Photon Netw Commun (2013) 26:84–94
DOI 10.1007/s11107-013-0411-5

Channel reusability for burst scheduling in OBS networks

Gustavo B. Figueiredo · Nelson L. S. da Fonseca

Received: 9 February 2013 / Accepted: 7 August 2013 / Published online: 25 August 2013

Abstract This paper presents a novel channel scheduling policy for optical burst switching networks called least reusable channel (LRC). LRC decides to which interval of the output channel (void) an incoming burst should be allocated on the basis of reuse of the remaining voids. LRC dynamically uses information available to make allocation decisions. It is shown here that LRC produces lower blocking probability and distributes losses more uniformly among routes than do other existing scheduling policies.

Keywords OBS networks · Channel scheduling · Channel reusability

1 Introduction

In optical burst switching (OBS) networks, packets are aggregated at edge nodes to create transmission units called bursts. A control packet is transmitted out-of-band, ahead of the burst, so that sufficient bandwidth can be reserved for the associated data burst. The control packet carries information about the burst, such as its size and the offset time, i.e., the time interval between the arrival of the control packet and the arrival of the data burst. One commonly used protocol for resource reservation in OBS networks is the just-enough-time (JET) [1]. JET reserves the channel for the duration of the transmission of a burst, starting at the expected arrival time (given by the offset time minus the burst processing time). If the request for bandwidth reservation is granted, a new offset time is calculated, and this information is inserted into the control packet being forwarded to the next hop along the route.

Since nodes at the network border do not wait for the confirmation of bandwidth reservation to transmit a burst, an incoming burst will be discarded at a core node if bandwidth has not been reserved for it. Schedulers at core nodes reserve bandwidth for incoming bursts, and this assignment should minimize the number of bursts lost. One way of minimizing the chance of the loss of a burst is to allocate bandwidth to maximize the chances of allocation. For this, scheduling policies need to have information about both channel occupancy and the quality of service requirements of users.

The occupancy of the output channel alternates between periods of occupancy and periods of idleness, called voids. These void intervals can be used to accommodate the transmission of new bursts. Indeed, a void interval, I_j, defined by its initial time, s_j, and that of termination, e_j, can be allocated to a burst with arrival time, r_s, and departure time, r_f, if and only if $s_j \leq r_s$ and $r_f \leq e_j$. Figure 1 illustrates the two voids created by the allocation of a burst: the preceding void (p) from s_j to r_s and the succeeding (s) void from r_f to e_j. However, since bursts have different offset times, they may arrive in a different order than that of their control packets. This can lead to fragmentation of the occupancy of the output channels.

When schedulers do not make appropriate decisions, bursts can be lost unnecessarily. Figure 2 provides an example of a situation in which poor scheduling decisions lead to the loss of bursts: Let A, B and C be control packets arriving in this order, and let their corresponding bursts arrive in the same order. If channel 1 were used for burst A and channel 2 for burst B, there would be no possibility of accommodating

G. B. Figueiredo (✉)
Department of Computer Science, Federal University of Bahia, Salvador, Bahia, Brazil
e-mail: gustavo@dcc.ufba.br

N. L. S. da Fonseca
Institute of Computing, University of Campinas, Campinas, São Paulo, Brazil
e-mail: nfonseca@ic.unicamp.br

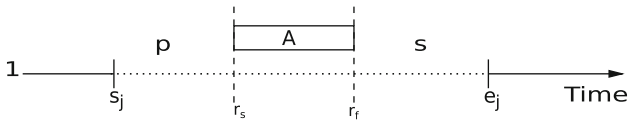

Fig. 1 Creation of voids by the allocation of a burst

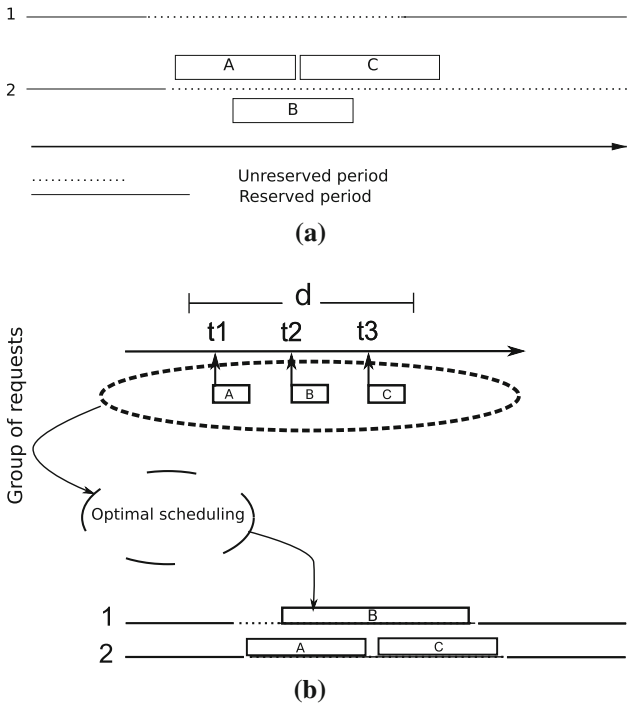

Fig. 2 Example of how batch scheduling can avoid losses of bursts. **a** Example of poor application of greedy strategies. **b** Batch scheduling of channels leading to no losses

burst C. However, if channel 1 were used for burst B and channel 2 for bursts A and C, no loss would occur. In this example, loss occurs because scheduling decisions did not consider the arrival of incoming requests.

The efficiency of a channel scheduling algorithm is determined by its capacity to minimize the blocking probability of requests (loss of data bursts). Although scheduling algorithms [2–5] proposed so far base their decisions on the chances of a remaining void being used by incoming requests, these algorithms fail to consider information dynamically available for making their decision.

This paper introduces a scheduling mechanism called *least reusable channel* (LRC) that first allocates voids that have a small chance of being allocated by future requests; such an allocation increases the probability of the utilization of the remaining voids by future requests. The proposed mechanism takes into account the traffic load as well as the probability of preceding and succeeding voids being able to accommodate incoming bursts. Numerical examples show that this approach produces lower burst losses than do previously proposed policies.

This paper is organized as follows. The next section reviews previous work. Section 3 explains issues in burst scheduling, which are addressed by the proposed discipline. Section 4 introduces the LRC discipline and its performance is analyzed in Sect. 5. Section 6 draws some conclusions.

2 Previous work

Several burst scheduling policies have been proposed in the past few years [2–13]. Turner [3] introduced the latest available unused channel (LAUC [14]) discipline also known as *Horizon* since it keeps track of the time on horizon to make the reservation. In this policy, a burst is scheduled on the channel with the lowest horizon value. This policy is simple and can be implemented in $O(\log W)$ steps, where W is the number of channels. However, it can lead to high burst loss rates and low utilization. In Xiong et al. [14], variants of this policy were proposed, but all of them have the same drawback of the original policy, since such limited information is used in the decision process. Figure 3 illustrates an allocation scenario in which the channel chosen by LAUC would be channel 0.

Variations of LAUC all leading to similar low network utilization and high blocking probabilities have also been proposed. The simplest version of LAUC is the first fit algorithm, which searches for available channels in a fixed and preestablished order (like round-robin) and picks the first available one.

To increase network utilization while decreasing blocking probability, Xiong et al. [2] proposed an algorithm entitled latest available unused channel with void filling (LAUC–VF) that uses information about existing voids. It schedules bursts in voids which have the latest starting time prior to the time of this arrival. It out performs LAUC, but at the cost of increased computational complexity, which is $O(W \log M)$, where M is the number of reservation requests and W the number of channels [15]. A simple adaptation of LAUC–VF is the RANDOM algorithm that randomly chooses a void among all feasible voids previously selected. In the scenario in Figure 3, the channel chosen by LAUC–VF would be channel 2. The RANDOM algorithm randomly chooses one among the 5 channels.

Xu et al. [16,17] introduced the MIN–SV, MIN–EV and *Best fit* policies. MIN–SV tries to minimize the interval between the beginning of a void and the starting time of the request being processed. Although similar to LAUC–VF, it searches for voids using a binary decision tree which results in a computational complexity of $(O(\log(M)))$, where M is the number of reservation requests. MIN–EV, on the other hand, tries to minimize the size of the void created by the new request and already existing reservations, and the *Best fit* policy tries to minimize the total void size. Results derived

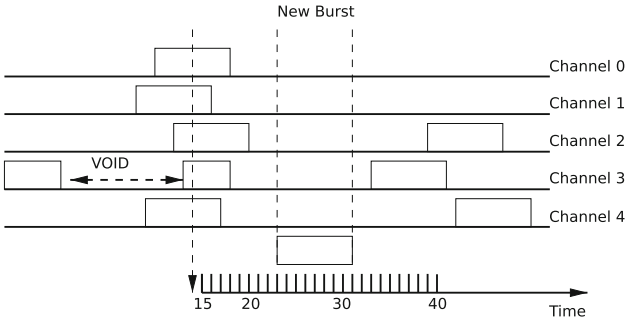

Fig. 3 Channel scenario for application of different policies

via simulation show that MIN–SV produces the best performance of the three policies. In Fig. 3, the channels chosen by the algorithms MIN–EV, MIN–SV and Best fit would be 3, 2 and 3, respectively.

The priority-based wavelength assignment (PWA) policy [18] assigns priority to output channels based on the accumulated loss of bursts on each channel. It assigns an incoming burst to the channel with lowest accumulated loss and consequently highest priority. Although efficient, this policy implies a high overhead for recording the accumulated loss of each channel.

The burst overlapping reduction algorithm (BORA) [19] avoids simultaneous arrivals of bursts (overlapping degree) at the output channel. BORA is implemented at the edge nodes to capitalize on the ability to electronically store burst by introducing delays to avoid burst losses. Several variations were proposed in Li and Qiao [19]. Both BORA and PWA are implemented at the network edge, which is in contrast with what has been described for other policies.

Detti et al. [20] proposed a burst scheduling algorithm with contention resolution called optical composite burst switching (OCBS) that minimizes the portion of the burst that will be discarded in the case of contention. In Vokkarane and Jue [21], the authors proposed a similar algorithm using more sophisticated strategies involving a combination of different actions, such as scheduling in alternative channels and segmentation of burst to allow partial transmission.

In Chang and Park [22], an algorithm is proposed that selects bursts considering the order of their arrival, rather than the order of the arrival of their control packets. In Kumar and Kumar [6], the authors propose a channel scheduling algorithm called burst delay feedback algorithm (BDFA) that combines the use of FDLs, random increase in offset time and window-based channel. In the window-based scheme, bursts are delayed so that the number of channels necessary to accommodate all bursts is decreased.

Ichikawa and Kamakura [7] proposed a forward resource reservation (FRR) scheme and a control packet scheduler to minimize the occurrence of buffer overflow of control packet and its associated blocking probability. However, differently from the above-mentioned schemes, the scheme proposed in Ichikawa and Kamakura [7] is designed to be employed at edge nodes.

Netak et al. [8] proposed a channel scheduling policy called reverse scheduling. In reverse scheduling, feasible voids are searched first, and if there are no such voids, the void associated with the horizon is chosen. In spite of this change, reported results show that the proposed scheme has burst loss ratio as low as that of void filling algorithms.

In Wu et al. [9], the authors proposed an index-based parallel scheduler based on the LAUC–VF that was built over an index vector of voids for each data channel. That index vector is used to and search for the feasible voids. The whole searching process was shown to run in $O(1)$ time which can make the scheduling process achieve high processing speed and high channel utilization at the same time.

Rogiest et al. [10] proposed a channel and delay selection algorithm to solve contention problem. The algorithm considers a set of voids and FDLs and tries to make the best decision about on which channel or FDL should the burst be scheduled. Results show that the algorithm produces better results than similar algorithms. Its main drawback is the fact that it only works on networks with FDLs.

In Figueiredo and da Fonseca [11], an linear time algorithm was proposed to schedule requests. Differently than the previously mentioned algorithm, the GreedyOPT algorithm considers a batch of request collected over a period of time. Since requests have to wait for processing, it is possible that an increase in the end-to-end delay occurs.

None of these previously scheduling policies, however, evaluate the potentiality of a void for utilization on the basis of dynamic conditions as is done by the proposed LRC mechanism.

3 Reuse of channels in the scheduling of bursts

Two concepts to the understanding of the LRC algorithm are explained in this section: the useful lifetime of a void and the probability of inversion of control packets and bursts. The LRC policy evaluates the potential use of a void for the scheduling of incoming bursts and selects the void with the lowest chance of being reused. Moreover, the algorithm must know the potentiality that newly generated voids have for the accommodation of incoming requests, as well as the probability that these requests will fit into each void.

3.1 Useful lifetime of voids

To evaluate the capacity of a void to accommodate incoming bursts, a metric called useful lifetime is used. This metric should be taken into consideration by the scheduling algorithm, since the capacity of voids to accommodate bursts

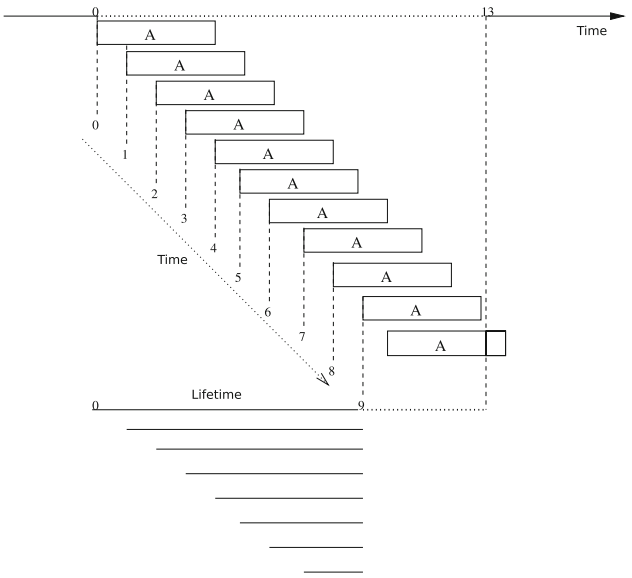

Fig. 4 Useful lifetime of a void

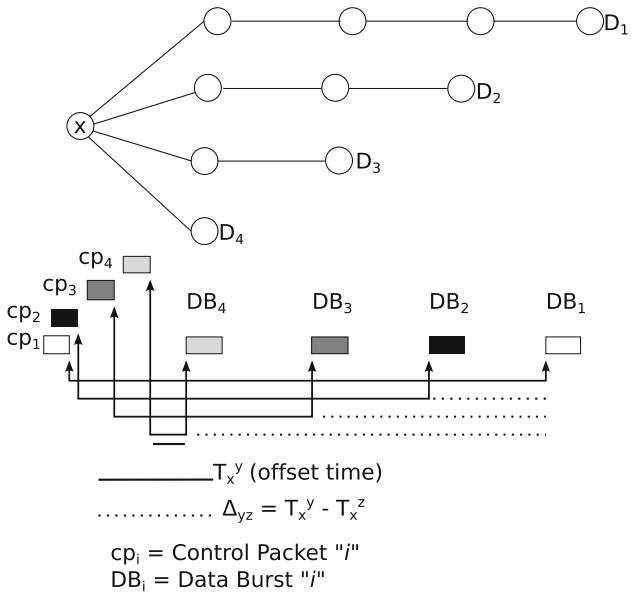

Fig. 5 Inversion in the arrival order of requests

diminishes with time, as illustrated in Fig. 4. The useful lifetime of a void is defined as the duration of the void created by a scheduled burst minus the mean burst transmission duration ($\overline{\beta}$), which can be obtained by keeping track of the reservations passing through the node. The mean burst duration is decremented from the duration of the void since any void shorter than the average burst size of an incoming flow has a low chance of accommodating incoming bursts.

Let s_j, e_j, r_s and r_f be the starting and ending time of a void of the rth request, the burst arrival time and departure time, respectively, and $\overline{\beta}$ the burst mean duration. The preceding void a_j, generated by the allocation of the rth request (see Fig. 1), can only accommodate a burst with transmission duration $\overline{\beta}$ until the instant $r_s - \overline{\beta}$; after that, the remaining void will be shorter than $\overline{\beta}$. Thus, the useful lifetime of the preceding void a_j is given by:

$$\upsilon(a_j) = r_s - \overline{\beta} - s_j = r_s - (s_j + \overline{\beta}). \quad (1)$$

Similarly, the useful lifetime of the succeeding void p_j is given by:

$$\upsilon(p_j) = e_j - \left(r_f + \overline{\beta}\right) \quad (2)$$

3.2 Arrival inversion

To identify the potentiality for the reuse of each new void created by the allocation of a request (see Fig. 1), it is necessary to estimate the chances that incoming bursts will fit into the preceding void. It is necessary, thus, to evaluate the chances that a data burst arrives prior to another burst whose control packet has already arrived. Such an event, in this paper, is called arrival inversion.

Figure 5 illustrates a scenario in which arrival inversion can potentially occur. Suppose node x is an intermediate node common to several source–destination paths (S_y, D_y) $y = 1 \ldots 4$. Moreover, suppose that the control packets of bursts sent to destinations D_1 and D_2 arrive simultaneously at node x at time t. The burst destined to D_1 arrives at node x at time $t' = t + T_x^1$ and the burst sent to D_2 arrives at time $t'' = t + T_x^2 = t + T_x^1 - \Delta_{12}$, where T_x^y is the offset time at node x of the control packet sent to destination y and $\Delta_{yz} = T_x^y - T_x^z$.

Two conditions are necessary for the occurrence of arrival inversion. The first is that the offset time of one control packet should be longer than that of a succeeding control packet. The second condition is that this second control packet should arrive no later than Δ_{yz} after the arrival of the first control packet, where Δ_{yz} is the difference between the two offset times. Although these conditions lead the inversion between the instant of arrival of the two bursts associated with these two control packets, they do not ensure that the burst associated with the second control packet would be completely allocated before the burst associated with the first control packet. For that to occur, it is necessary that the second control packet arrives no later than $\Delta_{yz} - \beta$ after the arrival of the first control packet, where β is the size of the second burst.

Figure 6 illustrates the idea of arrival inversion. Suppose that the control packet cp_1 destined to node y arrives at node x at time t_l^y. Consequently, its associated data burst, DB_1, will arrive at node x at time $t_l^y + T_x^y$. If the control packet cp_2 arrives at most $\Delta_{yz} - \beta$ after t_l^y, the data burst DB_2 will be entirely transmitted before the burst DB_1. On the other hand, if the control packet cp_3 arrives into the period $[(t_l^y + \Delta_{yz} - \beta); (t_l^y + \Delta_{yz})]$, the instant of arrival of bursts

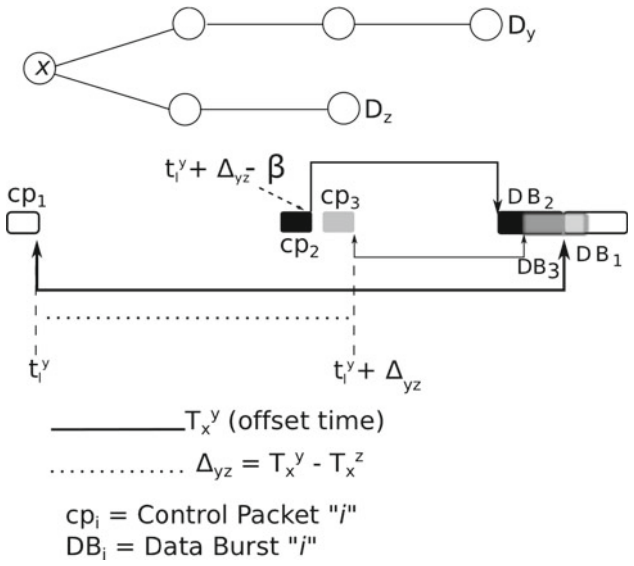

Fig. 6 Inversion: arrival of the second control packet within the interval $[t_l^y; t_l^y + \Delta_{yz} - \beta]$

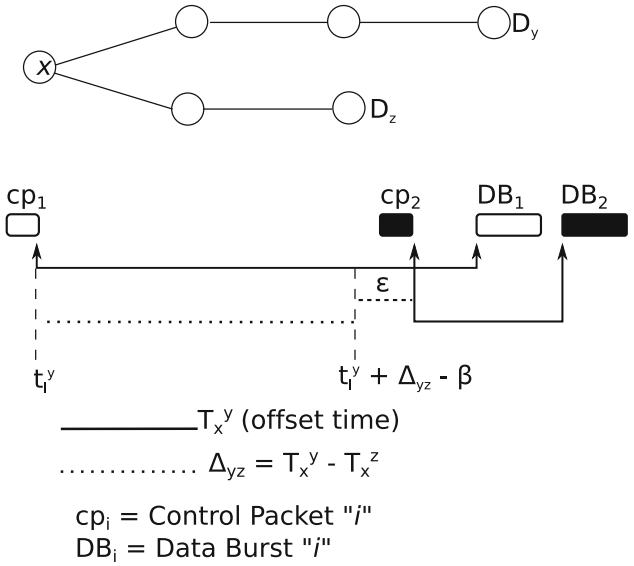

Fig. 7 Noninversion: arrival of the second control packet after the interval $[t_l + y; t_l^y + \Delta_{yz} - \beta]$

DB_1 and DB_2 will be inverted, but the second burst will not be entirely transmitted before DB_1.

Conversely, if the control packet destined to node z arrives $\Delta_{yz} + \epsilon$ time units after the control packet destined to node y, there is no inversion, and the burst destined for node z will arrive at node x after the burst destined for node y, as illustrated in Fig. 7.

3.3 Probability of arrival inversion

In this subsection, the computation of probability of arrival inversion will be derived considering a variety of burst assembly policies.

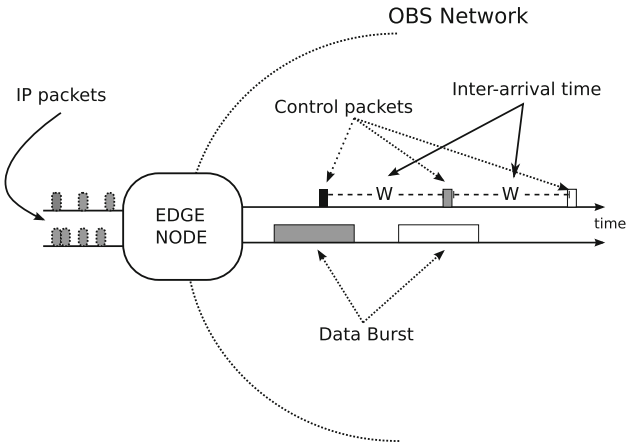

Fig. 8 Burst inter-arrival time (time window-based algorithm)

3.3.1 Computation of the probability of arrival inversion for time window-based assembly mechanisms

For time window-based assembly mechanisms, the burst inter-arrival time of each source–destination pair is constant and equals to W_i time units (Fig. 8). Let t_l^y be the arrival time of the last control packet belonging to the flow of source–destination pair (S_y, D_y) traversing the target core node and N the number of source–destination pairs. The probability of arrival inversion can be approximated considering the subset of source–destination pairs (S_z, D_z) whose control packets will arrive during the interval $[t_l^y; t_l^y + \Delta_{yz} - \overline{\beta}]$ besides having the target core node as an intermediate node. Thus, probability of arrival inversion is given by:

$$P_I = \frac{1}{N} \sum_{z=1}^{N} X_z \tag{3}$$

where

$$X_z = \begin{cases} 1, \text{ if } t_l^y \le t_l^z + W \le t_l^y + \Delta_{yz} - \overline{\beta}; \\ 0, \text{ otherwise} \end{cases}$$

and $\overline{\beta}$ represents the mean burst transmission duration.

3.3.2 Computation of probability of arrival inversion for volume-based assembly mechanisms

In volume-based assembly algorithms, a burst is assembled (and consequently the control packet released) only after the arrival of b_i bytes, which is equivalent to the arrival of n IP packets with mean size of B bytes; thus, $n = b_i / B$.

Let (S_y, D_y) be the source–destination pair for which the control packet is being processed at node x, and let t_l^y be the time at which the control packet arrived at node x. Inversion of arrivals occurs when the second control packet in the same

flow of pair (S_z, D_z) arrives during the interval $[t_l^y; t_l^y + \Delta_{yz} - \overline{\beta}]$.

For that to occur, the time spent assembling the burst associated with the second control packet should be equal to $[t_l^z; t_l^y + \Delta_{yz} - \overline{\beta}]$. In other words, it is necessary that n IP packets arrive at the assembly node of the pair (S_z, D_z) during the interval $[t_l^z; t_l^y + \Delta_{yz} - \overline{\beta}]$. Considering a Poisson arrival process of packets, this probability is as follows [23]:

$$P_I = \sum_{z=1}^{N} \frac{\lambda_z \left(\left[t_l^y - \left(t_l + z + \overline{\beta} \right) + \Delta_{yz} \right] \right)^n \mathrm{e}^{-\lambda_z \left(\left[t_l^y - (t_l^z + \overline{\beta}) + \Delta_{yz} \right] \right)}}{(n)!} \tag{4}$$

where λ_z is the mean arrival rate of IP packets of the pair (S_z, D_z).

4 The *least reusable channel* policy

The scheduling policies proposed so far (Sect. 2) try to minimize either the preceding void (LAUC, LAUC–VF and MIN–SV) or the succeeding void (MIN–EV). By adopting this type of strategy for scheduling bursts, the pattern of arrival of requests is ignored leading to under utilization and burst loss.

The LRC policy dynamically chooses the void for the allocation of a burst based on the potential reusability of existing voids for future requests. The void chosen is the one with the lowest chances reutilizing for future incoming bursts, which depends not only on the lifetime of the void created but also on the probability of arrival inversion to fill that void. The void reutilization function gives a chance that the voids created will be allocated to future requests. It is defined as:

$$\varphi(w) = P_I \times \upsilon(a_j) + (1 - P_I) \times \upsilon(p_j) \tag{5}$$

where P_I is the probability of inversion, and $\upsilon(a_j)$ and $\upsilon(p_j)$ give the useful lifetime of the preceding and of the succeeding voids, respectively.

The LRC is described in Algorithm 1. Let i be the node executing the LRC algorithm. The input of the algorithm is the set of output channels (W) and a reservation request for the period $[r_s, r_f]$.

Upon the arrival of a request, the algorithm determines the set W_v of channels capable of accommodating the request (the set of free wavelengths in the requested period) (line 1), and for each channel $w \in W_v$, the algorithm computes the reutilization function.

The probability of inversion is used to determine the chances that future requests will make reservations in a period prior to $[r_s, r_f]$. The computation of the probability of inversion considers only the set of source–destination pairs with packets which will visit node i. Thus, for each pair, node i have only to maintain information about the time of the arrival of the last control packet, which reduces the computational complexity of the algorithm.

After this, the algorithm computes the lifetime of the preceding void (a_w) and the succeeding one (p_w) which would have been created if channel w was chosen to schedule the request being processed (line 1).

The algorithm then computes the reutilization function for channel w (line 1), according to Eq. 5. This step determines the chances of channel w accommodating future requests. Thus, the algorithm chooses the channel w with the lowest probability of reutilization. If there are voids with the same value for reutilization, the rule is to choose the void with the smallest preceding void. Finally, the channel with the lowest chances of future reutilization will be used to allocate the request being processed (line 1).

The computational complexity of LRC is established by Theorem 1.

Theorem 1 *Let N_p be the number of source–destination pairs, W the set of wavelengths and S the set of resources already allocated in each channel. The computational complexity of LRC is $O(|N_p||W|log(|S|))$.*

Algorithm 1 LRC

INPUT
A set W of output channels of node i, a resource reservation request r to the interval $[r_s, r_f]$ with destination j.
OUTPUT
Reserved wavelength in the interval $[r_s, r_f]$.
LRC
1: Determine the set of channels $W_v \subseteq W$ capable of accommodating request r.
2: Determine the probability of inversion (P_I) according to the correspondent burst assembly algorithm.
3: **for all** ($w \in W_v$) determine: **do**
4: The lifetime of preceding and succeeding voids in relation to allocation of r in w (Eqs. 1, 2, respectively).
5: The reutilization function of w (Eq. 5).
6: Return w such that $\varphi(w)$ is the smallest.

Proof To determine the set (W_v) of channels that can be reserved for the request (line 1 and lines 3–5), it is necessary for the algorithm to preserve the same information preserved by LAUC–VF. When $W_v = W$, the time complexity is $O(|W| \log(|S|))$. Furthermore, the computation of the probability of inversion of arrivals (line 2) considers a subset of source–destination pairs, (N_p), resulting in a computational complexity of $O(|N_p||W| \log(|S|))$. □

Some remarks on the computational complexity of LRC are necessary. In the worst-case scenario, LRC will be highly complex as shown below. If N_n is the number of nodes of the network and N_p the number of source–destination pairs, $|N_p| = \frac{N_n(N_n-1)}{2}$, the computational complexity of LRC is

$O(N_n^2|W|\log(|S|))$. However, in operational networks, the network connectivity degree allows the use of traffic engineering techniques, such as load balancing and minimum interference routing [24,25]. The employment of these techniques can reduce the competition among source–destination pairs leading to a smaller number of source–destination pairs sharing the same links; this reduces the computational efforts demanded by LRC.

Table 1 summarizes the computational complexity of the algorithms compared in Sect. 5.

5 Performance evaluation

To evaluate the performance of the LRC policy, simulation was carried out using the OB2S simulator [26] and compared to the results of MIN–EV, RANDOM and LAUC–VF. The RANDOM policy chooses a void at random. Results for MIN–SV are not shown given their similarity to those of LAUC–VF. To generate a scenario where losses are more frequently, simulations were run at a very high load level, varying from 10 to 200 Erlangs.

The metrics used to compare the algorithms were the blocking probability, network utilization and the fairness factor, defined as the ratio between the number of longest blocked routes and the number of shortest blocked routes. For each simulation run, 200,000 requests were generated. Twenty replications for each scenario were generated to compute the confidence interval with confidence level of 95 %. Experiments were conducted using the topologies shown in Fig. 9. However, due to space limitations, results will be presented only for the topology in the Fig. 9a. Each link in the topologies has 32 wavelengths with 2.5 Gbps capacity. The processing time for the control packet was 50 μs.

Each network node is capable of full wavelength conversion, i.e., the scheduling algorithm can use any output channel to accommodate requests regardless of the input channel they arrived. Furthermore, each node can be either a source or a destination of traffic streams. A source–destination pair is randomly selected according to a uniform distribution when requests are generated. The route between the source nodes and the destination nodes follows the shortest path connecting these nodes.

For the time-based assembly algorithm, edge nodes are fed by traffic generated by a Poisson process, and the burst assembly time is set to 1ms. When the assembly algorithm is based on traffic volume, the burst threshold is 1,280 kb.

Table 1 Worst-case computational complexity of channel scheduling algorithms

LRC	RANDOM	MIN–EV	LAUC–VF
$O(\|N_p\|\|W\|\log(\|S\|))$	$O(\|W\|\log(\|S\|))$	$O(\log(\|S\|))$	$O(\|W\|\log(\|S\|))$

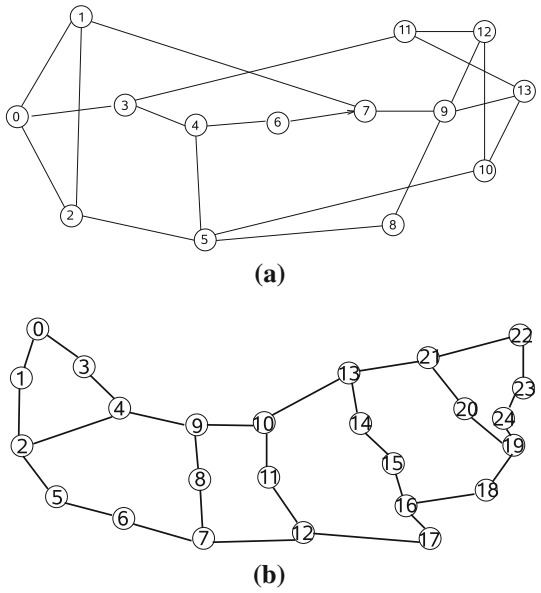

Fig. 9 Topologies used in the simulations. **a** *Backbone* NSFNet. **b** *Backbone* abilene

Figure 10 shows the blocking probability produced by the algorithms considering both time- and volume-based assembly. In both cases, the blocking probability increased as the network load increased as was expected. When the time-based assembly algorithm was employed by the ingress nodes (Fig. 10a), the highest blocking probability was produced by MIN–EV, followed by RANDOM. The LRC algorithm produced the lowest blocking probability of all the algorithms evaluated reducing by 13 % the blocking probability resulting from the use of LAUC–VF (second lowest blocking probability) and by 27.6 % for that of the MIN–EV algorithm.

Moreover, the results obtained when the assembly algorithm is based on traffic volume are similar. However, the blocking probability resulting from the use of all the algorithms was slightly higher when burst assembly was based on time. Similar results were found in Figueiredo et al. [27]. The reason is that the intensity of the burstiness of the traffic generated by the volume-based policy is greater than that produced by the time-based policy. It is the increase in resources demand by volume-based assembly policies, which leads to greater blocking probability values.

In comparison with MIN–EV, the difference in blocking probability decreased to 23, 16 % and in comparison with LAUC–VF, decreased to 9, 1 %. The difference when compared to the experiments using time-based assemblers is due to the more precise estimation of the inversion probability. Thus, the LRC algorithm can better evaluate the use of voids and make the best choice.

Figure 11 shows the network goodput as a function of the network load. Besides producing slightly higher goodput, LRC produced the lowest blocking probability of all the policies.

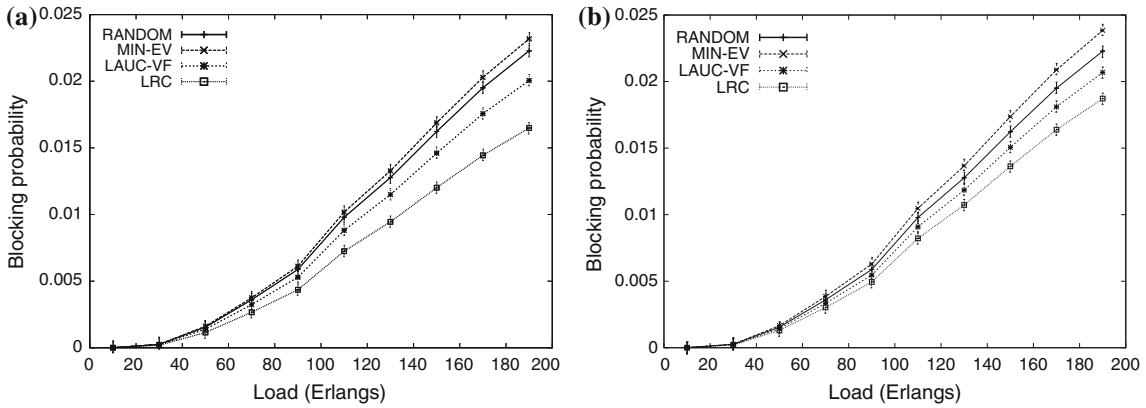

Fig. 10 Blocking probability. **a** Time window-based assembling. **b** Traffic volume-based assembling

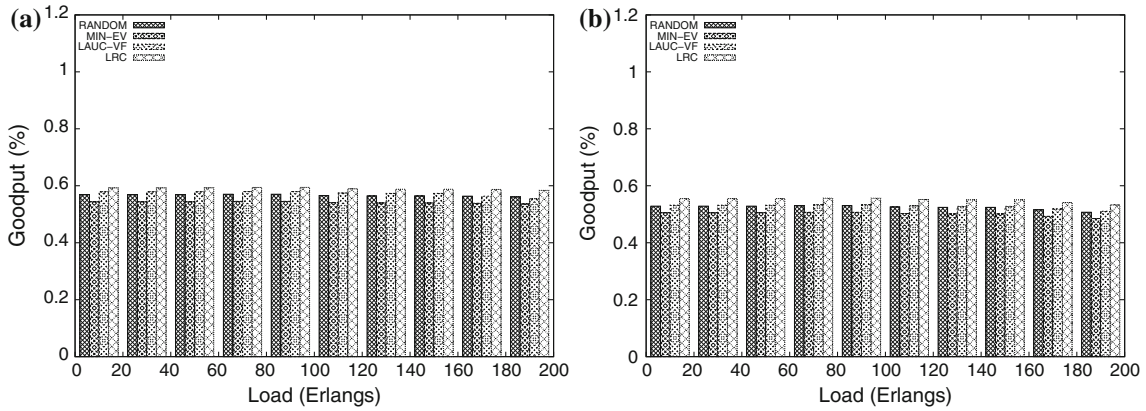

Fig. 11 Network goodput. **a** Time window-based assembly. **b** Traffic volume-based assembly

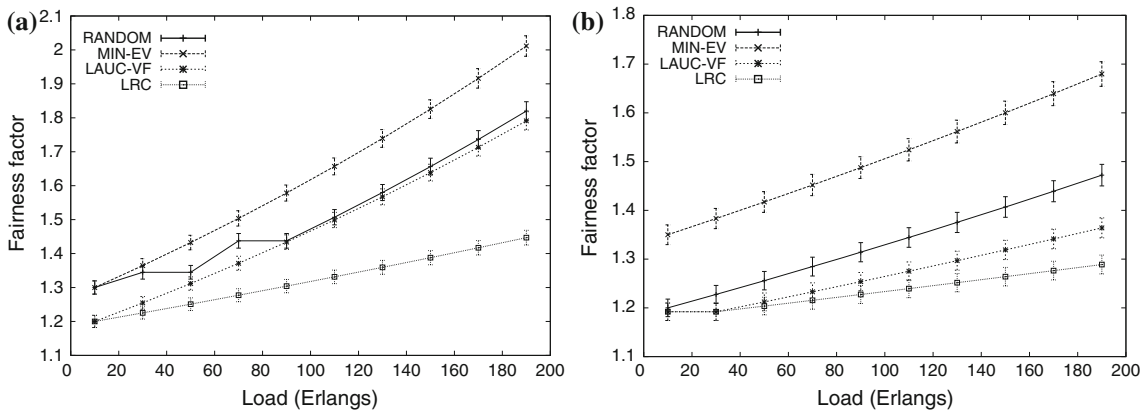

Fig. 12 Fairness factor. **a** Time window-based assembly. **b** Traffic volume-based assembly

Figure 12 shows the fairness factor as a function of the network load. Ideally, a scheduling algorithm should not favor requests based on the length of the route they take. Algorithms that prioritize a preceding void, however, do not consider the probability of inversion and consequently penalize short routes. On the other hand, algorithms that prioritize succeeding voids do not consider the arrival of requests with

Table 2 Average (maximum) number of source–destination pairs per node

Topology	30 nodes		50 nodes		NSF
Network density	Dense	Sparse	Dense	Sparse	–
Dijkstra	1.09 (2)	1.89 (3)	1.02 (2)	1.78 (2)	2.02 (3)
LMIR	0.33 (1)	0.75 (1)	0.27 (1)	0.68 (1)	0.82 (1)

Table 3 Relative gain in execution time

Topology	30 nodes		50 nodes		NSF
Network density	Dense	Sparse	Dense	Sparse	–
Reference value	3.11×10^{-3}	3.19×10^{-3}	3.08×10^{-3}	3.15×10^{-3}	3.44×10^{-3}
LRC (volume) (%)	0.0	0.0	0.0	0.0	0.0
LRC (time) (%)	0.03	0.07	0.02	0.09	0.02
RANDOM (%)	−1.72	1.98	1.73	1.99	2.04
MIN–EV (%)	1.41	1.30	1.25	1.42	1.54
LAUC–VF (%)	1.33	1.26	1.26	1.34	1.55

longer offset times and as a consequence, penalize longer routes. As can be seen in Fig. 12, the fairness factor increases as the network load increases, which means that longer routes will be penalized.

Since LRC adopts a dynamic strategy for choosing voids, it schedules bursts close to their destination (those with a small probability of inversion) on preceding voids. When scheduling requests that involve long paths, control packets being processed have large offset times and consequently a high probability of arrival inversion. LRC uses the succeeding void for these requests, leaving the preceding void for future requests. The LRC algorithm produced better results with less penalization for longer routes than other algorithms.

To evaluate the scalability of the proposed algorithm, the average number of connections traversing a given node was measured. For this, network topologies were created using Waxman's method [28,29]. In this method, the probability of the existence of link between u and v is given by

$$P(u, v) = \alpha e^{-d/(\beta L)},$$

where $0 < \alpha, \beta \leq 1$ are model parameters, d is the Euclidean distance between u and v, and L is the maximum Euclidean distance between any two nodes of the network. Topologies were classified into dense or sparse according to the number of edges. Dense topologies have approximately N^2 edges, whereas sparse topologies have approximately $N - 1$ edges, where N is the number of nodes.

Table 2 presents both the mean and maximum value obtained.[1] The routing schemes adopted were Dijkstra's algorithm and the light minimum interference algorithm (LMIR) [24]. In a general way, it can be seen that the number of flows going through a given node is highly dependent of the adopted routing strategy. Moreover, the higher is the network average node degree, the lower is the number of flows traversing the node, which is due the higher number of available paths between two nodes.

Table 3 shows the execution time of the evaluated algorithms. The reference value was the execution time of the slowest algorithm, which has 0 % of gain in the execution time. It can be seen that the slowest one was the LRC. However, two important aspects must be noticed. The first is that the average node degree, but not the topology impacts the performance; the larger the node degree, the higher the number of paths connecting any two nodes. As a consequence, the number of source–destination pairs traversing the node is smaller. The second important aspect is that LRC was outperformed by at most 2 % in the execution time, which shows that it has a good trade-off between blocking probability and execution time.

6 Conclusions

Scheduling is considered to be the major control function in OBS networks because it can significantly impact on burst loss. Since bursts have different offset time, they may arrive in an order other than that of their control packets, thus yielding fragmentation of the bandwidth of the output data channels of a core node. These void intervals can be used to accommodate the transmission of other incoming bursts. Scheduling policies differ in relation to the criteria used to allocate these voids.

In this paper, the LRC scheduling discipline has been proposed. It tries to maximize the reusability of voids for future bursts by considering route information, as well as load information. Comparison with the existing scheduling disciplines has shown that the LRC produced a lower blocking probability than do existing policies, and it distributed the loss of bursts more uniformly for routes with different lengths. The difference between the blocking probability of the LRC and

[1] The maximum value is presented between parenthesis.

the lowest resulting from other existing algorithms is 13 %. For future work, the derivation of expressions for determining the probability of arrival inversion considering the mixed time burst length algorithm and for traffic with long range dependencies [30,31] is recommended.

Acknowledgments This work was partially sponsored by CNPq and FAPESP.

References

[1] Qiao, C., Yoo, M.: Choices, features and issues in optical burst switching (OBS). Opt. Netw. Mag. **1**, 36–44 (2000)
[2] Xiong, Y., Vandenhoute, M., Cankaya, C.: Control architecture in optical burst-switched wdm networks. IEEE J. Sel. Areas Commun. **18**, 1838–1851 (2000)
[3] Turner, J.: Terabit bursts switching. J. High Speed Netw. **8**, 3–16 (1999)
[4] Murty, C., Gurusamy, M.: WDM Optical Networks: Concepts, Design and Algorithms. Prentice Hall, Upper Saddle River (2002)
[5] Yu, X., Li, J., Cao, X., Chen, Y., Qiao, C.: Traffic statistics and performance evaluation in optical burst switched networks. J. Lightwave Technol. **22**(12), 2722–2738 (2004)
[6] Kumar, R.K.P.P., Kumar, R.M.: Performance analysis of optical burst switching using burst delay feedback scheduling with different methods. In: International Conference on Computing, Communication and Applications, pp. 1–6 (2012)
[7] Ichikawa, H., Kamakura, K.: Dimensioning an scheduler buffer in OBS networks using forward resource reservation. In: International Conference on Computing, Networking and Communications (ICNC), pp. 282–286 (2012)
[8] Netak, L., Chowdhary, G., Suryawanshi, V., Borade, J.: Reverse scheduling approach for burst loss minimization in WDM OBS based networks. In: International Conference on Computer and Communication Technology (ICCCT), pp. 517–523 (2011)
[9] Wu, G., Zhan, T., Chen, J., Li, X., Qiao, C.: Design and implementation of an index-based parallel scheduler for optical burst switching networks. In: Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), pp. 1–3 (2011)
[10] Rogiest, W., De Turck, K., Laevens, K., Fiems, D., Bruneel, H., Wittevrongel, S.: Optimized channel and delay selection for contention resolution in optical networks. In: IEEE International Conference on Communications (ICC), pp. 1–5 (2011)
[11] Figueiredo, G., da Fonseca, N.L.S.: Algorithm with linear computational complexity for batch scheduling in OBS networks. In: IEEE International Conference on Communications, pp. 1–6 (2011)
[12] Zhou, B., Bassiouni, M. A.: Threshold-based preemption scheme for improving throughput in OBS networks. Photonic Netw. Commun. **24**, 1–10 (2011)
[13] Yang, X., Huang, S., Yan, H., Long, K.: A distributed backoff-channel deflection algorithm with load balancing for optical burst switching networks. Photonic Netw. Commun. 1–8 (2011)
[14] Xiong, Y., Vandenhoute, M., Cankaya, C.: Design and analysis of optical burst-switched networks. In: SPIE'99 Conference on All Optical Networking: Architecture, Control and Management Issues, vol. 3843, pp. 112–119 (1999)
[15] Chen, Y., Qiao, C., Xiang, Y.: Optical burst switching (OBS): a new area in optical networking research. IEEE Netw. **18**, 16–23 (2004)
[16] Xu, J., Qiao, C., li, J.: Efficient burst scheduling algorithms in optical burst-switched networks using geometric techniques. IEEE J. Sel Areas Commun. **22**, 1796–1811 (2004)
[17] Xu, J., Qiao, C., Li, J., Xu, G.: Efficient channel scheduling algorithms in optical burst switched networks. In: IEEE INFOCOM (2003)
[18] Wang, X., Morikawa, H., Aoyama, T.: Priority-based wavelength assignment algorithm for optical burst switched photonic networks. In: Optical Fiber Communications Conference, pp. 765–766 (2002)
[19] Li, J., Qiao, C.: Schedule bursts proactively for optical bursts switched networks. Comput. Netw. **44**, 617–629 (2004)
[20] Detti, A., Eramo, V., Listanti, M.: Performance evaluation of a new technique for IP support in a WDM optical network: optical composite burst switching (OCBS). J. Lightwave Technol. **20**, 154–165 (2002)
[21] Vokkarane, V.M., Jue, J.P.: Prioritized burst segmentation and composite burst-assembly techniques for QOS support in optical burst-switched networks. IEEE J. Sel. Areas Commun. **21**, 1198–1209 (2003)
[22] Chang, J., Park, C.: Efficient channel scheduling algorithm in optical burst switching architecture. In: IEEE Workshop on High Performance Switching and Routing, pp. 194–198 (2002)
[23] Papoulis, A.: Probability, Random Variables and Stochastic Process, chap. 3. McGraw-Hill, London (2002)
[24] Figueiredo, G.B., da Fonseca, N.L.S., Monteiro, J.A.S.: A minimum interference routing algorithm with reduced computational complexity. Comput. Netw. **50**, 1710–1732 (2006)
[25] Figueiredo, G.B., da Fonseca, N.L.S., Monteiro, J.A.S.: A minimum interference routing algorithm. IEEE Int. Conf. Commun. **4**, 1942–1947 (2004)
[26] Maranho, J., Soares, A., Giozza, W.F.: An architectural study of wavelength conversion in wdm networks with burst switching (in Portuguese). In: Proceedings of the Brazilian Symposium on Computer Networks (SBRC), pp. 133–146 (2007)
[27] Figueiredo, G., da Fonseca, N.L.S., Melo, C., Salvador, M.: On the transformation of multifractal traffic at ingress optical burst switches. In: IEEE International Conference on Communications (ICC '06), vol. 3, pp. 1040–1045 (2006)
[28] Zegura, E. W., Calvert, K. L., Bhattacharjee, S.: How to model an internetwork. In: IEEE INFOCOM, pp. 594–602 (1996)
[29] Waxman, B.M.: Routing of multipoint connections. IEEE J. Sel. Areas Commun. **6**(9), 1617–1622 (1988)
[30] Melo, C.A.V., da Fonseca, N.L.S.: Envelope process and computation of the equivalent bandwidth of multifractal flows. Comput. Netw. **48**, 351–375 (2005)
[31] da Fonseca, N.L.S., Mayor, G.S., Neto, C.A.V.: On the equivalent bandwidth of self-similar sources. ACM Trans Model. Comput. Simul. (TOMACS) **2**(10), 104–124 (2000)

Author Biographies

Gustavo B. Figueiredo received his B.Sc. degree in Computer Science from Salvador University (2001), and the M.Sc. (2003) and Ph.D. (2009) degrees in Computer Science from the State University of Campinas. Since 2010, he has been affiliated with the Department of Computer Science of the Federal University of Bahia, Bahia, Brazil, where he is currently a Associate Professor. His main research interest includes problems involving Network Performance Evaluation, Planning, Dimensioning and Optimization of Optical Burst Switched Networks.

Nelson L. S. da Fonseca received his Electrical Engineer (1984) and M.Sc. in Computer Science (1987) degrees from The Pontificial Catholic University at Rio de Janeiro, Brazil, and the M.Sc. (1993) and Ph.D. (1994) degrees in Computer Engineering from The University of Southern California, USA. Since 1995, he has been affiliated with the Institute of Computing of The State University of Campinas, Campinas, Brazil, where he is currently a Full Professor. Prof. Fonseca published over 250 refereed papers and supervised over 50 graduate students. He is the Editor-in-Chief of the IEEE Communications Surveys and Tutorials. He served as Editor-in-Chief for the IEEE Communications Society Electronic Newsletter and Editor of the Global Communications Newsletter. He is a member of the editorial board of: Computer Networks, IEEE Communications Magazine, Peer-to-Peer Networking and Applications, International Journal of Communication Systems and Journal of Internet Service and Applications. He served on the editorial board of the IEEE Transactions on Multimedia, Brazilian Journal of Computer Science and the Brazilian Journal on Telecommunications. He is the recipient of Elsevier Computer Networks Editor of the Year 2001. He served as ComSoc Director for Latin America. He served as ComSoc Director of On-line Services and served as technical chair for over 15 ComSoc symposia and workshops.