Feedback Based Load Balancing, Deflection Routing and Admission Control in OBS Networks

Sébastien Rumley, Christian Gaumier
Laboratoire de Télécommunications (TCOM), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Oscar Pedrola, Josep Solé Pareta
CCABA, Universitat Politècnica de Catalunya, Spain

***Abstract*—The Optical Burst Switching (OBS) paradigm allows statistical multiplexing directly at the optical layer. Thus, OBS networks are suited to carry traffic demands varying in either the short or long term. Due to the lack of buffering, burst contention due to short term variations can only be mitigated through deflection routing. For longer term variations, higher order mechanisms such as dynamic flow-balancing or flow shaping are generally proposed. In this paper, a unified scheme, based on a feedback mechanism combined with deflection routing and admission control is introduced to handle all types of traffic variations. The use of only one single scheme simplifies the architecture of OBS networks and enhances its flexibility. The validity of our technique is supported by simulation results.**

***Index Terms*—Optical burst switching, deflection routing, feedback mechanism, load balancing, admission control.**

I. Introduction

Optical networks already carry the vast majority of the long-distance communications. Their role is nevertheless promised to be even greater with the advent of ultra-high bandwidth home accesses. However, while conceived in the past to carry constant or at least stationary traffic, they have nowadays to deal with an increasing part of traffic demands varying both in the short and long term. For instance, in digital TV long term variation are due to the users switching on or off the TV whereas short term variations are caused by compression algorithms. Future optical networks are required to cope with these significant traffic variations without loss of quality.

Optical Burst Switched (OBS) networks, given their ability to achieve statistical multiplexing directly at the optical layer, are considered as a promising approach to efficiently satisfy future communication demands. Unlike Optical Circuit Switched Networks (OCS), they offer sub-wavelength bandwidth granularity. They remain, however, simpler than their Optical Packet Switched (OPS) counterpart, whose implementation is still questionable [1].

In the basic OBS scheme, traffic is aggregated in large packets called bursts. Before sending a burst in the network, an associated control packet is sent in advance. The control packet reserves the resources required to dispatch the burst toward destination. Emitted on a dedicated channel of reduced throughput, control packets are easy to decode, unlike OPS headers. Sent in advance, they let enough time to core nodes to prepare for the burst arrival. In this way, bursts travel transparently in the network, at very high rates [2].

This conventional OBS scheme presents nevertheless several major drawbacks. Due to its bufferless nature, it cannot solve transient congestions by shortly delaying some of the contending burst, as in a classical packet switching paradigm. This weakness causes unavoidable burst losses as soon as traffic is not determinist. Furthermore, the pre-allocation of resources leads to low channel utilisation ratios.

Various approaches have been investigated and combined to both reduce as much as possible the burst loss rate and to maximise the throughput. They can be distinguished in two classes. Within the first class, local node resources only are used to solve contentions. These resources can be optical buffers [3] or complex schedulers [4]. Local approaches are beyond the scope of this article, and will not be further discussed.

Within the second class, approaches include mechanisms which encompass the whole network. Two of these methods – Deflection Routing (*DR*) and Load-Balancing (*LB*) – rely on the frequent existence of several routes between a pair of nodes. Thus, if one route is saturated, traffic can be deflected over an alternate path. However, *DR* and *LB* differ in the scale at which they operate. Specifically, *DR* reroutes on a *per burst* and *per hop* basis (one individual burst is rerouted over one single hop) whereas *LB* achieves a *per flow* and *per path* rerouting (all the bursts of one flow are rerouted over a whole path).

The *per burst* aspect of the *DR* permits to individually deflect bursts to an alternate route, which is worthy when casual bursts fail to obtain a reservation on a given channel, due to short term variations [5]. Unfortunately, when facing long term traffic variations, which cause bursts to be systematically contended at one output port, DR might simply transpose the congestion to another link rather than solve it [6]. In this latter case, the traffic should instead be routed differently in the network, to avoid systematic congestions. This is exactly what Load-

Balancing (*LB*) achieves, by limiting the number of certain bursts injected into each particular route. LB can be done statically or dynamically. In the static case, traffic is balanced according to an extrapolation based forecast. In the dynamical case, flows are repetitively estimated on the fly, and the load-balancing is achieved according to these successive estimations.

The problem with *LB* is that too tight route restrictions may favour transient contentions. Hence, if a burst cannot be forwarded to the next hop of its predefined route, it must be dropped. It results that *LB* and *DR* have to be combined to mitigate both short term and long term congestions. Unfortunately, however, their simultaneous usage may lead to problematic situations: whilst *BL* tries to restrict the routes to particular ones, *DR* attempts to find alternative paths. Synchronizing both mechanisms might be particularly difficult, since they often differ in the way that they are implemented. While deflection mechanisms are intrinsically related to core nodes, load-balancing ones are more likely located at edge nodes, and sometime even operate at a different layer.

To avoid the previously discussed problems while keeping the benefits of both mechanisms, a unified technique called *Adaptive Deflection Routing* (*ADR*) is proposed in this article. This scheme, operating at core nodes, routes the traffic dynamically, selecting the output port which ensures, for the rest of the path, the best quality for current traffic conditions. If the preferred output port is congested, deflection is performed towards the second most efficient output port, and so forth.

Moreover, the transmission quality can be further improved by limiting the access of the bursts which will or are likely to be dropped anyway. This mechanism is referred to as Admission Control (*AC*) or congestion avoidance. The *ADR* scheme presented in this paper also integrates *AC* capabilities. Thus, a burst might be dropped rather than forwarded, even if the local port is not congested, if the risk of latter contention is high.

ADR estimates the risk and quality associated to each forwarding operation using feedback messages exchanged between core nodes. When a node voluntarily drops a burst or fails to reserve a channel, a negative feedback is sent to all core nodes previously visited by the burst. Similarly, positive feedbacks are sent when a burst reaches its final destination. In this way, a core node receiving a feedback knows *a posteriori* the consequences of one of its past choices.

In Section 2, the principles on which the *ADR* scheme relies are discussed with respect to other approaches proposed in the literature. Implementation details are given in Section 3. The behaviour of the scheme is analysed through simulations in Section 4. Section 5 provides some conclusions.

II. MODELLING OF *ADR*

ADR consists of four components: Deflection Routing, Load-Balancing, Admission Control and Feedback Based Adaptation. Each of them has already been largely studied in the past. The present section reviews them briefly.

A. Deflection Routing in OBS

Deflection Routing can be superposed to any other routing scheme. If a burst can be forwarded to the output port (or one of the output ports) defined by the primary routing scheme, no deflection occurs. If on the contrary all the primary ports are busy, the burst is deflected to an alternative port.

The selection of the alternative port can be achieved in various ways. In the most simple deflection scheme, an alternative port is selected randomly or according to an arbitrary order among the idle ones. In more complex cases, a list of alternative ports is assigned to each primary port or, better, to each burst final destination.

Lists permit either fixing an order for alternatives, either limiting the number of potential deflection ports, or both. The criterion for ordering and limitation is generally the distance separating the next node of the output port from the destination. Hence, deflections implicitly lengthening the burst journey are both less likely to be selected (due to ordering) and more likely to be excluded (by limitations).

The exclusion or selection of an output port highly depends on the burst remaining offset time t_{rem} (or, equivalently, on the remaining offset time unit $u_{rem}=t_{rem}/t_p$, where t_p is the burst header processing time). Indeed, each deflection operation is equivalent to a switching operation and consumes one unit. Thus, a burst deflected too many times might be dropped owing to an insufficient t_{rem} (insufficient offset time problem [20]). Avoiding deflecting bursts on routes incompatible with the remaining offset will lead to better performances. Lists may thus be setup for each potential final destination and possible value of u_{rem}. Besides, additional units of offset times can be granted to bursts at emission. These bonus units allow a burst to be deflected more times, and may help a particular burst to find its way in the network. However, this will increase the resources consumed by each burst, which can be counterproductive [6]

It is worth pointing out that the use of ordered lists prioritizing particular deflection ports implicitly affects the way traffic flows in the network. Thus, a type of local load balancing can be achieved by this means. This load-balancing can even be achieved dynamically if core nodes exchange feedbacks with their neighbours [7].

In our *ADR* scheme, deflection is achieved according to lists recomputed after each feedback reception. However, contrarily to [7], these feedbacks are not issued by neighbouring nodes only, but by all nodes traversed by a traffic flow.

B. Routing and Load-Balancing in OBS

Considerable work has been achieved to study how to optimally balance burst flows in a network. A survey of the basic routing schemes is available in [8].

The literature on *LB* can be separated in two groups, depending on the question: is an *a priori* knowledge of the traffic pattern assumed? If the answer is yes, load-balancing can be achieved using optimisation techniques. Several models have been developed to evaluate the global performance of an OBS network, depending on its topology, traffic matrix, and core node architecture. These models can be derived to perform linear [9] or non-linear optimisation [10]. In many situations, however, *a priori* knowledge of the traffic pattern is unavailable. This is generally true when traffic fluctuates too much (any estimate becomes either obsolete very quickly, or is too averaged to represent the real traffic). These situations call for dynamic flow balancing (DFB).

DFB in OBS can be achieved in a similar way that traffic engineering is performed in (G)MPLS networks. Information of the congestion in the network is given to a client of the OBS layer, by means of a flooding protocol dispatching link state information messages (e.g. OSPF). Based on this information, the client layer selects itself the routes of its bursts, avoiding congested links [11]. This solution is, however, problematic when link state information is obsolete. Therefore, network state information must be available prior to proceed to any further change (in particular, after a previous rerouting operation).

Rather than letting the client layer manage the routing, this last can be operated by the OBS layer itself. This approach permits taking into account OBS specific congestion metrics. It also allows a decentralisation of the routing decisions. Hence, individual edge nodes [12]-[15] or core nodes [7],[16]-[19] can decide independently on which route (for edge nodes) or on which output port (for core nodes) bursts must be sent.

Edge-decision based schemes generally take into account the whole network status, allowing optimal routing. However, as in the MPLS scheme, the time required to collect network state may be problematic. Core-decision based schemes, on the contrary, base their decisions on the state of a limited part of the network (e.g. their immediate neighbourhood). This provides shorter reaction times after traffic changes, since up-to-date information is available more rapidly. However, a local view is highly likely to lead to suboptimal configurations [13].

In both cases, individual nodes may react independently but simultaneously, which will produce oscillations in the congestion [12]. More generally, when load-balancing is applied, a trade-off must inevitably be found between very reactive decisions causing oscillations and less reactive ones, keeping the network in a suboptimal state for a longer time.

In our *ADR* approach, routing decisions, similarly to deflection, are taken by core nodes according to received feedbacks. However, these feedbacks are originated in all other nodes, conferring then a global nature to the *ADR* routing scheme.

C. Feedback mechanism

Relevant information exchanged between nodes can be referred to as *per burst* or *per link*.

In the *per burst* approach, core nodes simply send feedback each time a burst is either dropped, received, or switched [13]-[15],[18]. The duty of analysing the feedbacks and deducing the congestion state is left to the receiving node. In the *per link* perspective, each core node estimates the state of each of its own output links. It then broadcasts this information [7],[12],[19],[20].

The *per burst* approach permits reporting critical situations such as repeated burst losses, almost immediately. On the contrary, the *per link* type techniques average the link state over time. Sudden changes may thus need longer time to be detected. On the other hand, the *per burst* approach is likely to generate more control overhead to dispatch correctly all the feedbacks. Nevertheless, since a signalling channel is required in OBS anyway, the impact of this additional overhead is expected to be moderate [13].

The *ADR* feedback mechanism is based on a *per burst* paradigm. The feedback only consists of either ACK or NACK messages, associated with a burst identifier. However, contrarily to other *per burst* approaches, *ADR* feedback is sent to all previously visited nodes, and not only to the burst emitting one.

D. Admission Control

While in connection oriented networks, rejected traffic has no impact on the accepted one, this is not the case in datagram-based networks. Prior to be dropped, packets may consume and thus waste network resources. This may dramatically reduce the network total throughput [6]. Mimicking the connection oriented networks, mechanisms voluntarily dropping a part of the incoming datagram at network entrance have been proposed. This protection technique is usually referred to as Admission Control (*AC*).

AC in OBS can be achieved at either edge or core nodes. The dichotomy between *per path*, when AC is achieved at edges, and *per hop*, at cores, appears hence again. *AC*-equipped edge nodes shape the traffic they inject in the core network. This shaping operation can be performed either by buffering the traffic exceeding a given rate [21], or by dropping it. The targeted maximal output rate can be fixed by static planning. It may also be estimated dynamically, according to feedbacks received from the network, or simply according to the flow history.

There is no utility to perform *per hop* AC with fixed-routing schemes: all the exceeding traffic on an output port could be, in this case, shaped by the edge node. However, AC at core nodes becomes interesting when *DR* policies are applied. In fact, *DR* performs implicitly an admission control if the list of deflection alternatives does not contain all the output ports. Hence, several schemes have been proposed to limit the deflection alternatives according to some network state information [18][22].

The AC mechanism integrated in *ADR* proceeds similarly. Based on the feedbacks, core nodes include or exclude forwarding options. All forwarding options might even be excluded. In such a case, a burst is said *blocked*. This differs from the other approaches, which exclude options independently of the network state, and which never exclude them all.

III. *ADR* IMPLEMENTATION

ADR consists of a routing logic, providing ordered forwarding options whenever a control packet arrives at a core node. Its only restriction concerns the structure of the burst control packet (BCP). BCP must contain, at least, a unique identifier (ID), an indication of the burst remaining offset time, and a *history field* permitting to record the nodes visited so far. If an emulated OBS [23] architecture is used, the remaining offset time is replaced by a Time-To-Live field. Additionally, *ADR* requires an access to the signalling channel in order to transmit the feedbacks.

ADR performs four distinct operations:
A) record in a local memory its last decisions
B) send feedbacks to other nodes
C) collect feedbacks to analyse its previous choices
D) decide about burst forwarding

A. Decision record

When a core node *admits* a burst (i.e. schedule a resource reservation), it associates the burst ID with an information triplet (D, R, N) that consists of the *burst final destination (D),* the *burst remaining offset (R),* and the *selected forwarding port (N*). This ID=(D, R, N) association is stored in a local memory.

B. Emission of feedback packets

Each time a burst reaches its destination or is dropped, a feedback is sent to all its previously visited nodes. This feedback follows the path formerly followed by the burst, but in the opposite direction. Feedback packets contain the ID of the corresponding burst, and either an ACK or a NACK flag.

C. Reception of feedback packets

Upon reception of a feedback message, core nodes retrieve the (D, R, N) triplet associated to the feedback ID. Using these three values, it accesses a specific memory structure, called Time Sliding Feedback Counters (TSFCs, described later on), and increments the number of received feedbacks.

TSFCs store all the feedbacks received in the recent past. They are split in Q cells. Each cell stores the positive and negative feedbacks received in a finite amount of time $[t, t+\Delta]$. Thus, TSFCs record feedbacks collected during a time $Q\Delta$.

The feedbacks received within the time interval $[n\Delta, (n+1)\Delta]$ are stored at the address $n \,(mod\, Q)$. Periodically, at each $t_k = k\Delta$, $k=Q, Q+1, Q+2\ldots$, records taken during time interval $[(k-Q)\Delta, (k-Q+1)\Delta]$ are erased, allowing more recent feedback to be stored.

C. Forwarding and dropping decisions

Upon reception of a BCP, a core node calls its *ADR* routing logic, which immediately refers to its TSFCs. Let us assume that:

- the burst announced by the BCP is destined to d
- it has r units of offset remaining
- it arrives from node s
- current node has M output ports $p_1,\ldots,p_m$

then, the TSFCs corresponding to the triplet (d, r, l) are retrieved, with $l=1 \ldots m$, $l \neq s$, s being deduced from the burst history field (i.e. the TSFCs of all ports except the one the burst comes from). The values

$$v_l = \textit{total feedbacks}$$
$$\pi_l = (\textit{positive feedbacks}/\, v_l\,)$$

are extracted from each retrieved TSFC. If no feedback has been collected yet, π_l is set to 1.

In the next step, the unfavourable forwarding choices are excluded. An option j is considered unfavourable if the following conditions are met:

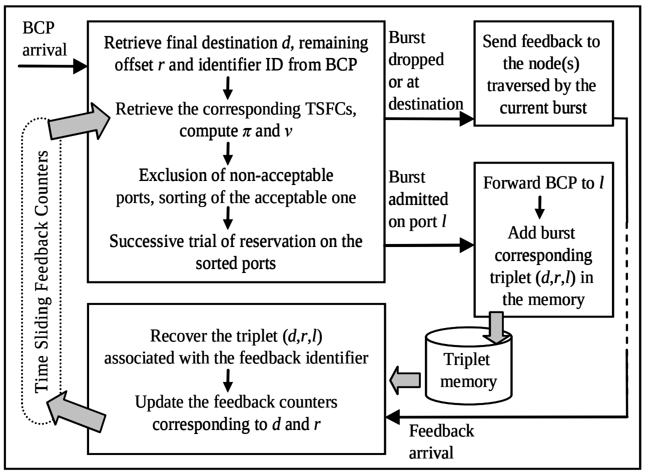
- $\pi_j < \theta_\pi$
- $v_j > \theta_v$

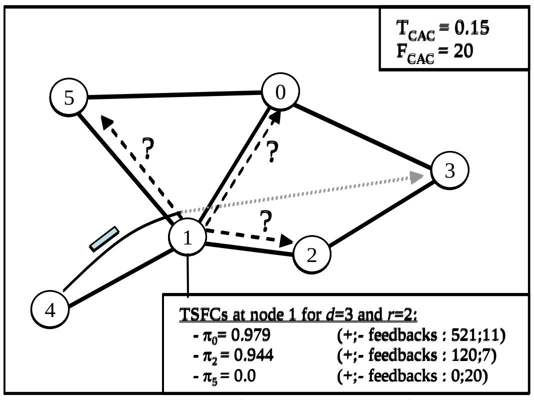
where θ_π and θ_v are fixed thresholds. The first condition excludes ports which led to bad results in the past (i.e. those showing a low π ratio). Setting a minimal number of feedback θ_v avoids excluding a possibility which has not been fully assessed yet.

Once the remaining forwarding alternatives sorted according to the π values, *ADR* operates similarly to the classical DR scheme. A reservation is attempted on the first item of the list. If this attempt fails, the following items are considered until a reservation is scheduled. If two or more options share the same π value, they are chosen randomly.

If a reservation is achieved on port l, the decision is recorded together with the d and r values, as previously discussed. On the contrary, if all the proposed output links are saturated, or if all ports have been excluded, the BCP is not forwarded, and a negative feedback is sent back. The block diagram of Fig. 1 summarises the *ADR* operation.
This procedure is also exemplified through the following example. Assuming the network topology depicted in Fig. 2, a burst destined to node d=3 with r=2 units of remaining offset arrives at node 1 from node 4. Node 1 has three forwarding options: 0, 2 or 5. Option 5 is excluded because:

- $\pi_5=0 \;\; < \theta_\pi = 0.15$
- $v_5=20 \geq \theta_v = 30$


BCP arrival
Retrieve final destination d, remaining offset r and identifier ID from BCP
Retrieve the corresponding TSFCs, compute π and v
Exclusion of non-acceptable ports, sorting of the acceptable one
Successive trial of reservation on the sorted ports
Burst dropped or at destination
Send feedback to the node(s) traversed by the current burst
Burst admitted on port l
Forward BCP to l
Add burst corresponding triplet (d,r,l) in the memory
Triplet memory
Time Sliding Feedback Counters
Recover the triplet (d,r,l) associated with the feedback identifier
Update the feedback counters corresponding to d and r
Feedback arrival
Figure 1. Operational block diagram of the *ADR* scheme.



T_{CAC} = 0.15
F_{CAC} = 20
5
0
3
1
2
4
?
?
?
TSFCs at node 1 for d=3 and r=2:
- π_0= 0.979 (+;- feedbacks : 521;11)
- π_2= 0.944 (+;- feedbacks : 120;7)
- π_5= 0.0 (+;- feedbacks : 0;20)

Figure 2. Node 1 considers the forwarding options for a burst destined to 3.

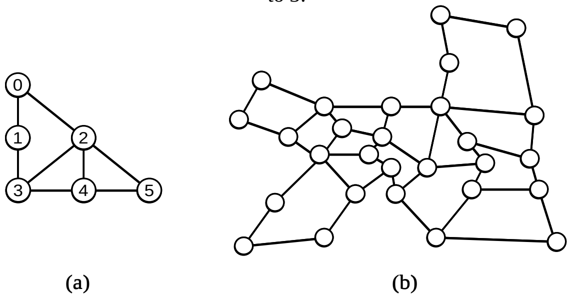


Figure 3. (a) SIMPLE and (b) EON topologies.

Node 1 tries first to schedule the burst on the link towards 0 (π_0 = 0.979). If all the wavelengths of the link are occupied, same operation is made with port towards node 2. If this last attempt fails again, the burst is dropped. In this precise case, no positive feedbacks have been received for next-hop node 5. This is obvious, since route 1-5-0-3 counts 3 hops. No burst holding an offset r = 2 can thus join node 3 following this route.

ADR hence operates solely by sending, counting, and analysing feedbacks. At the initialization stage, only the parameters θ_{π}, θ_v, C and Δ have to be set. Moreover, *ADR* performs all operations (routing, deflection, admission control) in an integrated manner.

IV. NUMERICAL RESULTS

Several simulations have been performed to estimate the performance of our *ADR* scheme using the JAVOBS [24] simulation tool. For all links, 16 wavelengths at 10 Gbit/s have been assumed. Emission rate is normalized to the link capacity. For an offered rate ρ=1, each node emits a total of 160 Gbit/s, uniformly distributed on the remaining nodes. Traffic is generated according to the Poisson model. Mean burst size is fixed to 1.2 Mbit.

Since the performance of *ADR* is independent of the scheduling algorithm used, switching time t_s and processing time t_p are neglected. However, the number of offset unit remaining u_{rem} is still set as it would be with a non null processing time.

Simulations have been performed on two different topologies. First, the SIMPLE network (Fig. 3a), which has 6 nodes and 8 links, has been used to analyse in deep the ADR behaviour. In this case, links are assumed to have very short lengths, and thus, no delay is taken into account for feedbacks.

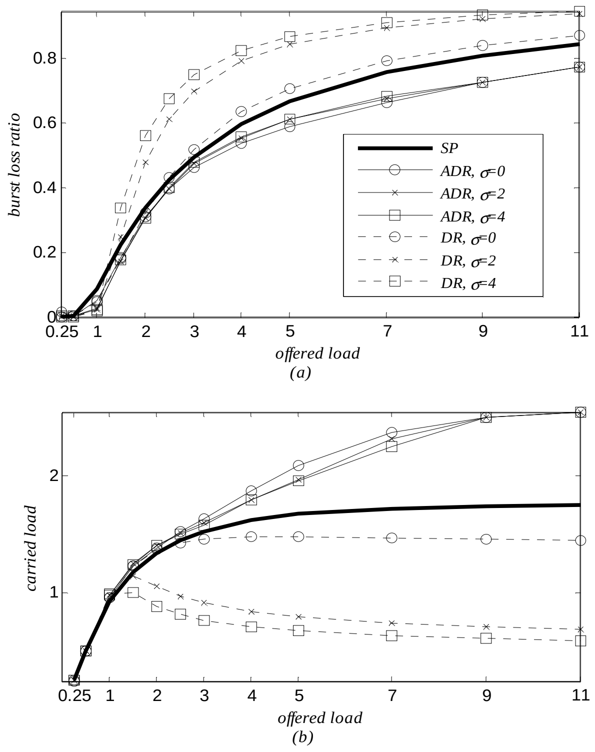


Figure 4: Performance comparison with other routing approaches, for high loads

Second, *ADR* has been tested in more practical situations using the EON topology (Fig. 3b [26]), which consists of 28 nodes and 41 links. Real link lengths, corresponding to geographical distances, have been used to simulate transmission delays, for both burst and feedbacks.

A. Simulations on SIMPLE topology

ADR performance is first compared to the well-known Shortest-Path (*SP*) Routing and to the Deflection Routing (*DR*). It is assumed that *DR* systematically tries all forwarding options which, given the remaining offset, permit to reach the destination. The options requiring fewer hops are considered in priority (random selection for equalities).

A parameter σ taking values 0, 2 or 4 represents an offset time supplement granted to each burst. It can be exploited by both *ADR* and *DR*. The thresholds θ_{π} = 0.5 and θ_v = 10 are used with TSFCs of Q=2000 cells and Δ=40µs.

Performances in terms of burst loss ratio (BLR) and carried load are represented in Fig. 4(a) and 4(b). *ADR* performs better than the other approaches as the load increases. Contrarily to *DR*, *ADR* performances do not fall below the SP. Even more, for ρ>5, while the carried load remains stable as the offered traffic increases (*SP*, *DR* with σ = 0) or even drops (*DR* with σ > 0), the *ADR* carried traffic still increases and gets stable only for extreme loads (ρ>11).

Fig. 5 focuses on the burst loss ratio for lower loads (ρ<2). For almost null rates, *SP* and *DR* exhibit low burst ratios. For ρ=0.1, using *SP* or *DR* and σ = 0, and for

$\rho<0.7$ using *DR* and $\sigma > 0$, no burst loss has even been recorded within the simulated time of 200ms. On the contrary, burst loss ratio of *ADR* is never null, due to two facts. Firstly, *ADR* has no infinite memory and has to relearn periodically the good forwarding options. Secondly, a minor part of the traffic needs to be lost during the learning process, when forward success probabilities are not yet known. However, as load increases, the percentage represented by this lost traffic decreases. For $\rho>1.5$, ADR with $\sigma > 0$ outperforms again the other schemes.

To apprehend the differences between DR and ADR, the load ρ_p offered on the port *p* has been measured, for each port. An average *per port offered load*

$$\overline{\rho} = \frac{1}{P}(\rho_1 + ... + \rho_m)$$

has then been computed, *P* being the total amount of ports and ρ_i the load of port *i*. The recorded $\overline{\rho}$ values are represented in Fig. 6.

Assuming a SP routing and lossless conditions, the ideal average *per port offered load* $\tilde{\rho}_{SP}$ can be computed as

$$\tilde{\rho}_{SP} = \frac{\rho}{(N-1)} \cdot \alpha \cdot \frac{1}{P}$$

where $\rho/(N\text{-}1)$ is the load carried by each route, *P* is the total number of ports in the network and α is the total amount of hops for all routes in the network. Thus, $\tilde{\rho}_{SP}$ corresponds to the load offered on each route, multiplied by the number ports (resources) employed by these routes, and averaged over all ports. On the SIMPLE topology, α=46, *P*=16, and *N*=6, thus $\tilde{\rho}_{SP}$ = 0.575 for ρ=1 and $\tilde{\rho}_{SP}$ = 2.3 for ρ=4. These values are represented on Fig. 6 with thin vertical dotted lines.

For ρ=1, $\overline{\rho}_{SP}$ is just below $\tilde{\rho}_{SP}$ since $\overline{\rho}_{SP}$ is decreased by the burst losses [27]. On the contrary, $\overline{\rho}_{DR}$ and $\overline{\rho}_{ADR}$ are greater than $\tilde{\rho}_{SP}$, especially when $\sigma > 0$. This is because in *DR* and *ADR* the amount of intermediate nodes (hops) is higher due to the deflection. The offset time supplement σ, contributes obviously to the increase in the number of traversed hops since it extends burst lifetimes.

For ρ=4, $\overline{\rho}_{SP}$ is drastically reduced by the losses and appears clearly below $\tilde{\rho}$. However, the largest difference consists in the explosion of $\overline{\rho}_{DR}$. Indeed, a more loaded network conducts *DR* to exhaust all forwarding possibilities before dropping a burst. Giving a high offset time bonus strengthens this effect. This increased amount of extra visited ports explains why performance drops at high loads. With *ADR* on the contrary, $\overline{\rho}_{ADR}$ is far below $\tilde{\rho}_{SP}$, and even below $\overline{\rho}_{SP}$. Hence, due to admission control mechanisms, *ADR* drops in excess bursts prior to offering them to any port. This action spares capacity for other bursts and explains why performances are not affected.

Fig.7 represents the amount of bursts dropped after a journey of *n* hops. For ρ = 1, *DR* and *ADR* do not differ much in terms of hops traversed by dropped bursts. For ρ = 4, lost bursts travel a much longer distance before being dropped (for σ = 4, about 1000 burst travelled over 6 hops). On the contrary, *ADR* blocks earlier these "resource waster". Since *SP* does not use offset larger than required and since the longest path includes three hops, no bursts are dropped after 3 hops.

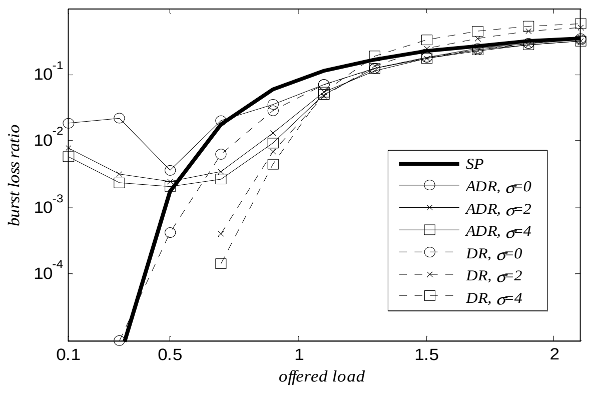


Figure 5. Performance comparison with other routing approaches, for low loads.

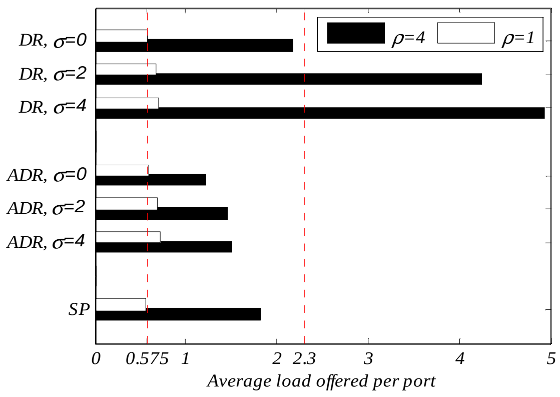


Figure 6: Average per port offered load $\overline{\rho}$ for SIMPLE topology. Dotted lines represent $\tilde{\rho}$ estimates.

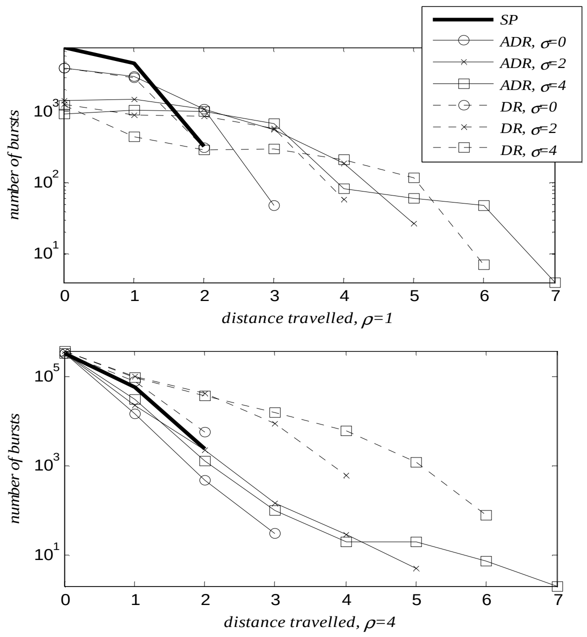


Figure 7. Record of the distances (in hops) travelled by dropped bursts.

TABLE I
BREAKDOWN OF THE REASONS FOR BURSTS LOSSES

		Hops achieved when dropped							
σ	***Drop reason***	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***
2	***No availabilities***	42.97%	98.67%	95.41%	38.37%	6.06%	0.00%	46.73%	
	Blocked	**57.03%**	**1.33%**	4.59%	6.40%	0.00%	0.00%	53.24%	
	Offset exhausted	0.00%	0.00%	0.00%	55.23%	93.94%	100.00%	0.03%	
4	***No availabilities***	45.50%	99.01%	92.36%	87.29%	91.30%	31.82%	25.00%	0.00%
	Blocked	**54.50%**	**0.99%**	7.64%	12.71%	8.70%	9.09%	0.00%	0.00%
	Offset exhausted	0.00%	0.00%	0.00%	0.00%	0.00%	59.09%	75.00%	100.00%

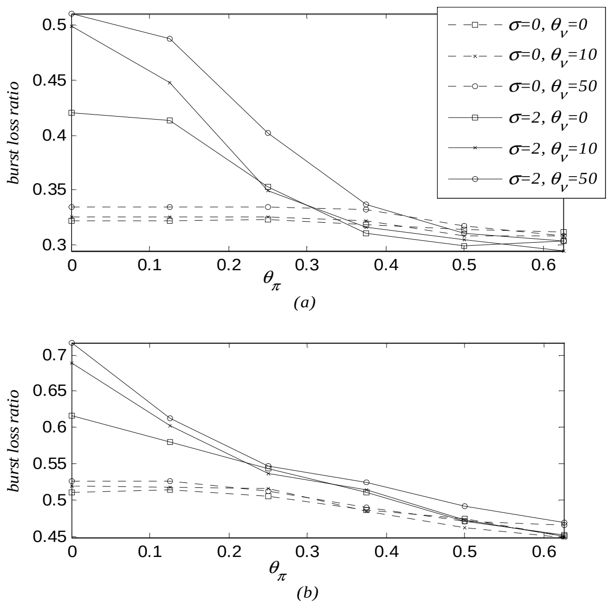
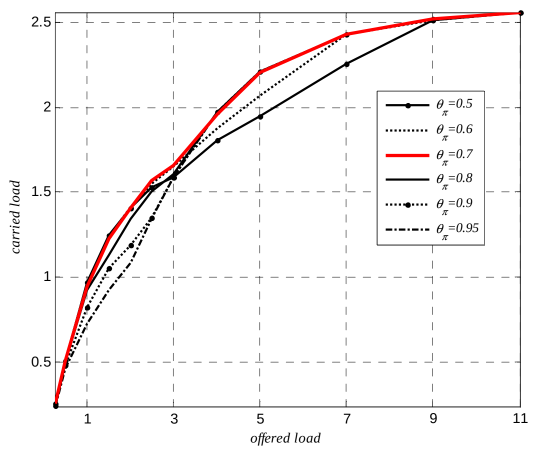


Figure 8. Burst loss vs. θ_π, for *ρ=2 (a)* and *ρ=3*



(b).

Figure 9. Performance of ADR for various θ_π values.

Finally, the reasons generating burst drops are analysed in Table 1. Among the bursts dropped after 0 hops (i.e. at the network entrance), about half is *blocked*. A blocked burst is not offered to any port prior to be dropped. After one hop, only about 1% of the bursts are blocked. Thus *ADR* drops exceeding traffic at network entrance, which considerably limits the waste of resources.

Additional experiments have been driven to measure the impact θ_π and θ_v. As depicted in Fig. 8, high values of θ_π improve the performance at high loads and when additional offset is provided. However, for low loads, a high θ_π is a handicap. θ_v has a limited impact on the performance. The capabilities of the ADR scheme for various offered loads and various θ_π values are illustrated by Fig. 9. For $\rho<3$, values of $\theta_\pi \leq 0.7$ lead to the best performance, while for higher loads, θ_π has to be ≥ 0.7 to reach the best carried load. Thus we select the value $\theta_\pi = 0.7$ hereafter.

B. *EON topology*

Simulations have also been performed on the EON topology under similar conditions, except that, in this case, the σ parameter takes the values 0, 1, 2 or 4 and that θ_π is fixed to 0.7.

As in the SIMPLE topology case, ADR is compared to both SP and DR approaches. Performances are represented in Fig. 10. For high loads, SP does not only provide better performances than DR, but also than ADR.

Two reasons account for that. Firstly, half of the nodes of the EON topology are peripheral. They emit anyway the same traffic than other central nodes. Thus, the core part of the network quickly becomes a bottleneck as load increases. The *SP* scheme, by trying only once to cross the bottleneck, drops the bursts earlier, which spares resources for other bursts.

To explain the second reason for *SP* outperforming, the burst losses over time have been plotted in Fig. 11, for ρ=4. Three phases can be distinguished. During the first 16 ms, several losses are due to exhausted offset times. ADR, without knowing where to forward particular bursts, causes many of them to finish their lives far from their destination. Due to these frequent mislaid bursts, network resources are highly utilised. The number of burst dropped for unavailability is therefore high, too. After this first phase, *ADR* learns from its mistakes and does not mislay bursts anymore, for the next 65ms. On the contrary, it starts to block bursts instead of forwarding them. This blocking spares the resources, which contributes to lower the unavailable links.

After 90ms of simulation, the number of lost bursts increases again and the number of blocked burst falls, which leads to an explosion of the total losses. This is due to the fact that core node memories are limited to 2000 cells of 40μs each, giving a total memory time of 80ms.

Upon expiration of this delay, memories will start to override the information contained in their first cells, causing a progressive “amnesia”. This obliges the network to relearn the routing information.

Thus, *ADR* underperforms *SP* because it must deduce repeatedly the network structure, and needs to lose bursts in the network for this purpose. In the less complex SIMPLE topology, the network was easy to apprehend. In the EON, by contrast, many routes have to be excluded prior to find the right paths.

C. Improvement of ADR

To avoid the aforementioned handicap, the *ADR* scheme described in subsection III-C is modified in the following way. A forwarding option is still considered unfavourable if the values stored by the corresponding TSFC do not match the θ_π and θ_v thresholds. However, a second requirement is added. A burst with final destination *d* and remaining offset *r* is forwarded to the output port *j* if and only if

$$ShortestPath(j,d) \leq r+1$$

This condition guarantees that a burst will be forwarded only if it carries enough offset units to flow from the output terminal node to its destination. The same mechanism than *DR* is thus implemented. We call this modified scheme Adaptive Restricted Deflection Routing (*ARDR*).

In Fig. 12, the analysis of the burst drops over time permits to visualise the effect of the route restriction. Using *ARDR*, during the learning phase, the number of blocked burst increases steeper since fewer trials are required before starting to block burst. The number of burst dropped due to their insufficient remaining offset is oblivious null with *ARDR*. *ARDR* is also affected by the amnesia effect, but relearning is again achieved quicker due to the additional restriction. On the top lines, one can see that *ARDR* generally leads to a reduction of the number of losses.

Finally, the comparison depicted in Fig. 10 has been reproduced, substituting *ARDR* to *ADR* in Fig. 13. The *ARDR* outperforms the SP approach for σ=0 and leads to similar performances for σ=1.

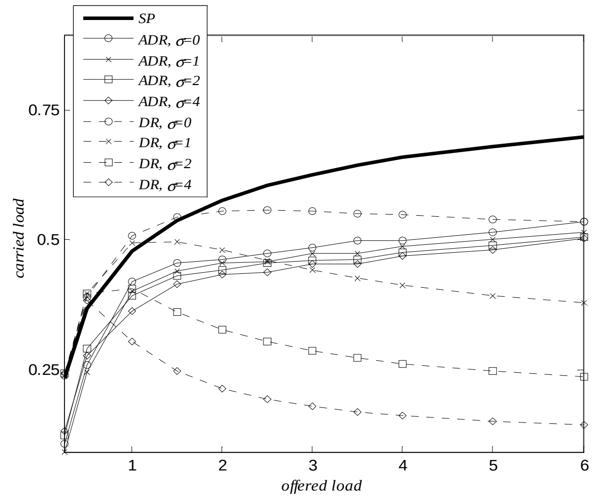


Figure 10. Performance of shortest path routing (SP), deflection routing (DR) and Adaptive Deflection Routing (*ADR*) in terms of carried load using the EON topology.

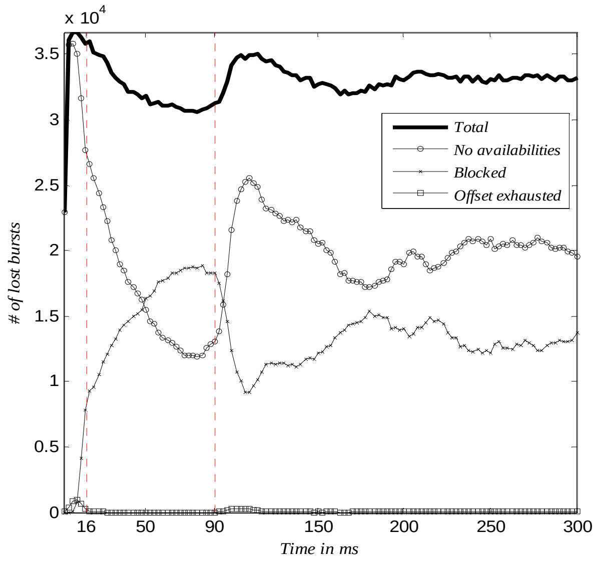


Figure 11. Lost bursts along simulation time.

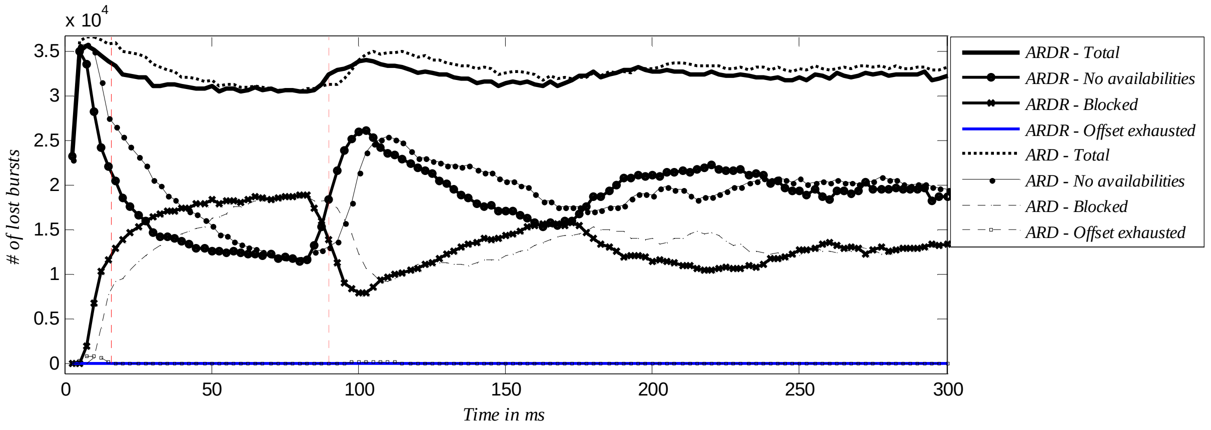


Figure 12. Comparison of the ARDR and ARD

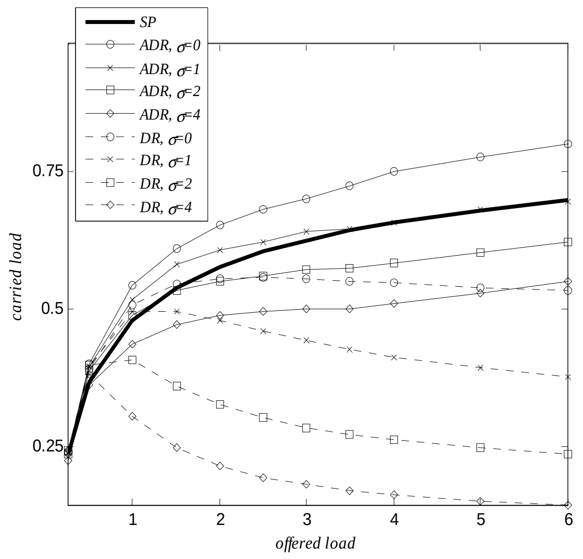


Figure 13. Comparison of the ARDR scheme with SP and DR routing approaches, on the EON topology

V. CONCLUSIONS

In this paper we propose an Adaptive Deflection Routing scheme, able to achieve simultaneously the operations: deflection routing, load balancing, and admission control. *ADR* performs all its operations by processing and analysing the feedback messages exchanged by core nodes. Nodes do not require any configuration, since they learn from past experience. Thus, the overhead required by *ADR* (TSFC, decisions mechanisms) is compensated by its simplicity.

ADR performance has been analysed by numerical simulation using the tool JAVOBS. Without extensively trying to deflect bursts impossible to dispatch, *ADR* clearly outperforms the *DR* scheme, especially for high loads. However, *ADR* has one disadvantage. It requires a minimal number of burst losses to be able to correctly apprehend the network architecture. This is problematic when low loads are injected or over networks including a large number of nodes.

To mitigate this penalty in large networks, a modified version of the *ADR* scheme, the Adaptive Restricted Deflection Routing has been set up. This improvement reduces the number of burst losses happening in the learning stage. Contrarily to *ADR*, *ARDR* outperforms the SP on the EON large topology. Unfortunately, the routing restriction requires the core nodes to be aware of the network topology, since it requires the knowledge the shortest paths for each source-destination pair.

The approach consisting in performing routing, deflection and admission control decisions relying only on feedbacks, without prior knowledge of link utilization, appears to be valid. This permits to achieve all these operations directly at the OBS core nodes, and spares an additional overlaid control plane. However, further analysis must be conducted, in particular regarding the convergence of both *ADR* and *ARDR* techniques. Alternative ways of storing feedback, as well as other ways to use them, could also be investigated.

REFERENCES

[1] C.-F. Hsu, T.-L. Liu, N.-F. Huang, “Performance Analysis of Deflection Routing in Optical Burst-Switched Networks”, *IEEE INFOCOM*, 2002.

[2] C. Qiao, M. Yoo, “Optical Burst Switching (OBS) – a new paradigm for an optical internet”, *J. High Speed Network* vol. 8, no. 1, pp. 69-84, 1999.

[3] S. Yao, B. Mukherjee, S. J. B. Yoo, S. Dixit, “A Unified Study of Contention-Resolution Schemes in Optical Packet-Switched Networks”, *J. of Lightwave Technology*, vol. 21, no. 3, 2003.

[4] Y. Chen, J. S. Turner, P.-F. Mo, “Optimal Burst Scheduling in Optical Burst Switched Networks”, *J. of Lightwave Technology*, vol. 25, no. 8, 2007.

[5] X. Wang, H. Morikawa, T. Aoyama, “Burst optical deflection routing protocol for wavelength routing WDM networks,” *SPIE Opt. Netw. Commun. Conf.*, 2000.

[6] A. Zalesky, H. L. Vu, Z. Rosberg, E. W. M. Wong, M. Zuckerman, “Stabilizing Deflection Routing In Optical Burst Switched Networks”, *J. on Selected Area in Communications*, vol. 25, no. 6, 2007.

[7] H. Tanida, K. Ohmae, Y.-B. Choi, H. Okada, “An Effective BECN/CRN Typed Deflection Routing for QoS Guaranteed Optical Burst Switching”, *IEEE GLOBECOM*, 2003.

[8] M. Klinkowski, D. Careglio, J. Solé-Pareta, “Reactive and proactive routing in labelleed optical burst switching networks”, *IET Commun.*, vol. 3, no 3, pp. 454-464, 2009.

[9] J. Zhang, H.-J. Lee, S. Wang, X. Qiu, K. Zhu,Y. Huang, D. Datta, Y.-C. Kim, B. Mukherjee “Explicit Routing for Traffic Engineering in Labeled Optical Burst-Switched WDM Networks”, *ICCS*, LNCS 3038, Springer, 2004.

[10] M. Klinkowski, M. Pióro, D. Careglio, M. Marciniak, and J. Solé-Pareta, “Non-linear Optimization for Multipath Source-Routing in OBS Networks”, *IEEE Communications Letters*, vol. 11, no. 12, 2007.

[11] P. Pedroso, J. Solé-Pareta, D. Careglio M. Klinkowski, “Integrating GMPLS in the OBS Networks Control Plane”, *IEEE ICTON* 2007.

[12] G. Thodime, V.Vokkarane, J. Jue, ”Dynamic Congestion-Based Load Balanced Routing in Optical Burst-Switched Networks”, *IEEE GLOBECOM* 2003.

[13] L. Yang, G. N. Rouskas, “Adaptive Path Selection in OBS Networks”, *J. of Lightwave Technology*, vol. 24, no. 8, 2006.

[14] D. Ishii, N. Yamanaka, I. Sasase, “Self-learning route selection scheme using multipath searching packets in an OBS network,” *J. Opt. Netw*. Vol. 4, no. 7, pp. 432-445, 2005.

[15] S. Ganguly, S. Bhatnagar, R. Izmailov, C. Qiao, “Mutli-path Adaptive Optical Burst Fowarding”, *IEEE HPSR* 2004.

[16] X. Gao, M. A. Bassiouni, “Fairness-Improving Adaptive Routing in Optical Burst Switching Mesh Networks”, *IEEE ICC* 2008.

[17] H. Pan, T. Abe, Y. Mori, Y.-B. Choi, H. Okada, “Feedback-based Load Balancing Routing for Optical Burst Switching Networks”, *IEEE Asia-Pacific Conference on Communications*, 2005.

[18] S. Lee, K. Sriram, H. Kim, J. Song, “Contention-Based Limited Deflection Routing Protocol in Optical Burst-Switched Networks”, *J. on Selected Area in Communications*, vol. 23, no. 8, 2005.

[19] J. Perelló, S. Spadaro, J. Comellas, G. Junyent, “Burst Contention Avoidance Schemes in Hybrid GMPLS-enabled OBS/OCS Optical Networks”, *ONDM* 2009.

[20] T. Coutelen, H. Elbiaze, B. Jaumard, “An Efficient Adaptive Offset Mechanism to Reduce Burst Losses in OBS Networks”, *IEEE GLOBECOM* 2005.

[21] F. Farahmand, Q. Zhang, J. P. Jue, “A Feedback-Based Contention Avoidance Mechanism for Labeled Optical Burst Switching Networks”, *Photon. Netw. Commun.*, vol. 14, no. 3, pp. 307-316, 2007.

[22] Y. Chen, H. Wu, D. Xu, C. Qiao, “Performance Analysis of Optical Burst Switched Node with Deflection Routing”, *IEEE ICC* 2003.

[23] M. Klinkowski, D. Careglio, J. Solé-Pareta, and M. Marciniak, “Performance Overview of the Offset Time Emulated OBS Network Architecture”, *J. of Lightwave Technology*, vol. 27, no. 14, 2009.

[24] O. Pedrola, M. Klinkowski, D. Careglio, J. Solé-Pareta, S. Rumley, C. Gaumier, “JAVOBS: A Flexible Simulator for OBS Network Architectures”, *J. of Networks*, vol. 5, no. 2 pp. 256-264, 2010.

[25] J. Teng, G. N. Rouskas, “A detailed analysis and performance comparison of wavelength reservation schemes for optical burst switched networks”, *Photon. Netw. Commun*, vol. 9, 2005.

[26] B. Mikac, R. Inkret, A. Kuchar, “Advanced infrastructure for photonic networks, *extended final report of Cost Action 266,* ISBN 953-184-064-4.

[27] Z. Rosberg, H. L. Vu, M. Zuckerman, J. White, “Performance analyses of optical burst-switching networks”, *IEEE J. on Selected Areas in Communications,* vol. 21, no.7, pp. 1187-1197, 2003.

Sébastien Rumley received the M.S degrees in Communication Systems for the Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, after studies in Lausanne, Zurich and Santiago de Chile. Since 2005 he is with the Laboratoire de Télécommunation of EPFL. His research focuses on software engineering of network simulators and planners, applied to optical networks.

Christian Gaumier received his Ph.D. from Ecole Polytechnique Fédérale de Lausanne (EPFL) in 1995. His doctoral research focused on modelling the propagation of signals over singlemode fibres in both linear and nonlinear regimes. Since 2001 he is director of the Telecommunications Laboratory of EPFL. His active research area includes dispersion compensation techniques, measurement techniques for fibre optics and dimensioning and performance analysis of core photonic communication networks. He participated to several European joint projects and is author and co-author of more than 50 publications in conferences, journals and books. Dr. Gaumier is member of Communication Society of IEEE.

Oscar Pedrola received the M.S. degree in Telecommunications engineering and the M.S. degree in information and communication technologies both from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain in 2008. Currently, he is a PhD student at the UPC, in the Broadband Communications Research Group (CBA). At present, he is involved in the FP7 Network of Excellence BONE. His current research interests are in the field of optical networks with emphasis on burst/packet based switching technologies.

Josep Solé-Pareta obtained his M.Sc. degree in Telecom Engineering in 1984, and his Ph.D. in Computer Science in 1991, both from the Technical University of Catalonia (UPC). In 1984 he joined the Computer Architecture Department of UPC. Currently he is Full Professor with this department. He did a Postdoc stage (summers of 1993 and 1994) at the Georgia Institute of Technology. He is cofounder of the UPC-CCABA (http://www.ccaba.upc.edu/). His publications include several book chapters and more than 100 papers in relevant research journals (> 20), and refereed international conferences. His current research interests are in Nanonetworking Communications, Traffic Monitoring and Analysis and High Speed and Optical Networking, with emphasis on traffic engineering, traffic characterization, MAC protocols and QoS provisioning. He has participated in many European projects dealing with Computer Networking topics.