11 / 11 / 11
N =8	6 / 6 / 6	8 / 9 / 8	11 / 11 / 11

5. Conclusions and future work

As far as the authors know, the PI-OBS is the first proposal of a parallel-iterative scheduler for OBS switches. In contrast to conventional greedy approaches, all the headers received in a given time window are jointly processed. This opens a field for a performance gain, when compared to greedy approaches like the LAUC-VF scheme. Also, algorithm convergence studies show a response time approximately independent from switch size. Observing the similarities with VOQ schedulers, authors are working on a practical parallel implementation of the scheduler, exploring the trade-off between implementation complexity and algorithm response time.

Authors consider the PI-OBS as a first step. Variations of PI-OBS can explore other strategies for joint resource allocations, yielding to performance improvements and/or hardware simplification.

7. References

[1] Y. Chen, C. Qiao, X. Yu, “Optical Burst Switching: A New Area in Optical Networking Research”, *IEEE Network*, vol. 18, no. 3, May/June 2004.

[2] C. Guillemot, et al., "Transparent optical packet switching: the European ACTS KEOPS project approach", *IEEE J. Lightwave Technol.*, vol. 16, no. 12, pp. 2117-2134, Dec. 1998.

[3] S. L. Danielsen, C. Joergensen, B. Mikkelsen and K. Stubkjaer, "Analysis of a WDM packet switch with improved performance under bursty traffic conditions due to tunable wavelength converters", *IEEE Journal of Lightwave Technology*, vol. 16, no. 5, pp. 729-735, May 1998.

[4] M. Yoo and C. Qiao, “Just-enough-time (JET): A High Speed Protocol for Bursty Traffic in Optical Networks,” *IEEE/LEOS Tech. Global Info. Infra.*, pp. 26–27, Aug. 1997.

[5] N. McKeown, “iSLIP: A Scheduling Algorithm for Input-Queued Switches”, *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188-201, April 1999.

[6] F. Callegati, W. Cerroni, G. S. Pavani, “Key Parameters for Contention Resolution in Multi-Fiber Optical Burst/Packet Switching Nodes” (Invited Paper), *Proc. of IEEE Broadnets 2007*, Raleigh, NC, Sept. 2007.

[7] J. Turner, “Terabit Burst Switching”, *Journal of High-Speed Networks*, vol. 8, no. 1, 1999.

[8] Y. Xiong, M. Vandenhoute, H. C. Cankaya, “Control Architecture in Optical Burst-Switched WDM Networks”, *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Oct. 2000.

[9] M. Yang, S.Q. Zheng, Dominique Verchere, “A QoS Supporting Scheduling Algorithm for Optical Burst Switching DWDM Networks”, in *Proc. IEEE Globecom 2001*, vol , pp.86-91.

[10] J. Xu, C. Qiao, J. Li, 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, Nov. 2004.

[11] G. Muretto, C. Raffaelli, P. Zaffoni, “Effective implementation of void filling in OBS networks with service differentiation”, *Proc. of WOBS 2004*, San José, CA, USA, Oct. 2004.

[12] Y. Chen, J. Turner, P.F. Mo, “Optimal Burst Scheduling in Optical Burst Switched Networks”, *IEEE/OSA Journal of Lightwave Technology*, vol. 25, no. 8, Aug. 2007.

[13] F. Callegati, A. Campi, W. Cerroni, “A cost-effective approach to optical packet/burst scheduling”, *Proc. of IEEE ICC 2007*, Glasgow, UK, June 2007.

[14] C. K. Hung, et al. “Design and implementation of high-speed arbiter for large-scale VOQ crossbar switches”, *Proc. of ISCAS’03 Conference,* 2003.

[15] S. Mneimneh. “Matching From the First Iteration: An Iterative Switching Algorithm for an Input Queued Switch”, *IEEE/ACM Transactions on Networking*, vol.16, no. 1, Feb.2008.

[16] Y. Jiang and M. Hamdi, “A fully Desynchronized Round-Robin Matching Scheduler for a VOQ Packet Switch Architecture”, *IEEE Workshop on High Performance Switching and Routing*, Dallas, pp. 407-411, 2001.

[17] P. Pavon-Marino, J. Garcia-Haro, A. Jajszczyk, “Parallel Desynchronized Block Matching: A Feasible Scheduling Algorithm for the Input-Buffered Wavelength-Routed Switch”, *Computer Networks*, vol. 51, no. 15, pp. 4270-4283, Oct. 2007.

[18] A. Rostami, A. Wolisz, “Impact of Edge Traffic Aggregation on the Performance of FDL-Assisted Optical Core Switching Nodes”, *IEEE International Conference on Communications, 2007*, June 2007.

[19] http://www.omnetpp.org. Last accessed 29th Jan. 2009.

[20] J. L. Garcıa-Dorado, J. E. Lopez de Vergara, J. Aracil, “Analysis of OBS burst scheduling algorithms with stochastic burst size and offset time”, *Proc. 10th Conf. Optical Networks Design and Modelling*, 2006.