Photon Netw Commun
DOI 10.1007/s11107-015-0549-4

On traffic scaling transformation at ingress optical burst switches

Gustavo B. Figueiredo[1] · **Cesar A. V. Melo**[2] · **Nelson L. S. da Fonseca**[3]

Received: 25 January 2015 / Accepted: 6 August 2015

Abstract Ingress nodes in optical burst switching (OBS) networks are responsible for assembling burst from incoming packets and forwarding these bursts into the OBS network core. Changes in the statistical characteristics of a traffic stream at an ingress switch can affect the capacity of the network to provide quality of service. Therefore, the statistical characteristics of the output flow of an ingress node must be known for appropriate network dimensioning. This paper evaluates the impact of burst assembly mechanisms on the scaling properties of multifractal traffic flows. Results show that the factor most relevant in determining the nature of the output traffic flow is the relationship between the cut-off time scale of the input traffic and the time scale of assembly threshold. Moreover, a procedure for the detection of the cut-off scale of incoming traffic is introduced.

Keywords OBS networks · Multifractality · Burst assembly algorithms

✉ Nelson L. S. da Fonseca
nfonseca@ic.unicamp.br

Gustavo B. Figueiredo
gustavo@dcc.ufba.br

Cesar A. V. Melo
cavmelo@icomp.ufam.edu.br

[1] Department of Computer Science, Federal University of Bahia, Salvador, Brazil

[2] Institute of Computing, Federal University of Amazonas, Manaus, Brazil

[3] Institute of Computing, University of Campinas, Campinas, Brazil

1 Introduction

In optical burst switching (OBS) networks, packets are aggregated at the edge nodes of the network to form transmission units called bursts. A signalling packet is transmitted ahead of each burst to reserve resources for that burst; hence, the edge node does not need to wait for a set-up confirmation to start transmitting the burst. In fact, it transmits a burst some time after sending the set-up request message in the hope that the message will reach the destination. Should the set-up fail, the burst will be discarded at the node where reservation was not possible.

OBS networks are potentially able to cope with bursty traffic in a cost-effective way. However, their performance depends on the actual traffic carried. Poorly dimensioned networks involve a high blocking probability. Therefore, it is of paramount importance to understand the characteristics of the traffic injected into the core of an OBS network by an ingress node.

Previous research [1–3] has investigated the interaction between monofractal traffic and burst assembly algorithms employed in OBS networks. IP flows can be modelled by either a monofractal or multifractal processes, depending on the specific network characteristics. In other words, some IP flows can be modelled by monofractal processes, while others call for more complex multifractal modelling. There is no definite model that should be used for all network scenarios [4–7]. The best model can only be identified by measuring the characteristics of a specific flow.

The employment of a monofractal model to represent a multifractal traffic stream can result in the underutilization of network resources since it tends to overestimate multifractal behaviour. Such overestimation is due to the use of a single parameter, the Hurst parameter, to describe burstiness of the traffic. This parameter, which is a constant value, is a global

Published online: 20 August 2015

measure of explosiveness and is unable to capture the activity on small time scales of multifractal traffic [4,5].

An accurate description of the burstiness of a traffic stream is essential for the characterization of the traffic, since this indicates the demands of resources to support that stream. Moreover, long-range dependencies influence the duration of contention periods, which can lead to high blocking probability values. Thus, understanding the nature of changes in scaling is most relevant for the proper dimensioning of OBS networks.

This paper addresses the issue of the transformation of scaling traffic at ingress nodes of OBS networks. It sheds some light on whether the scaling characteristics of an incoming traffic flow is maintained in the outgoing flow of an ingress node. Policies based on time and on the amount of traffic are considered in the present study. Simulation using real network traffic was carried out, and the results indicate that the nature of the outgoing flow of an ingress node depends on the relationship between the cut-off time scale of the incoming traffic and the time scale of the assembly process. Moreover, a procedure was introduced to identify the cut-off scale, a procedure, which allows the automatic identification of the traffic parameters necessary for burst assembly.

The rest of this paper is organized as follows. Section 2 discusses the relationship between the burst assembly process and traffic scale changes. Section 3 provides some concepts involved in traffic scaling. Section 4 presents a method for automatic detection of the cut-off time scale. Section 5 presents burst assembly policies aimed at promoting traffic scale changes. Section 6 presents numerical examples of the proposed burst assembly policies, and finally, Section 7 concludes the paper.

2 Analysis of the relation between traffic scaling changes and the burst assembly process

Whenever a traffic stream goes through a network node, the statistical characteristics of that stream may change due to buffering and traffic aggregation. In OBS networks, burst assembly mechanisms at ingress nodes are responsible for the transformation of the statistics of the incoming traffic. In order to verify the occurrence of such transformations, the relationship between the burst assembly process and the changes in the traffic statistics was investigated.

2.1 Burst assembly process in OBS networks

An ingress OBS node contains a set of queues to store packets incoming from its input ports, with the number of queues depending on the burst assembly criteria adopted. Figure 1 exemplifies the architecture of a basic OBS node with a two-queue ingress node.

The burst assembly mechanism at the ingress node monitors these queues, and whenever the assembly criterion is satisfied, a burst is assembled and scheduled for transmission. For each burst assembled, a reservation request packet (control packet) specifies the source, the destination, the duration and the wavelength on which the burst should be transmitted. Once the burst is scheduled for transmission, this request packet is transmitted and, some time later, the actual burst is transmitted.

Various criteria have been defined for burst assembly [1, 8–11], mostly based on either time [1] or byte counting [10]. In a time-based algorithm, once the first packet arrives in a queue i, a queue timer starts ticking. When the timer reaches a predefined threshold value t_i, a burst is created by gathering all the packets stored in that queue, and the burst is scheduled for transmission as discussed above. Typical assembly threshold values for mechanisms based on time vary from 1 [12]–600 ms [13] (Table 1).

Byte counting-based algorithms employ a byte counter, b_i at each input queue i, and this counter is updated as packets are added or removed from the queue. Whenever a predefined threshold for the amount of traffic is reached, a burst is created and scheduled for transmission.

In operational networks, the threshold for burst assembly depends on the delay requirement of the associated class of service (CoS). The stricter the requirements, the shorter is threshold for the burst assembly.

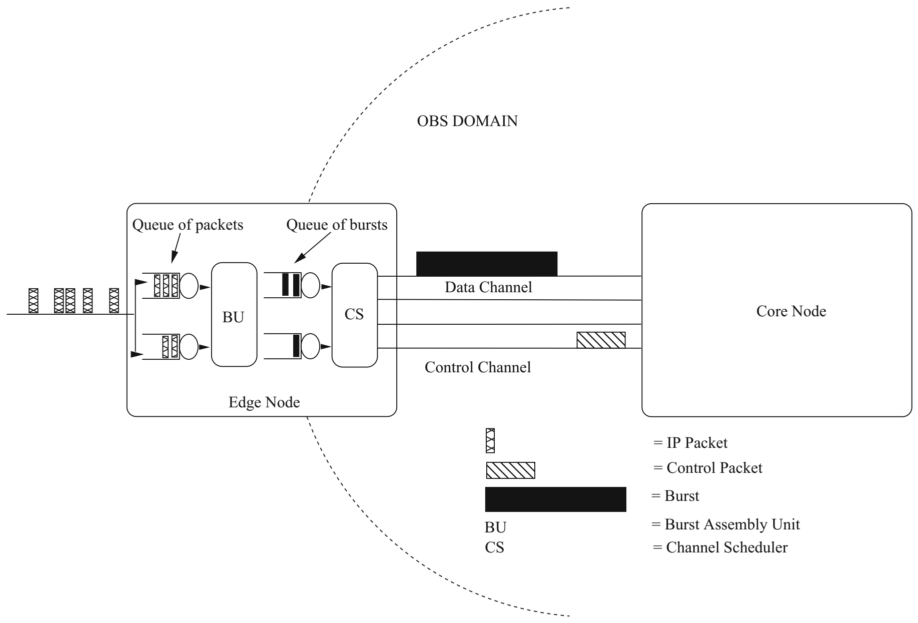
Criteria based on service classes have also been proposed. In [13], the criteria depend on the CoS, the maximum burst size and the maximum burst assembly time (i.e. maximum queuing delay for the head-of-the-line packet). In the proposal presented in [14], packets are gathered in bursts according to a descending order of priority with priority criteria depending on the number of CoS, the maximum number of packets of each CoS that can be included in each burst, and the maximum assembly time for bursts.

2.2 Assessment of the impact of burst assembly parameters on traffic scaling network traffic scaling

The aggregation of IP packets done by burst assembly algorithms employed at edge nodes can change the time scale on which those packets are transmitted. To evaluate these changes, a careful analysis was conducted, and all the results obtained are shown in this section.

Simulation were conducted using the Network Simulator 2 (NS-2) to evaluate the changes in the statistical characteristics of the traffic passing through an ingress OBS node. The OBS module defined in [15] was used in the simulations. After that, the set of tools proposed in [16] was used to analyse the resulting traffic. The following sections describe the methodology employed and some findings regarding traffic scale changes.

Fig. 1 An example of OBS network architecture



2.2.1 The methodology for traffic scaling evaluation

Since the seminal work of Leland et al. [17], various studies have shown that network traffic is characterized by scale invariance, or scaling, which means that there is no specific time scale the "burstiness" of a traffic stream can be characterized. Hence, an accurate description of the network traffic must account for a variety of time scales. Such traffic is characterized by long-range dependency (LRD), which implies that the auto-correlation of the traffic decays very slowly, or hyperbolically; moreover, this auto-correlation is non-summable across the different time scales. Most network traffic is characterized by LRD, which impacts critically on the dimensioning and performance of queues, since packet loss does not decrease substantially with an increase in buffer size. Various types of network traffic can be modelled by self-similar or (mono) fractal processes such as local area network traffic and some wide area network traffic. Scaling of such monofractal traffic is measured by a single constant value: the Hurst parameter.

Multifractal (multiscaling) processes are also used for modelling network traffic. These processes have richer scaling behaviour which are associated with non-uniform local variability, i.e these processes exhibit nonlinear behaviour at different moments. In addition to long-range dependencies, multifractal traffic has a high level of activity on small time scales, which differs significantly from that of monofractal traffic with the same scales. The burstiness on small scales diverges from that on larger scales. In these processes, the local regularity of sample paths can usually be described by the Holder exponent function, a generalization of the Hurst parameter. Moreover, the incremental process cannot be described by a gaussian distribution as in monoscaling traffic.

Some wide area network traffic can be modelled as multifractal traffic as will be shown in the next section. The use of a monoscaling process [18] to model such multiscale traffic would lead to misleading results since such a model tends to overestimate the bandwidth needed by the traffic considerably. IP traffic will present different behaviours depending on the magnitude of the time scale on which it is observed. On large time scales (above hundred of milliseconds), the IP traffic presents self-similar behaviour which can be accurately modelled by monofractal process [16]. On small time scales, the traffic can present multiscaling behaviour that is not well captured by the monofractal process, and must be modelled more precisely by multifractal process.

The scale Δ^*, which separates those two regimes, is called the cut-off time scale [19]. Its identification plays an important role in the traffic modelling process, since it allows the determination of a range of scales in which the traffic will behave in either a monofractal or multifractal way. It can thus be viewed as an additional parameter help in the scaling characterization of network traffic, as will be discussed bellow.

Table 1 Typical threshold values for Burst assembly [13]

Class of service	Minimum size (KB)	Maximum size (KB)	t_i (ms)
EF	5	5	4.8
AF	30	50	55
BE	125	125	600

The tools used for the traffic analysis [16] are based on wavelet transformation, and they have been largely employed on the literature to characterize the aforementioned traffic regimes.

Such tools are based on the fact that any self-similar (or monofractal) q order process $\overline{X}(t)$ has statistical moments defined by [16]:

$$E|X(t)|^q = E|X(1)|^q.|t|^{qH} \tag{1}$$

where H is the Hurst parameter. This structure restricts the burstiness of a process to a uniform pattern across different time scales.

On the other hand, the statistical moments of a multiscaling (multifractal) process are defined as [16]:

$$E|X(t)|^q = E|X(1)|^q.|t|^{\zeta(q)} \tag{2}$$

where $\zeta(q)$ is the scaling function.

In the wavelet domain, the relationship established in Eq. 2 is defined as:

$$E|d_X(j,k)|^q \approx 2^{j\zeta(q)} \quad j \to -\infty \tag{3}$$

where $d_X(j,k)$ is the series of increments (details) obtained by the decomposition of the process $X(t)$ using the discrete wavelet transform. The scaling function, $\zeta(q)$, is defined as:

$$\zeta(q) = \alpha_q - \frac{q}{2} \tag{4}$$

where α_q is the scaling exponent, which has its value bound by the burstiness of the traffic. For multifractal processes, it varies at different statistical moments (q).

Abry et al. [16] describe a method called Multiscale Diagram (MD) used to determine the occurrence of multifractalty in a process. This method consists of verifying the behaviour of the function $\zeta(q)$ at different moments. A multifractal process is characterized by a nonlinear function, whereas a monofractal one reveals linear behaviour.

Estimating the values of the function $\zeta(q)$ requires the determination of the scaling exponent α_q, as defined in Eq. 4. The Logscale Diagram (LD) method is used to determine α_q, which is defined as the inclination of the curve that is close to the curve generated by the relation between μ_j and 2^j on a logarithmic scale. The value of μ_j is given by:

$$\mu_j = \frac{1}{n_j}\sum_{k=1}^{n_j} |d_X(j,k)|^q \approx E|d_X(j,k)|^q \tag{5}$$

where n_j is the number of details $d_X(j,.)$, on time scale j, generated by the decomposition of $X(t)$ using discrete wavelet transform.

Multifractality can also be detected by using a Linear Multiscale Diagram (LMD) method, which plots $h_q = \zeta_q/q$ against q. In such a diagram, monofractal behaviour is revealed by a horizontal alignment, whereas multifractal behaviour is reflected by a non-horizontal alignment.

Veitch et al. [20] showed that the information in various papers in the literature is misleading due to the overstatement of the multiscaling nature of some of the traffic, resulting from the misuse of tools for identifying scaling and the misidentification of the proper scale for analysis. To avoid such overestimation, in this paper we have carefully followed the steps proposed in [20], which are as follows:

1. Define the proper process based on an analysis of the traffic traces.
2. Set up the signal analysis tool accordingly.
3. Determine the relevant time scales.

Four processes are considered for the calculation of the discrete wavelet transform by Veitch et al. in [20] although the arrival time point process, $X(t)$, is the most studied in the literature. Therefore, this was the one selected for the signal analysis tool. The byte arrival process, $W(t)$, has been mentioned as an alternative process, since studies have shown results similar to those given by the arrival point process. In the present study, we define and analyse the byte arrival process from the evaluated traffic traces.

A signal analysis tool based on wavelet discrete transform was used to carry the studies presented in [20]. Aware of the limitations of this tool, essentially the computation of the confidence interval in a non-Gaussian context, in [20], the authors have emphasized the fine tunings necessary to obtain precise results. These include (1) turning on the non-Gaussian signal option since traffic in small time scale does not, by definition, show Gaussianity and (2) modifying the hard codes of the analysis tools the confidence interval can be estimated from the signal. These tunings have been implemented in the mentioned tool.

Finally, relevant time scales of the traffic traces have been identified by [20]. Three time scales have been considered to be relevant in [20]: the j^{IAT}, j^{**}, and j^{*}. The first, the "Inter-Arrival Time" scale, identifies the scale of isolation of individual packets. The second, the "breakup" scale, defines the start of scaling region, and the third, the "biscaling knee" scale, defines the possible regime changes, with multifractal on the left and monofractal on the right. In the present study, all three scales were considered (Table 2).

Based on the aforementioned guidelines, the multiscaling behaviour was investigated for all the traces considered in the present paper. Figure 2 summarizes the analysis for trace BWY-1069762448, and Table 2 summarizes the information contained in these traces specially the values for the time scales j^{IAT}, j^{**}, and j^{*}.

Table 2 Traces used in the investigation [21]

Trace	Date	Start time	Duration (s)	Rate (MBps)	j^{IAT} (ms)	j^{**}	j^{*}
ANL-1111548257	03/22/05	20:11	90	2.30	1 (0.091342)	1	4
BWY-1069762448	11/25/03	04:49	90	0.08	4 (0.441773)	4	9
IPLS-CLEV-090000-0	14/08/02	09:00	90	412.131	1 (0.014292)	1	8
IPLS-CLEV-090000-1	14/08/02	09:00	90	457.852	3 (0.012235)	3	8
IPLS-CLEV-091000-0	14/08/02	09:10	90	363.754	1 (0.039280)	1	6
MEM-1111247410	03/19/05	07:56	90	2.63	1 (0.082378)	1	5
MEM-1111679715	22/03/05	14:10	90	4.42	5 (0.014003)	5	9
MEM-1112013766	03/28/05	04:49	90	1.87	1 (0.056090)	1	5
MEM-1053844177	05/24/03	23:54	90	2.65	2 (0.097299)	2	6
TXS-1113503155	04/14/05	11:31	90	6.8	4 (0.079272)	1	4
20040601-193121-0	06/01/04	19:00	90	770.00	3 (0.009230)	3	7
20040601-193121-1	06/01/04	19:00	90	1.648	3 (0.004532)	3	8
20040601-194000-1	06/01/04	20:00	90	828.616	5 (0.007238)	5	9

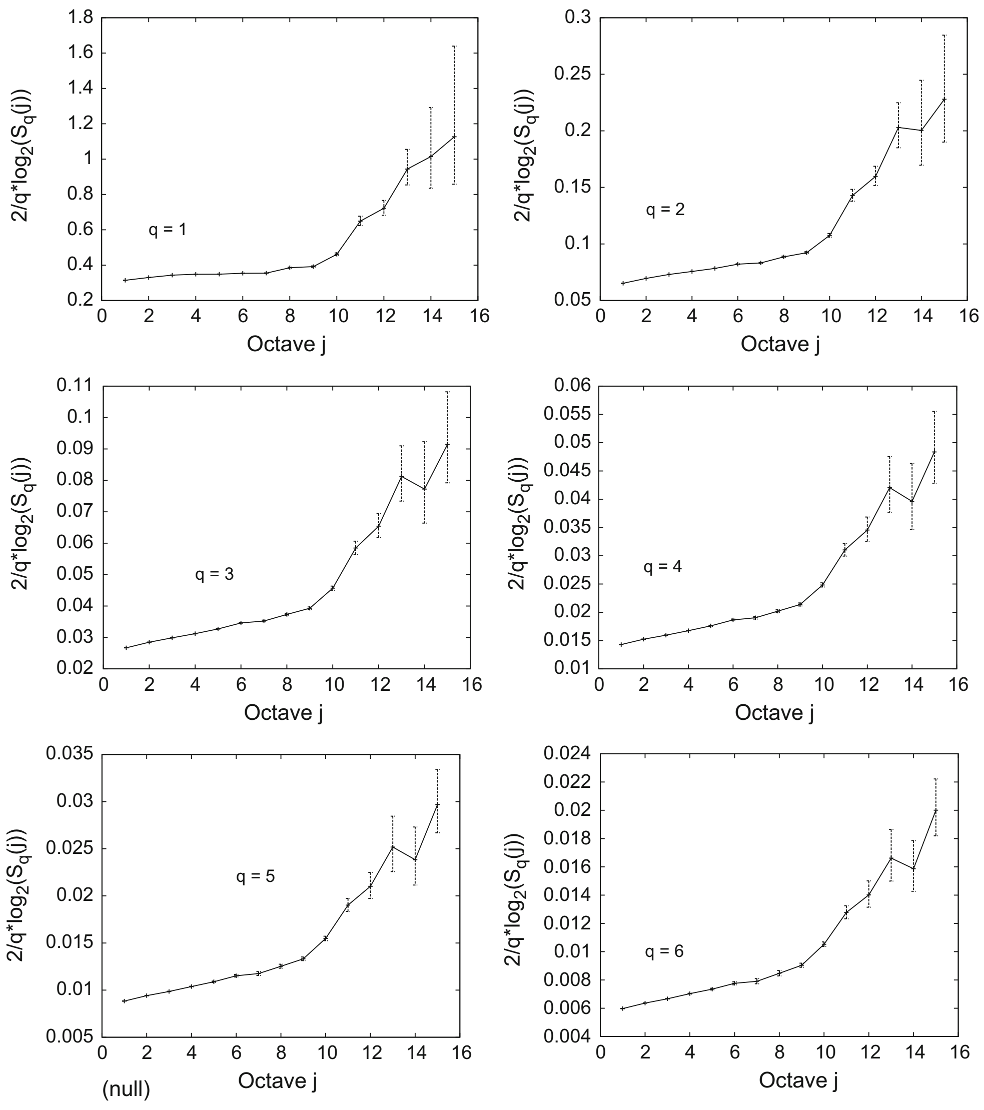


Fig. 2 qth order Logscale Diagrams for UDP flow of trace BWY-1069762448

Table 3 Scaling characteristics of real traffic traces used in the paper

Trace	Δ^* (ms)	$t_i > \Delta^*$ (ms)	$t_i < \Delta^*$ (ms)	Mean value of Holder exponent	Var	C. I.
ANL-1111548257	2.7	3	1	0.726	0.03	(0.719, 0.733)
BWY-1069762448	250	400	100	0.681	0.0023	(0.678,0.683)
MEM-1111247410	3.3	4	1	0.695	0.0085	(0.685, 0.705)
MEM-1111679715	5.4	6	3	0.758	0.008	(0.727, 0.789)
MEM-1112013766	3.0	4	1	0.72	0.009	(0.715, 0.725)
MEM-1053844177	6.7	7	6	0.687	0.007	(0.684, 0.692)
TXS-1113503155	1.3	2	1	0.89	0.0408	(0.86, 0.92)
IPLS-CLEV-090000-0	3	4	1	0.83	0.0400	(0. 80, 0.86)
IPLS-CLEV-090000-1	2.8	4	1	0.792	0.007	(0.791, 0.793)
IPLS-CLEV-091000-0	3.2	4	1	0.675	0.001	(0.672, 0.678)
20040601-193121-0	1.6	2	1	0.683	0.0014	(0.674, 0.692)
20040601-193121-1	1.3	2	1	0.689	0.0021	(0.676, 0.702)
20040601-194000-1	1.3	2	1	0.622	0.0027	(0.621, 0.623)

Multiscaling involves the manifestation of a single underlying scaling phenomenon, represented by a number of scaling exponents $\{\alpha_q\}$, which constitute the slope of the straight lines in the q-LD (LD) for the same range of scales. Hence, a signal involving multiscaling provides evidence of multifractality [20]. Looking for these evidences, in Fig. 2, we show the Logscale Diagrams $q = [1, 2, 3, 4, 5, 6]$ have twin scaling regimes: at fine scales, $[j^{**}, j^*] = [4, 9]$ and coarse scales, $[j^{**}, j^*] = [9, 15]$. Therefore, the data exhibit multiscaling in each scale range.

Since the biscaling identified in the LD analysis means there are twin scaling regimes, we must study the scaling properties on the identified scale ranges defined by the *breakup* and *knee* scales. We thus plot the LMD of the traces in Table 2. For coarse (large) scales, a clearly horizontal h_q functions emerges, suggesting the adequacy of a single value to describe it. Monofractal models should be used to represent the traffic dynamics over such scales. On the other hand, for fine (small) scales, all the h_qs values reveal a non-horizontal shape suggests that multifractal models will be required.

The Holder exponents for the traces used were computed according to the procedure in [22]. Table 3 displays the mean and the variance of this exponent, as well as the confidence interval used to compute the mean value. These values will be compared with those of the output traffic.

2.2.2 Evidences of the impact of burst assembly parameters on traffic scaling changes

The next two subsections evaluate the influence of time threshold values and the byte counter threshold on the scaling properties of the outgoing traffic. Due to space limitations, results will be presented only for the trace BWY-1069762448.

2.2.2.1 The impact of time threshold values on the time scale of the outgoing traffic Two scenarios are presented in this section to verify the relationship between the time threshold value and the cut-off time scale of the input traffic. In the first scenario, t_i is greater than the cut-off time scale value (Δ^*) of the input traffic traces. In the second scenario, t_i is smaller.

Figure 3 shows the MD and the LMDs for the output traffic when $t_i > \Delta^*$ resulting from the transformation of the input traffic in the trace BWY-1069762448. The linear behaviour of the curves in the MD indicates that the output traffic is monofractal since the cascading function $\zeta(q)$ reveals a linear behaviour at various statistical moments q. This can be confirmed by the LMD, which shows a horizontal alignment for the two traces. Therefore, assembling bursts on time scales greater than the cut-off time scale of a multifractal input traffic changes the scaling characteristics from multifractal to monofractal. The second column of Table 5 shows the Hurst parameter (H_t) of the monofractal output traffic when time-based policies are employed.

Figure 4 shows the results for $t_i < \Delta^*$. The curves of this MD do not present a linear behaviour indicating multifractality. Multiscaling can also be perceived in the curves of the LMD. None of them has horizontal alignment. Therefore, assembling bursts on time scales smaller than the cut-off scale maintains multiscaling characteristics.

2.2.2.2 The impact of the amount of traffic on the nature of the scaling of the output traffic As shown in the previous section, the nature of the output traffic at an ingress node depends on the relationship between the time threshold for burst assembly and the cut-off time scale of the input traffic. However, since most assembly algorithms use a byte counter as a trigger for assembling bursts, the relationship between

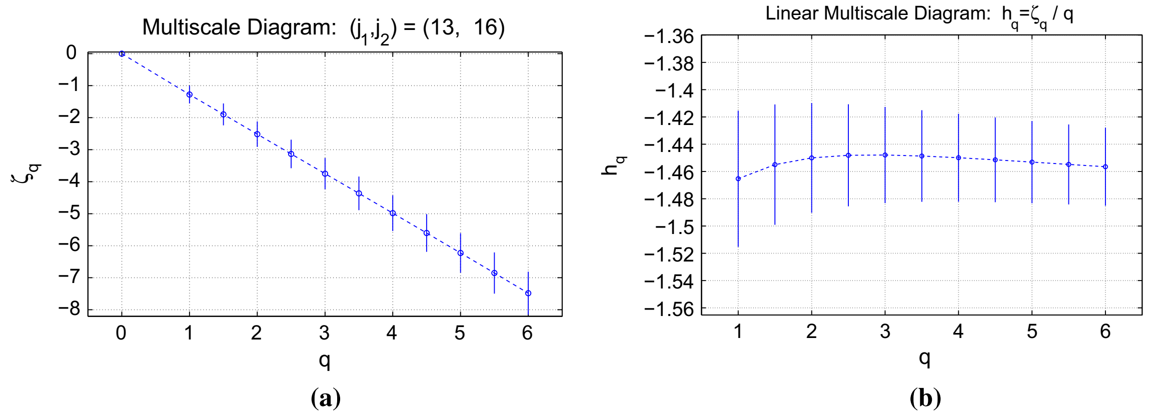


Fig. 3 Trace BWY-1069762448 analysis with $t_i > \Delta^*$, **a** Multiscale Diagram of the output traffic, **b** Linear Multiscale Diagram of the output traffic

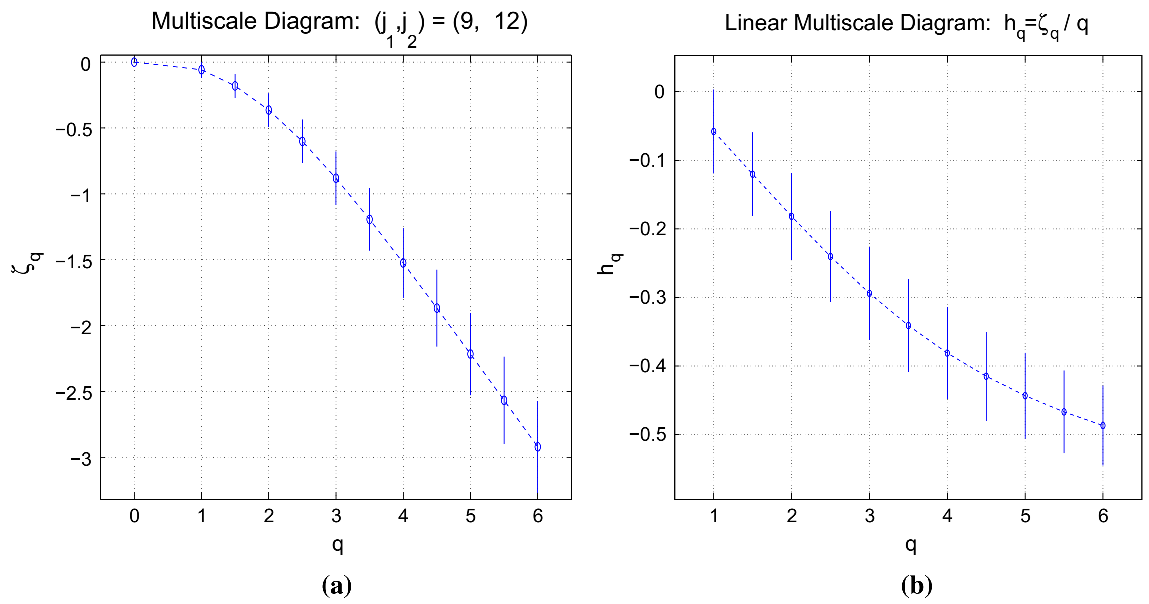


Fig. 4 Trace BWY-1069762448 analysis with $t_i < \Delta^*$, **a** Multiscale Diagram of the transformed output traffic, **b** Linear Multiscale Diagram of the output traffic

the number of bytes (b_i) for the threshold and the cut-off time scale was also investigated.

To investigate a potential relationship between the byte counter and its time cut-off scale, the byte counter was divided by the mean arrival rate so that the corresponding time scale could be identified. This time scale was then compared to that of the cut-off scale of the input traffic.

Investigations were conducted setting thresholds that led to assembly time scale smaller than the cut-off time scale as well as larger than that. The chosen threshold values, 1 and 125 KB, were obtained from previous research [12,13].

The trace BWY-1069762448 has a mean arrival rate (λ) of 0.0829944 Mbps, which results in assembly time scales (t_i) of 150 ms when a byte counter threshold value of 1 KB is used.

This value is below the input traffic cut-off time scale Δ^*. Moreover, the byte counter threshold of 125 KB gives assembly time scales of 12 s, which is above the cut-off time scale of the trace.

Figure 5 shows the results of the analysis of the traffic resulting from the burst assembly process with $b_i/\lambda > \Delta^*$. The MDs show similar behaviours, with the cascading function $\zeta(q)$ presenting a linear behaviour. Monofractality is confirmed by the horizontal alignment of the LMD. This linear behaviour of the cascading function indicates the occurrence of monofractal nature in the traffic.

Thus, assembling bursts with byte counter threshold greater than the product between the input flow mean arrival rate and its cut-off time scale transforms multifractal traffic into monofractal traffic. The third column of Table 5 shows

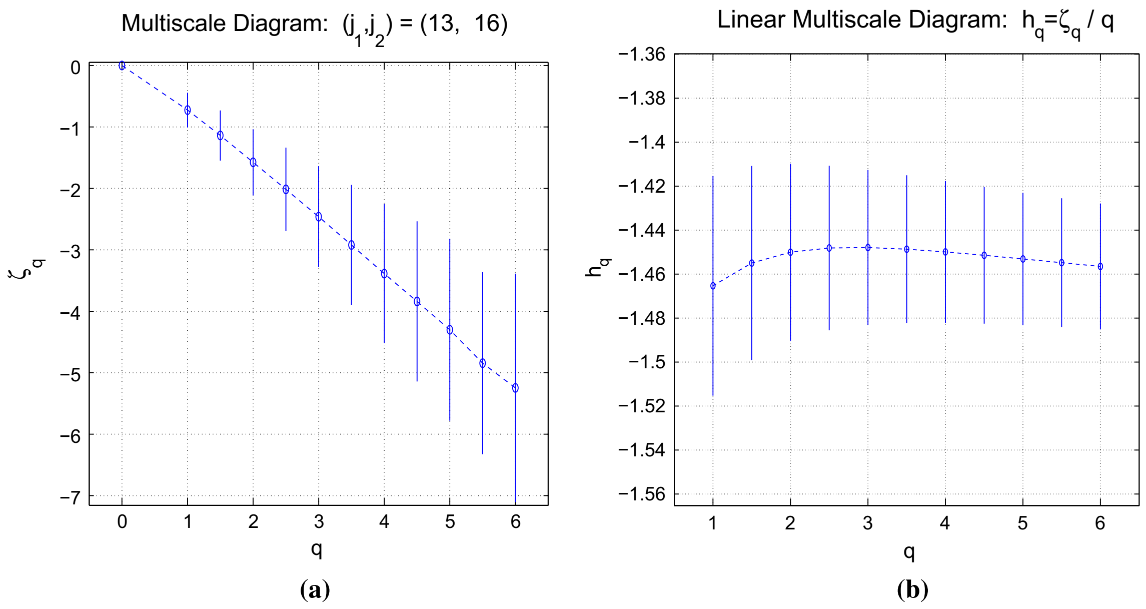


Fig. 5 Trace BWY-1069762448 analysis with $b_i/\lambda > \Delta^*$, **a** Multiscale Diagram of the output traffic, **b** Multiscale Diagram of original input traffic

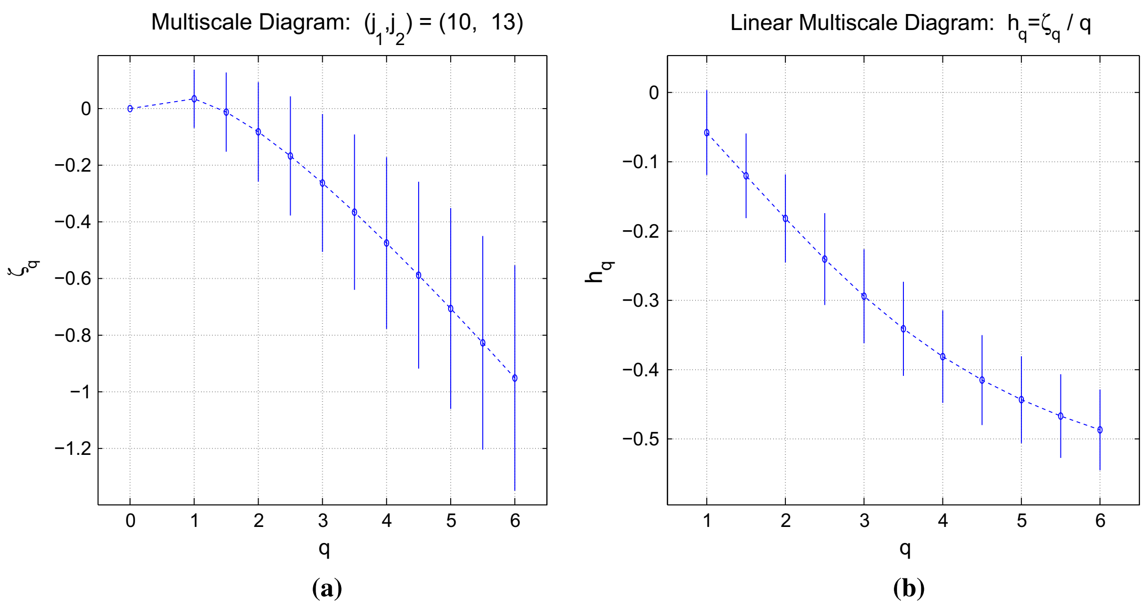


Fig. 6 Trace BWY-1069762448 analysis with $b_i/\lambda < \Delta^*$, **a** Multiscale Diagram of the transformed output traffic, **b** Multiscale Diagram of original input traffic

the Hurst parameter (H_v) for outgoing monofractal traffic when byte counting-based policies are employed.

Figure 6 provides an analysis of the traffic resulting from a burst assembly with $b_i/\lambda < \Delta^*$. Again, in the MD, the absence of linear behaviour indicates the presence of multifractality in the traffic. Moreover, the LMDs show a non-horizontal alignment.

2.2.3 The smoothing effect of burst assembly policies

In order to evaluate the smoothing effect of different assembly policies, the scaling characteristics of the output processes produced by these policies are compared.

Table 4 presents the mean, the variance of the Holder exponent values of multifractal output traffic. The subscripts

Table 4 Holder exponent of multifractal output traffic

Trace	Mean_t	Var_t	C.I._t	Mean_v	Var_v	C.I._v
ANL-1111548257	0.546	0.008	(0.544, 0.548)	0.603	0.016	(0.583, 0.623)
BWY-1069762448	0.581	0.003	(0.578, 0.584)	0.593	0.008	(0.585, 0.601)
IPLS-CLEV-090000-0	0.752	0.012	(0.743, 0.761)	0.774	0.008	(0.759, 0.789)
IPLS-CLEV-090000-1	0.743	0.010	(0.730, 0.756)	0.749	0.008	(0.739, 0.759)
IPLS-CLEV-091000-0	0.663	0.010	(0.658, 0.668)	0.665	0.008	(0.660, 0.670)
MEM-1111247410	0.601	0.006	(0.592,0.61)	0.653	0.002	(0.652, 0.654)
MEM-1111679715	0.598	0.008	(0.587, 0.609)	0.632	0.004	(0.63, 0.634)
MEM-1112013766	0.601	0.009	(0.598, 0.604)	0.622	0.004	(0.62, 0.624)
TXS-1113503155	0.655	0.010	(0.642,0.668)	0.732	0.008	(0.717, 0.745)
20040601-193121-0	0.653	0.010	(0.647, 0.659)	0.661	0.008	(0.655, 0.667)
20040601-193121-1	0.672	0.010	(0.659, 0.684)	0.674	0.008	(0.661, 0.687)
20040601-194000-1	0.589	0.010	(0.572, 0.606)	0.609	0.008	(0.596, 0.524)

Table 5 Hurst parameter of the monofractal output traffic

Trace	H_t	H_v
ANL-1111548257	0.598	0.639
BWY-1069762448	0.551	0.562
IPLS-CLEV-090000-0	0.722	0.747
IPLS-CLEV-090000-1	0.698	0.717
IPLS-CLEV-091000-0	0.623	0.629
MEM-1111247410	0.669	0.682
MEM-1111679715	0.595	0.687
MEM-1112013766	0.629	0.675
TXS-1113503155	0.672	0.797
20040601-193121-0	0.645	0.658
20040601-193121-1	0.672	0.680
20040601-194000-1	0.586	0.601

t and v denote time-based policies and byte counting-based policies, respectively. When comparing to the mean and to the variance of the input traffic, it is clear that the assembly process smoothes the input traffic. Both the mean and the variance of the Holder exponent values decreased. Lower mean values imply that the burstiness on small scales decreased. Moreover, lower variance values imply that the variability of the burstiness on these scales also decreased. Comparing the two different policies, it can be seen that the mean value of the Holder exponent of the traffic produced by time-based policies is lower than the traffic produced by byte counting-based policies, which indicates smoother activities on small time scales.

Table 5 provides the Hurst parameter value of the output traffic generated by the assembly policy based on time and that based on byte counter. The Hurst parameter was computed using the A–V estimator available at [23]. The threshold for the time-based policy was computed as the ratio between the byte counter threshold and the mean arrival rate so that both policies assemble burst on the same time scale.

It can be seen that policies based on byte counting produce monofractal traffic with higher Hurst parameter values than do the policies based on time. An explanation for those higher values is that policies based on byte counting produce longer bursts than do policies based on time. Under policies based on time, some packets are carried in different bursts, whereas under policies based on byte counting, these packet are carried in the same burst. Consequently, longer periods of activity and silence were produced by policies based on byte counting, leading to stronger long-range dependencies.

2.2.4 *Impact of traffic changes in the network dimensioning*

In the previous subsections, the impact of the burst assembly process on the statistical properties of the carried traffic was shown . Another important but complementary question that arises is: *What is the impact of such changes on network resource dimensioning?*

To answer this question, simulations were carried out. The objective of this simulation is to show how traffic changes could impact on the network dimensioning. To do that, this impact in terms of blocking probability was measured. The scenario used in the simulations is depicted in Fig. 7. The figure shows an OBS network composed by three edge nodes (two ingress nodes and one egress node) and one core node. The idea is to have a simple topology in which the phenomenon of interest can be isolated, eliminating the occurence of losses due to bad routing strategies or different channel scheduling strategies [24].

The distance between nodes is 1000 Km, and each link connecting the network nodes has a single fibre with 16 data channels and two control channels operating at 10 Gbps. The channel scheduling algorithm used by the core node was the LAUC-VF (Latest Available Unused Channel with Void Fill-

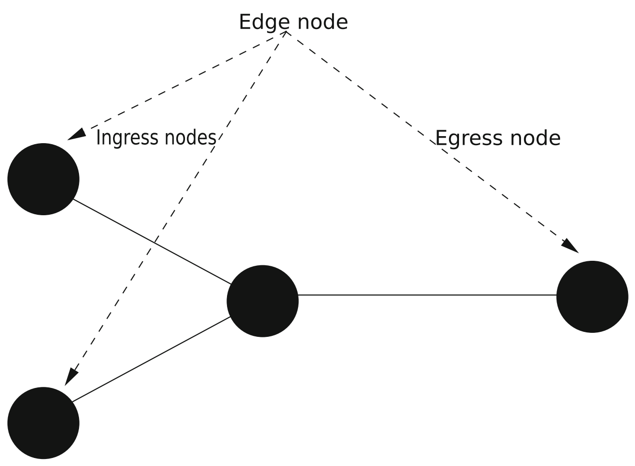


Fig. 7 Scenario adopted in the evaluation of the impact of traffic scaling in the network dimensioning

ing) [25], which schedules the requests in the channel that minimizes the gap between the arriving request and the ending of last reservation, if there are available voids or in the nearest horizon. To produce multiscaling and monoscaling traffic, assembly parameters were the same used in previous sections, i.e. 1 (multiscaling) and 150 ms (monoscaling) for time-based assembly algorithms and 1 (multiscaling) and 150 KB (monoscaling) for byte counting-based ones. The processing time of the control packet was set up to 50 us. The ingress nodes are fed with multifractal traffic generated from traffic traces presented in Table 2.

The idea was to measure the impact of changes in statistical properties of traffic on the network dimensioning. For this end, we measure the blocking probability (BP), the number of additional channels (compared to the number used to measure the blocking probability) so that the network has blocking probability equal to zero (C).

Results are shown in Table 6. Regardless the adopted burst assembly policy, if the outgoing traffic remains multifractal, the blocking probability is higher than it would be if the traffic characteristics were changed to monofractal. It occurs due to the high frequency of burst and control packets generated by the network when the traffic remains multifractal, demanding more resources.

Moreover, the traffic-based policy leads to higher blocking probability values than time-based policies, because the former, as pointed out in Sect. 2, produces a more explosive traffic contributing to longer periods of contention.

3 Identification of traffic scaling regimes

The analysis presented in Sect. 2 showed that the set up of the parameter of the assembly mechanism has great influence on the change in the traffic scaling. Moreover the cut-off time scale has also a great influence. Thus, it is necessary to determine how to estimate the cut-off time scale. Since it represents the transition scale between the monofractal and multifractal regimes, it is necessary to characterize such regimes before presenting the estimation of the cut-off itself. This section presents the concepts related to the identification of such scaling regimes.

3.1 Characterization of multifractal scaling

Let $X(t)$ be the traffic arrival process defined as the total amount of traffic that arrives in the interval $[0, t)$. The associated increment process $X_\Delta(i)$ is defined by:

$$X_\Delta(i) = X(i\Delta) - X((i-1)\Delta) \tag{6}$$

Their statistical moments behave as:

$$\sum_i X_\Delta(i)^q \sim c(q)\Delta^{-\tau(q)} \qquad \Delta \to 0 \tag{7}$$

If $\tau(q)$ is a linear function of q, the process is monoscaling; otherwise, it is multiscaling.

Taking the logarithm in Eq. 7, it follows that:

$$\log\left(\sum_i X_\Delta(i)^q) \approx -\tau(q)\log(\Delta) + \log(c(q)\right), \tag{8}$$

which shows that $\log(\sum_i X_\Delta(i)^q)$ depend linearly on $\log(\Delta)$ and that $-\tau(q)$ the slope of the curve obtained.

To evaluate the function $\tau(q)$, we use the function partition sum. For the traffic arrival process $X(i)$, $1 \le i \le N$ defined in the interval $[0, T]$ and the scale $\delta = T/N$, the partition sum is given by:

$$\sum_i X_\Delta(i)^q = \sum_{k=1}^{N/\Delta} \left(X_k^\Delta\right)^q \tag{9}$$

where N is the number of aggregation scales and

$$X_k^\Delta = \sum_{l=1}^{\Delta} X(k-1)\Delta + l \tag{10}$$

is the process obtained by aggregating arrivals on a time scale $\delta = T\Delta/N$.

Aggregating the process $X(i)$ using the partition sum function for a certain q_i and varying Δ, we obtain a set of points in the plain $\log(\Delta)$x$\log(\sum_i X_\Delta(i)^q)$. Thus, the function $\tau(q_i)$ can be obtained using linear regression on the points of the plane $\log(\Delta)$x$\log(\sum_i X_\Delta(i)^q)$. Finally, the function $\tau(q)$ can be built by interpolating the values of $\tau(q_i)$ in different statistical moments.

Table 6 Blocking probability and demand for additional channels

Assembly policy	Byte counting based				Time based			
Scaling	Multi		Mono		Multi		Mono	
Trace	BP (%)	C	BP (%)	C	PB (%)	C	PB (%)	C
ANL-1111548257	4	1	0	0	2	1	0	0
BWY-1069762448	6	1	0	0	3	1	0	0
MEM-1111247410	2	1	0	0	1	1	0	0
MEM-1111679715	1	1	0	0	2	1	0	0
MEM-1112013766	3	1	0	0	1.3	1	0	0
TXS-1113503155	0	0	0	0	0.1	1	0	0
Mean	2.66	1	0	0	1.56	1	0	0

3.2 Characterization of monofractal scaling

Consider a process $Y(t)$ and its associated stationary incremental process $Y_\Delta(i) = Y(i\Delta) - Y((i-1)\Delta)$. Assume that $Y(t)$ is *H-self-similar*, i.e.

$$Y(Ci) \overset{d}{=} C^H Y(i)$$

Let

$$Y_k^\Delta = \frac{1}{\Delta}\sum_{l=1}^{\Delta} Y(k-1)\Delta + l = \frac{1}{\Delta}\overline{Y}_k^\Delta$$

Then [26]:

$$Y_\Delta(i) \overset{d}{=} \Delta^{1-H} Y_k^\Delta \tag{11}$$

As proposed in [27], a test of the self-similar characteristic of a traffic stream can be done by the analysis of the behaviour of its statistical moments.

Let:

$$\mathbb{E}|Y^\Delta|^q = \frac{1}{N/\Delta}\sum_{k=1}^{N/\Delta} |Y_k^\Delta|^q$$

In analogy to multifractal analysis presented in Sect. 3.1, we have:

$$\sum_i Y_\Delta(i)^q = \sum_{k=1}^{N/\Delta} |\overline{Y}_k^\Delta|^q = \Delta^{q-1} N \cdot \mathbb{E}|Y^\Delta|^q$$

If $Y_\Delta(i)$ is self-similar (Eq. 11) [26]:

$$\log\left(\sum_i Y_\Delta(i)^q\right) \approx \gamma(q)\log(\Delta) + \text{const} \tag{12}$$

which shows that $\log(\sum_i X_\Delta(i)^q)$ depends linearly on the $\log(\Delta)$ and $\gamma(q)$ is linear in q such that:

$$\gamma(q) = qH - 1. \tag{13}$$

Thus, the self-similar test proposed by Taqqu et al. [27] determines whether there exists a value of γ such that the Eq. 12 is valid. If γ depends linearly on q, then the traffic is self-similar.

3.3 Identification of the cut-off time scale

As discussed in Sect. 3.1 and 3.2, different scaling regimes can be visually observed in the log–log plot of the aggregated incremental process $X_\Delta(i)$ as a function of the aggregation interval Δ.

On large time scales, the traffic is modelled by monoscaling (self-similar) process according to Eq. 11. This makes the log–log plot looks linear according to Eq. 12. The slope of the lines is then given by Eq. 13.

On small time scales, the traffic is modelled by multiscaling process (Eq. 7). The log–log plot in such scales can also be linear. However, it is possible to observe a change in the slope of the curves due to the presence of different scaling regimes (monoscaling and multiscaling). Consequently, different coefficients ($\tau(q)$ and $\gamma(q)$) lead to the change in the slope of the curves formed in each regime. The cut-off time scale is defined as the maximum inflexion point, which differentiates the slopes of the curves, among all statistical moment [19].

Such analysis was conducted for all traffic traces used in this paper. Figure 8 illustrates the results obtained from the evaluation of this relation for the trace BWY-1069762448. The time scales Δ indicate the number of realizations of the original process $\overline{X}(t)$ that is used to generate a single realization of the aggregated process $X^{(\Delta)}(t)$. The curves have different slopes in different magnitudes of time scale. The cut-off scale for this example is approximately 300 ms.

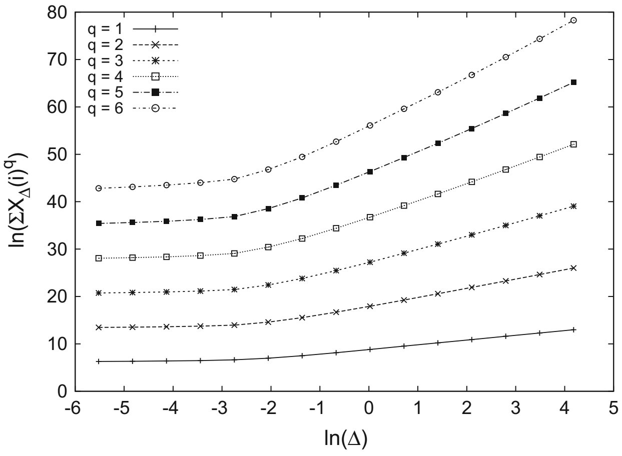


Fig. 8 Cut-Off scale of traffic trace BWY-1069762448

4 Non-visual identification of the scaling characteristic of network traffic

As stated before, the detection of the traffic scaling characteristic is done through visual inspection of diagrams [19,23], which can lead to misleading results since the derivative of the function $\tau(q)$ can vary over a small range, making the plot look linear. To overcome such problem, in this section, it is proposed a method based on least square regression model to identifying the cut-off time scale of a multifractal flow.

4.1 Automatic detection of the cut-off scale

As discussed previously, the cut-off scale determines the aggregation scale beyond which the traffic flow presents monoscaling characteristics. It means that, beyond the cut-off scale, the points (x_i, y_i) where $x_i = \log(\Delta^*)$ and $y_i = \log(\sum_i X_\Delta(i)^q)$ are linearly disposed according to the relation presented in Eq. 8. In addition, the slope of the curves is given by the function $\tau(q)$.

Now consider the set of ordered pairs

$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$.

It is known that the points $\{(x_{\Delta^\star}, y_{\Delta^\star}), ..., (x_n, y_n)\}$ are linearly disposed, where $(x_{\Delta^\star}, y_{\Delta^\star})$ represents the cut-off scale. Moreover, the points $\{(x_1, y_1), ..., (x_{\Delta^\star-1}, y_{\Delta^\star-1})\}$ are either linearly arranged in a regime driven by Eq. 12 which slope is given by $\gamma(q)$ or arranged in a nonlinear fashion.

Given a set of n points, $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, the quality of a linear regression model is measured by the coefficient of determination, R^2 which is a measure of the proportion of the variability of a variable that is explained by the variability of another one. In a perfect correlation, $R^2 = 1$. The coefficient of determination is given by the following expression:

$$R^2 = \frac{\left(\left(\sum_{i=1}^n y_i^2\right) - n\bar{y}^2\right) - \sum_{i=1}^n y_i^2 - b_0\sum_{i=1}^n y_i - b_1\sum_{i=1}^n x_i y_i}{\left(\sum_{i=1}^n y_i^2\right) - n\bar{y}^2} \tag{14}$$

with $b_0 = \overline{y} - b_1\overline{x}$ and $b1 = \frac{\sum xy - n\overline{x}\overline{y}}{\sum x^2 - n(\overline{x})^2}$.

The proposed method to detect the cut-off scale consists on observing the behaviour of the function formed by the different values assumed by the coefficient of determination as new points (x_i, y_i) are added to the regression. It is known that either in monoscaling or in multiscaling regimes the coefficient of determination is close to 1 since the points are disposed linearly. The presence of an inflection point (denoting the transition between multiscaling and monoscaling regimes) causes an abrupt decrease in the coefficient of determination value.

Algorithm 1 Compute-cut-off scale

```
INPUT
Set of $n$ ordered pairs of numbers
OUTPUT
Cut-off scale $\Delta^*$.
Compute $\Delta^*$
 1: $M \leftarrow 0$
 2: for all ( $q_{\min} \le q \le q_{\max}$): do
 3:   $S \leftarrow \{\}$
 4:   $i \leftarrow n$
 5:   $S \leftarrow \cup\{(x_i, y_i)\}$
 6:   Compute $R^2$ using $S$ points
 7:   while $((i \ge 1) \wedge (R^2 \ge \delta))$ do
 8:     $i \leftarrow (i-1)$
 9:     $S \leftarrow \cup\{(x_i, y_i)\}$
10:     Compute $R^2$ using $S$ points
11:   if $(R^2 < \delta)$ then
12:     if $(i \ge M)$ then
13:       $M \leftarrow i$
14: $\Delta^* \leftarrow M$
15: Return $\Delta^*$
```

The proposed method works as follows: starting with the point (x_n, y_n), the method incrementally recalculates the value of the coefficient of determination as new points are added to the set. Since the points of $\{(x_{\Delta^*}, y_{\Delta^*}), ..., (x_n, y_n)\}$ are linearly disposed, it is expected that the value of R^2 to be close to 1, since the regression will have good quality. On the other hand, as the points belonging to the nonlinear region are added to the set, the value of R^2 deteriorates. The method is formally presented in Algorithm 1.

As input, the algorithm receives, for each statistical moment q ($q_{\max}$ means the maximum allowed value of q), a set S of n points (corresponding to the number of aggregation scales to which the $X(t)$ process was subject). As output, the algorithm returns the approximate value of Δ^*.

The value of R^2 is computed over a set of points belonging to S. As new points are added to the set, R^2 is recalculated.

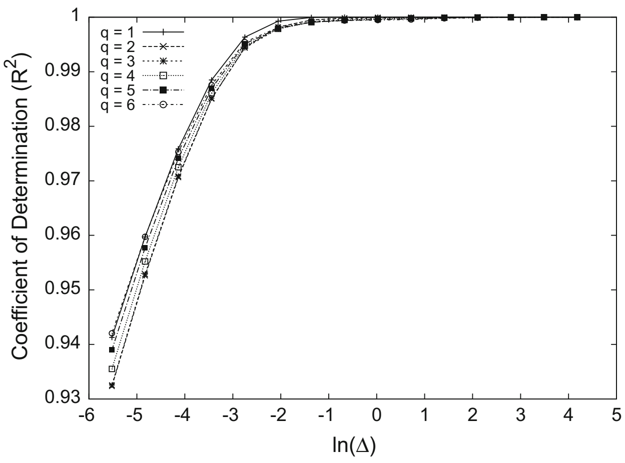


Fig. 9 Coefficient of determination for different scales

This process is repeated until the value of R^2 becomes lower than a threshold δ.

The algorithm keeps also the variable M, which stores the maximum value (among all statistical moments q) of the time scale in which the linear regression is no longer acceptable, corresponding in this way, to the cut-off time scale.

Figure 9 illustrates the procedure showing the coefficient of determination of the linear regression applied over the curves.

4.2 Computational complexity

The following two theorems establish the computational complexity of the proposed procedure. The complexity of the estimation of the cut-off scale depends on both the complexity of traffic aggregation on different scales and the proposed procedure.

To aggregate the arriving process $X(t) \in [0, t)$, on different time scales $\Delta \in [\Delta_{\min}, \Delta_{\max}]$, it is necessary to compute Eq. 10 for each statistical moment $q_{\min} \leq q \leq q_{\max}$.

Theorem 1 *The execution time of the aggregation procedure is a linear function of the number of aggregating point n of the process $X(t)$.*

Proof To compute Eq. 10 for fixed values of Δ and q, a total of $\frac{N\Delta}{\Delta}$ interactions are necessary. Let $|M|$ be the number of statistical moments and $|E|$ the number of scales onto which the process will be aggregated. The aggregation is pursued

$$(|M|) \cdot (|E|) \cdot n = O(n)$$

times, leading to a linear execution time. □

After aggregating the process $X(t)$ onto different time scales, the procedure compute-cut-off scale is executed. Let $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ $x_i = \log(\Delta)$ e $y_i = \log(\sum_i X_\Delta(i)^q)$, the set of points resulting from the aggregation procedure. Theorem 2 establishes the time complexity of the procedure compute-cut-off scale.

Theorem 2 *The execution time of the algorithm compute-cut-off scale is a linear function of n, the number of points of set S.*

Proof The compute-cut-off procedure starts with a set of points and progressively adds at most n points at each step until the value of the coefficient of determination differs by δ to the unit value. By using the partial sums of y_i's and x_i's corresponding to the S' with j points, it is possible to compute the coefficient of determination for a set S'' with $j + 1$ points. Therefore, for each additional point added to the set, it is necessary to update the partial sums which is repeated at most n times. The algorithm compute-cut-off is executed for all statistical moments. The execution time is, therefore,

$$(|M|) \cdot n = O(n)$$

where $|M|$ is the number of moments employed. □

5 Burst assembly policies for traffic changing

In Sect. 2, it was shown that the traffic characteristic can be changed depending on the set up of the burst assembly parameters.

Let b be the burst size threshold used by the byte counting-based assembly algorithm and λ the mean packet arrival rate (in terms of bytes per second), we have:

$$b = t \cdot \lambda \tag{15}$$

Using the previous discussion and Eq. 15, one can conclude that if:

$$\frac{b}{\lambda} > \Delta^\star \tag{16}$$

the traffic has its multiscaling characteristics changed to monoscale.

Therefore, the burst assembly can be done by adjusting the parameters t and b to the calculated values of $\Delta^\star$ and λ, thus guaranteeing the traffic transformation. For the computation of $\Delta^\star$, the proposed method (Algorithm 1) can be used to obtain the arrival rate from service-level agreement (SLA).

The problem is more challenging if delay requirements are added to the burst assembly process, i.e. if the assembly time has to be lower than the cut-off time scale. In this case, simply adjusting t or b to a value higher than the cut-off scale may cause a negative effect on the QoS provisioning to the applications.

Consider a set $\mathcal{C}$ of classes of service, each having a different maximum tolerable end-to-end delay for class i, D_i, which can be expressed as:

$$D_i \geq \alpha_i + T_i + d_i(b) \qquad (17)$$

where T_i is the offset time of class i, d_i is a factor that considers the propagation and transmission delays of a burst with size b belonging to class i and finally α_i is the maximum assembly time. Let t_i be assembly time, b_i the byte counter threshold and λ_i the mean packet arrival rate (expressed in bytes per second) of class i. Thus, $\alpha_i = t_i$ or $\alpha_i = \frac{b_i}{\lambda_i}$ depending on the employed burst assembly algorithm.

Again, if the assembly time of all classes of service is longer than the cut-off time scale, the assembler can adjust t_i or b_i to induce the production of monoscaling traffic. However, if there is any class with assembly time shorter than the cut-off time scale, traffic transformation is only possible by employing different burst assembly techniques.

Considering an OBS network in which the burst size is restricted to the range $[s : S]$, where s is the minimum burst size and S is maximum burst size, it is possible to determine by using Eq. 16, values of λ_i, Δ^*.

We propose different policies that transform the scaling of a traffic flow and yet guarantees delay requirements. Criteria used by different policies differ by how the priority level of a service class is considered as well as the backlog of a class at the assembler.

Notation will be introduced for describing the proposed policies (Table 7). Let $\mathcal{C}$ be the set of CoS each with its own end-to-end delay requirement D_i, and $\mathcal{C}'$ a subset of $\mathcal{C}$, the set of classes with enqueued packets and let p_i be the priority level of class i such that $p_0 > p_1 > \cdots > p_n$. Let Q_i be the queue of packets of class i and $|R|$ size of the burst being assembled and β the required minimum burst size. Finally, it is assumed that the cut-off time scale ($\Delta^\star$) of the multifractal flow and the mean packet arrival rate of class i are known, thenceforth, β can be determined.

Table 7 Table of symbols used in the algorithms

Symbol	Meaning
$\Delta^\star$	Cut-off time scale
t_i	Assembly timer
b_i	Bytes threshold
λ_i	IP packet arrival rate of class i
β	Minimum burst size
$\mathcal{C}$	Number of classes of service
$\mathcal{C}'$	Non-empty classes of service
R	Assembled burst
Q_i	Assembly queue of class i
max_unused($\mathcal{C}'$)	Function that selects the class with highest priority
head(Q_i)	Function that takes the first packet in the ith assembly queue
X	Surplus of burst R

The Round Fit policy (Algorithm 2) creates a burst with packets of class i. If the size of the resulting burst is smaller than β, the burst is filled with packets of lower priority classes to reach the required minimum size (β). The policy is depicted in Fig. 10a.

Algorithm 2 Round Fit (RF)

```
INPUT
Cut-off scale $\Delta^*$, $t_i$ (or $b_i$) and $\lambda_i$
OUTPUT
Burst with size $\beta$
Round Fit
1: Calculate $\beta$ according to Eq. 16
2: while $Q_i \neq 0$ do
3:   $R \leftarrow R \cup head(Q_i)$
4: if $|R| < \beta$ then
5:   $j \leftarrow (i+1) mod |\mathcal{C}'|$
6:   while $|R| < \beta$ do
7:     while $Q_j$ do
8:       $R \leftarrow R \cup head(Q_j)$
9:     $j \leftarrow (i+1) mod |\mathcal{C}'|$
```

The RF algorithm works as follows: In Line 1, the minimum burst size is determined using Eq. 16. In Lines 2–3, packets belonging to the ith class are added to the burst. Line 4 verifies if the minimum burst size was reached. If not, packets from other classes are added to the burst in a round robin fashion until the minimum burst size is achieved (Lines 6–8).

The *High Priority Fit (HPF)* policy (Fig. 10b) is similar to the RF policy except that it fills bursts smaller than the minimum required size in a rounding robin fashion starting from the highest priority class. The main differences between the HPF and the RF algorithm are that Line 5 is replaced by $j \leftarrow$ max_unused($\mathcal{C}'$) and Line 9, by $j \leftarrow$ max_unused($\mathcal{C}'$).

In the Proportional Fit (PF) algorithm (Algorithm 3), the space required to complete the minimum burst size divided equally among all classes with queued packets.

The PF algorithm differs from the RF algorithm after Line 5. The remaining space burst is divided equally among all other non-empty classes (Lines 5 and 6) and, starting from the highest priority class (Line 7), packets from all classes are added to the burst (Lines 9–14).

The Backlog Fit (BF) policy (Algorithm 4) uses the unfinished work in each traffic class to define the contribution of that class to the filling of a burst with minimum a size. The process is carried out in a round robin fashion, starting with the longest queue. By doing this, the policy prioritizes traffic classes that have recently produced more traffic.

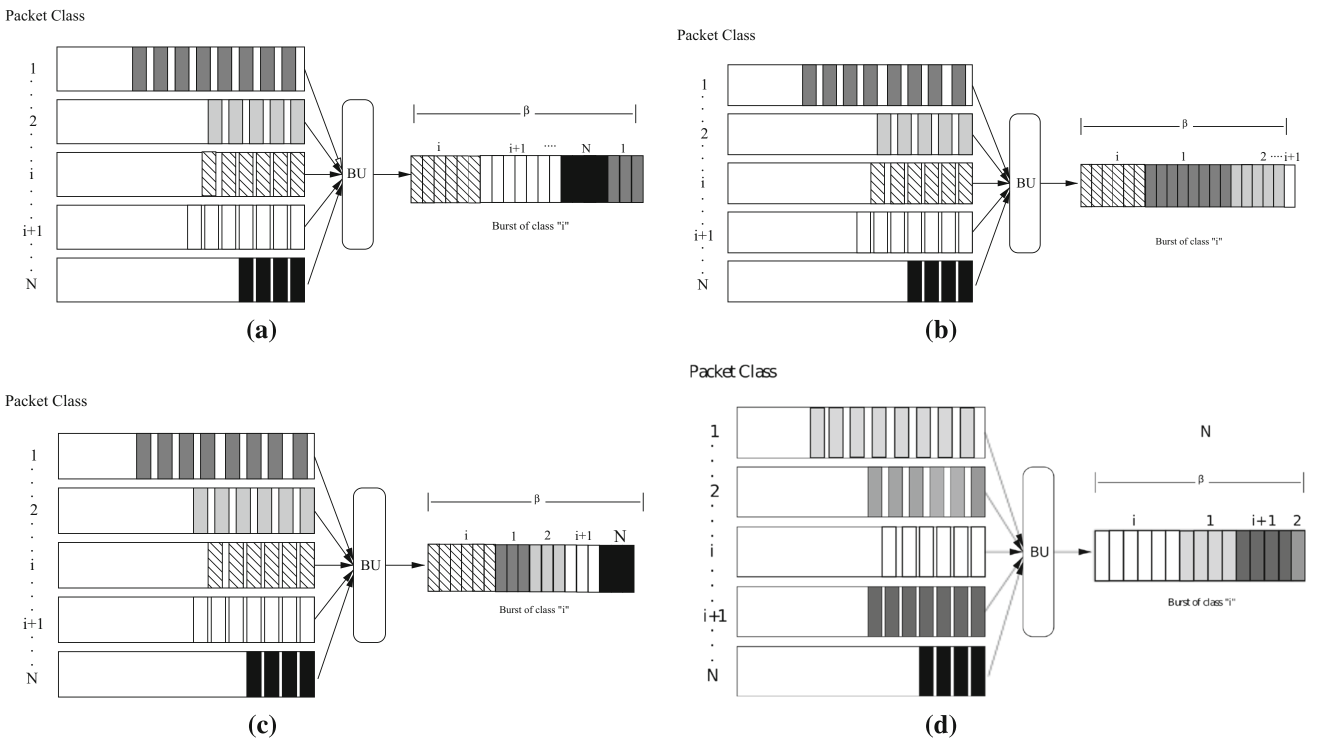


Fig. 10 Burst assembly algorithms, **a** Round Fit, **b** High Priority First, **c** Proportional Fit, **d** Backlog

Algorithm 3 Proportional Fit(PF)

INPUT
Cut-off scale Δ, t_i, b_i) and λ_i
OUTPUT
Burst with size β
Proportional Fit

```
1: Calculate $\beta$ according to Eq. 16
2: while $Q_i \neq 0$ do
3:    $R \leftarrow R \cup head(Q_i)$
4: if $|R| < \beta$ then
5:    $X \leftarrow (\beta - |R|)$
6:    $F \leftarrow \lceil X/(\mathcal{C}' - 1) \rceil$
7:    $j \leftarrow max\_unused(\mathcal{C}')$
8:    $l \leftarrow 0$
9:    while $|R| < \beta$ do
10:       while $(Q_j) \wedge (l < F)$ do
11:          $R \leftarrow R \cup head(Q_j)$
12:          $l++$
13:       $l \leftarrow 0$
14:       $j \leftarrow max\_unused(\mathcal{C}')$
```

Algorithm 4 Backlog Fit

INPUT
Cut-off scale Δ, t_i, (or b_i) and λ_i
OUTPUT
Burst with size β
Backlog Fit

```
1: Calculate $\beta$ according to Eq. 16
2: while $Q_i \neq 0$ do
3:    $R \leftarrow R \cup head(Q_i)$
4: if $|R| < \beta$ then
5:    $X \leftarrow (\beta - |R|)$
6:    $j \leftarrow max\_unused(\mathcal{C}')$
7:    $l \leftarrow 0$
8:    while $|R| < \beta$ do
9:       $F_i \leftarrow \lceil \frac{|Q_j|}{\sum_i |Qi|} \rceil \cdot X$
10:       while $(Q_j) \wedge (l < F_i)$ do
11:          $R \leftarrow R \cup head(Q_j)$
12:          $l++$
13:       $l \leftarrow 0$
14:       $j \leftarrow max\_unused(\mathcal{C}')$
```

The main difference between the PF and BF algorithms is that when the minimum burst size is not reached in the BF algorithm (Line 4, the remaining buffer space is filled by taking packets from all non-empty queues (Lines 5–14), proportionally to queue occupancy (Lines 9–14).

6 Performance evaluation

To illustrate the benefits of adopting burst assembly policies driven by traffic scaling changes, simulations using the NS-2 tool were carried out. First, changes in scaling brought about by the burst assembly were investigated, and then the impact of burst assembly policies on provision of QoS was assessed.

6.1 Traffic scaling changes

In the first set of simulations, an edge node with a buffer size of 32 MB, which can serve five traffic classes was simulated. Real network traces showing multifractal scaling were used

Table 8 Scaling characteristics of real traffic traces used in the evaluation of the proposed burst assembly policies

Trace	λ (Mbps)	Δ^* (ms)	β (KB)	Holder exponent	Var	C. I.
IPLS-CLEV-090000-0	412.131	3	205	0.83	0.0400	(0. 80, 0.86)
IPLS-CLEV-090000-1	457.852	2.8	230	0.792	0.007	(0.791, 0.793)
IPLS-CLEV-091000-0	363.754	3.2	180	0.675	0.001	(0.672, 0.678)
20040601-193121-0	770.00	1.6	200	0.683	0.0014	(0.674, 0.692)
20040601-193121-1	1.648	1.3	410	0.689	0.0021	(0.676, 0.702)
20040601-194000-1	828.616	1.3	210	0.622	0.0027	(0.621, 0.623)

to feed the traffic classes. Moreover, the minimum burst size (β) necessary to ensure $\beta/\lambda > \Delta^\star$ was derived for each trace (Table 8).

The assembly time t_i of each CoS i was adjusted to $1ms$, below the cut-off time scale of all traces used. Moreover, to ensure that burst assembly was triggered by the timeout, the byte counter (b_i) of each class was adjusted to 32 MB.

Figure 11 shows the cascading function of the ingress traffic when the BF burst assembly policy was employed. The scaling behaviour was scrutinized on large time scale (32, 1024 ms). The linear behaviour of the cascading function suggested that the traffic injected into the network core was monofractal. Similar behaviour was found when the other three policies were applied for the same network scenario.

Although the scaling property of traffic was not altered by the assembly policy, it did smooth the burstiness during the process. Considering the Hurst parameters, (Table 9), and the corresponding average Holder exponent, (Table 8), it is clear that the burst assembling changed the variability of traffic under fractal assumptions.

Furthermore, the queuing delay, i.e. the delay introduced due to the burst assembly policies, was measured for all of the policies (Table 8) and none of them surpassed the established upper bound, i.e. one millisecond. In fact, the PF and BR decrease the delay. Both policies serve packets from all queues, taking fewer packets from each class. This holds their timer trigger, reducing the gap between assembling events. Moreover, the policies RF and HPF serve traffic from class i (the class with an expired timer) and use the neighbouring classes to complete the burst payload. By doing so, traffic classes in the vicinity will have their timers restarted and packets are thus delayed until their predefined assembling threshold, as can be seen in Table 8.

Finally, the PF and BF policies produce throughput greater than that given by the RF and HPF policies (Table 8). The first two policies fill the burst payload with packets from all classes, even though their timers are still running. These opportunistic transmissions open a space for the accommodation of incoming traffic in the current time window that would be served only in further assembling cycles. Hence, for the same assembling period, the PF and BF policies are able to send more traffic than do the RF and HPF policies. Since fewer classes are served, more packets from each class are transmitted in each cycle, resetting their timers. Therefore, packets from those classes will be served again in the next one millisecond cycle.

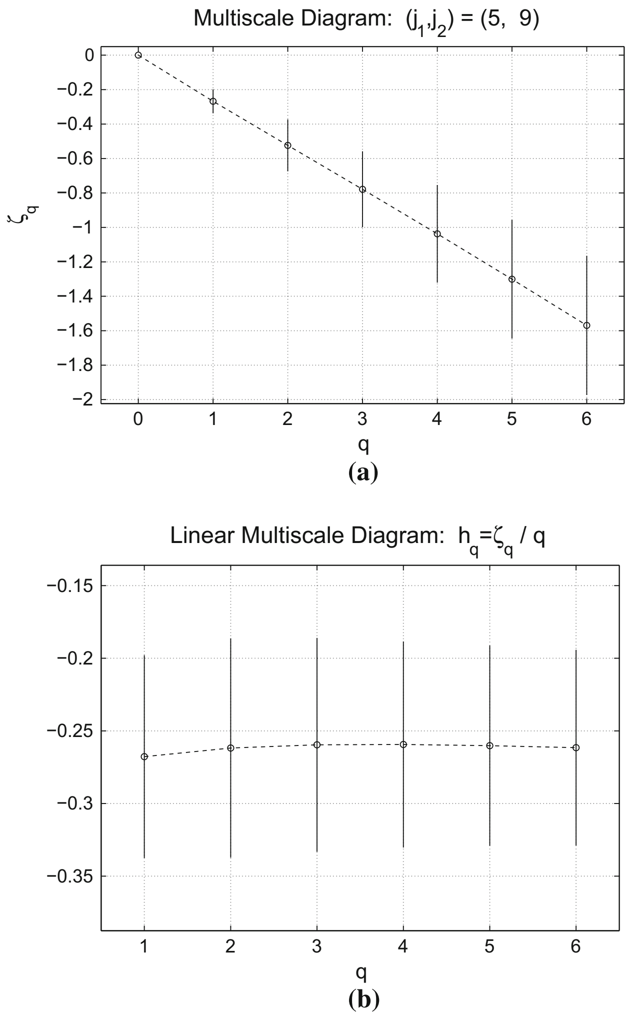


Fig. 11 Multiscale and Linear Multiscale Diagrams of the output traffic generated by the BF policy, **a** Multiscale Diagram of the traffic generated by the BF policy, **b** Linear Multiscale Diagram of the traffic generated by the BF policy

Table 9 Hurst parameter, Assembly delay and throughput for the trace IPLS-CLEV-090000-0

Algorithm	H	Assembly delay (s)	Throughput (Mbps)
RF	0.745	0.001	410.41
HPF	0.728	0.001	410.41
PF	0.751	0.00098	412.02
BF	0.781	0.000993	411.99

6.2 Assessing policy performance under heavy traffic load

In the previous section, the proposed policies were evaluated considering only the ability to make changes in traffic scaling. An additional question that needs to be answered is how these policies impact the QoS provided. To assess this impact, simulations were employed with a scenario in which an increase in packet arrival rate could be controlled.

To generate synthetic traffic with monofractal characteristics, the traffic was generated by multiplexing several ON–OFF Pareto distributed sources implemented in the OBS-ns module [28]. The ON and OFF periods were set up with the same mean duration of 100 ms. Additionally, when the sources were in the ON state, they transmitted with rate of 50 Mbps. The assembly delay experienced by IP packets and the average throughput were measured as a function of the number of classes of service supported.

Figure 12 shows the average throughput as a function of the number of classes of service. All policies reacted similarly to the traffic load by improving the average throughput of the system. When queues are under heavy traffic, policies could fill the burst being assembled. Even if a class runs out of packets a policy can take packets from other classes. The RF and HPF revealed similar results. Since the networking setup is a heavy traffic scenario, the burst filling dynamic does not distinguish between high and low priority packets, which means that in the burst assembly process, either a high or low priority packets can be selected. Compared to RF and HPF, the BF and PF policies, on the other hand, produced greater throughput for heavy traffic load.

There was an increase in the throughput produced by all of the algorithms evaluated, since as the number of classes of service supported increases, the number of options available to the algorithm to fill a minimum size burst increases. Therefore, if a given class does not have any packet available for transmission, the algorithm can take packets from other classes.

The results of the RF and HPF policies were also very similar. This is due to the fact that these two classes have the same packet arrival rate. Thus, it makes no difference wherever the bursts are filled using packets from the highest priority class or from any other. The second best algorithm was the BF algorithm, while the highest throughput was obtained by the PF algorithm.

The BF and PF algorithms both revealed better performance than did the RF and HPF policies due to the difference in their operation. The RF and HPF policies select all the packets belonging to class i (restarting its timer) as well as packets from the subsequent class (j). Thus, class j timer will also be restarted. Consequently, the classes i and j will only send packets inside bursts of a neighbouring class or when their timer has expired.

On the other hand, in the BF and PF algorithms, when a timeout expires for class i, all the packets of class i are put in the burst (timer is restarted), but only a fraction of the packets belonging to other classes is used; the timer of these classes thus continues running, which allows a new burst to be sent as soon as a timeout occurs. Since some packets of class j were already sent in the burst of class i, fewer packets

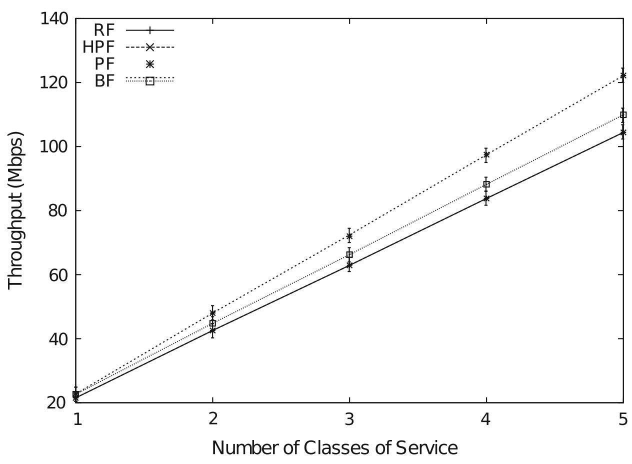


Fig. 12 Throughput as a function of the number of classes of service

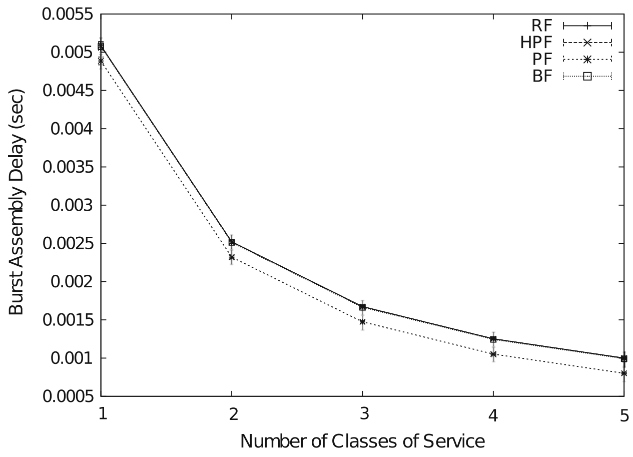


Fig. 13 Burst assembly delay as a function of the number of classes of service

from this class are added to the burst, leaving space for more packets from the other classes. This results in a continuous transmission of bursts sending data.

A similar argument can be used to explain the lower assembly delay of BF and PF algorithms, as presented in Fig. 13. Since the timers of piggybacked classes are not restarted, the assembly period is lower than that of the HPF and RF policies.

7 Conclusions

This paper has investigated the effect of burst assembly criteria on the scaling of outgoing flows of multifractal flows. Results derived via simulation have revealed that the assembly mechanisms smooths the burstiness of the input traffic. Both the mean and the variance of the Holder exponents of the multifractal output traffic are lower than those of the input traffic. Moreover, assembly time thresholds greater than the cut-off time scale of the input flow produce monofractal output flows, whereas lower values produce multifractal output flows. For policies based on byte counting, if the threshold is greater than the product between the input traffic mean arrival rate and its cut-off time scale, the resulting flow will be monofractal. Otherwise, it will be multifractal.

Furthermore, policies based on byte counting lead to monofractal output traffic with larger Hurst parameters than that produced by time-based policies. Moreover, byte counting-based policies produced multifractal output traffic with higher Holder exponents than that produced by time-based policies.

Therefore, the use of time-based policies in ingress OBS switches is recommended rather than those based on byte counting, since the former produce smoother traffic with weaker long-range dependencies; consequently, the traffic injected into the network requires fewer network resources.

Acknowledgments This research work was partially supported by CNPq, Funttel and FAPESP.

References

[1] Ge, A., Callegati, F., Tamil, L.S.: On optical burst switching and self-similar traffic. IEEE Commun. Lett. **4**, 98–100 (2000)

[2] Izal, M., Aracil, J.: On the influence of self-similarity on optical burst switching traffic. In: GLOBECOM, vol. 3, pp. 2308–2312 (2002)

[3] Hu, G., Dolzer, K., Gauger, C.: Does burst assembly really reduce the self-similarity?. In: Optical Fiber Communications Conference, OFC, vol.1, pp. 124–126 March 2003

[4] Melo, C.A.V., da Fonseca, N.L.S.: An envelope process for multifractal traffic modeling. In: Proceedings of IEEE International Conference on Communications, ICC, Paris, France, 20–24 June 2004, pp. 2168–2173. http://dx.doi.org/10.1109/ICC.2004.1312902

[5] Melo, C.A.V., da Fonseca, N.L.S.: Envelope process and computation of the equivalent bandwidth of multifractal flows. Comput. Netw. **48**(3), 351–375 (2005)

[6] Ribeiro, V.J., Zhang, Z.L., Moon, S., Diot, C.: Small-time scaling behavior of internet backbone traffic. Comput. Netw. **48**(3), 315–334 (2005)

[7] Veitch, D., Hohn, N., Abry, P.: Multifractality in tcp/ip traffic: the case agains it. Comput. Netw. **48**(3), 293–313 (2005)

[8] Toksz, M., Akar, N.: Dynamic threshold-based assembly algorithms for optical burst switching networks subject to burst rate constraints. Photonic Netw. Commun. **20**(2), 120–130 (2010)

[9] Ozsarac, S., Karasan, E.: Congestion window-based adaptive burst assembly for tcp traffic in obs networks. Photonic Netw. Commun. **20**(2), 138–150 (2010)

[10] Yu, X., Chen, Y., Qiao, C.: Study of traffic statistics of assembled burst traffic in optical burst switched networks. In: Opticomm, pp. 149–159 (2002)

[11] Cao, X., Li, J., Chen, Y., Qiao, C.: Assembling tcp/ip packets in optical burst switched networks. In: IEEE Globecom, pp. 2808–2812 (2002)

[12] Gowda, S.R.K., Shenai, K.M., Sivalingam, Cankaya, H.C. : Performance evaluation of TCP over optical burst-switched (OBS) WDM networks. In: ICC, vol. 2, pp. 1433–1437 (2003)

[13] Long, K., Tucker, R.S., Wang, C.: A new framework and burst assembly for ip diffserv over optical burst switching networks. In: GLOBECOM, pp. 3159–3164 (2003)

[14] Vokkarane, V., Zhang, Q., Jue, J.P., Chen, B.: Generalized burst assembly and scheduling techniques for QoS support to optical burst-switched networks. In: GLOBECOM, pp. 2747–2751 (2002)

[15] Gowda, S., Shenai, R., Sivalingam, K.M., Cankaya, H.: Performance evaluation of TCP over optical burst-switched (OBS) WDM networks. In: IEEE ICC, pp. 1–6

[16] Abry, P., Baraniuk, R., Flandrin, P., Riedi, R., Veitch, D.: The multiscale nature of network traffic discovery. IEEE Signal Process. Mag. **19**, 28–46 (2002)

[17] Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V.: On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Trans. Netw. **2**(1), 1–15 (1994)

[18] 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., vol. 10, no. 2, pp. 104–124. http://doi.acm.org/10.1145/364996.365003 (2000)

[19] Erramilli, A., Narayan, O., Neidhart, A., Saniee, I.: Multi-scaling models of TCP/IP and sub-frame VBR video traffic. J. Commun. Netw. **3**, 383–395 (2001)

[20] Veitch, D., Hohn, N., Abry, P.: Multifractality in tcp/ip traffic: the case against. Comput. Netw. **48**(3), 293–313 (2005)

[21] www.nlanr.net

[22] Cavanaugh, J.E., Wang, Y., Davis, J.W.: Self-similar processes and their wavelet analysis, Handbook of Statistics 21: Stochastic Processes: Modeling and Simulation. Elsevier, Amsterdam (2003)

[23] Veitch, D.: D. veitch home page. http://www.cubinlab.ee.mu.oz.au/darryl/ (2003). Accessed July 2003

[24] Figueiredo, G.B., da Fonseca, N.L.: Channel reusability for burst scheduling in obs networks. Photonic Netw. Commun. **26**(2–3), 84–94 (2013)

[25] Xiong, Y., Vandenhoute, M., Cankaya, C.: Control architecture in optical burst-switched wdm networks. IEEE J. Sel. Areas Commun. 1838–1851 (2000)

[26] Riedi, R.H., Vhel, J.L.: TCP traffic is multifractal: a numerical study. INRIA, Tech. Rep. (1997)

[27] Taqqu, M., Teverovsky, V., Willinger, W.: Is the ethernet data self-similar or multifractal? Fractals **5**, 63–73 (1997)

[28] OBS-ns Manual. http://wine.icu.ac.kr/obsns/docs.php (2009). Accessed 2 Feb 2009

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 University of Campinas. Since 2010, he has been affiliated with the Department of Computer Science of the Federal University of Bahia, Bahia—Brazil, where is currently a Associate Professor. His main research interest includes problems involving Network Performance Evaluation, Planning, Dimensioning and Optimization of Optical Networks.

Cesar A. V. Melo received M.Sc (1999) and Ph.D. (2005) in Computer Science from State University of Campinas. Since 2009, he has been with Computing Institute of Federal University of Amazonas, doing research on traffic modeling, planning and implementing traffic control mechanism for multimedia content distribution on Internet. César A. V. Melo is Associate Professor, teaching computer network and programming.

Nelson L. S. da Fonseca received his Ph.D. degree in Computer Engineering from the University of Southern California in 1994. He is Full Professor at Institute of Computing of the University of Campinas, Campinas, Brazil. He has published 350+ papers and supervised 50+ graduate students. He is Director for Conference Development of the IEEE Communications Society (ComSoc). He served as Vice President Member Relations of the ComSoc, Director of Latin America Region and Director of online Services. He is the recipient of the 2012 IEEE Communications Society (ComSoc) Joseph LoCicero Award for Exemplary Service to Publications, the Medal of the Chancellor of the University of Pisa (2007) and the Elsevier Computer Network Journal Editor of Year 2001 award. He is past EiC of the IEEE Communications Surveys and Tutorials. He is Senior Editor for the IEEE Communications Surveys and Tutorials and Senior Editor for the IEEE Communications Magazine, a member of the editorial board of Computer Networks, Peer-to-Peer Networking and Applications, Journal of Internet Services and Applications and International Journal of Communication Systems.