Computer Networks 54 (2010) 3144–3158

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An efficient and lightweight method for Service Level Agreement assessment ☆

René Serral-Gracià [a,*], Marcelo Yannuzzi [a], Yann Labit [b,c], Philippe Owezarski [c], Xavi Masip-Bruin [a]

[a] Advanced Network Architectures Lab (CRAAX), Technical University of Catalonia (UPC), Spain
[b] Université de Toulouse, UPS, 118 Route de Narbonne, F-31062 Toulouse, France
[c] LAAS-CNRS, 7, avenue du Colonel Roche, F-31077 Toulouse, France

A R T I C L E   I N F O

Article history:
Received 4 June 2009
Received in revised form 23 April 2010
Accepted 3 June 2010
Available online 10 June 2010
Responsible Editor: N. Agoulmine

Keywords:
SLA assessment
Inter-Packet Arrival Times
Hausdorff Distance

A B S T R A C T

Traditional approaches to on-line end-to-end Service Level Agreement (SLA) assessment have focused on the estimation of network QoS parameters. These approaches, however, face a trade-off between accuracy and the amount of resources needed to achieve such accuracy. This paper offers an alternative approach, where instead of estimating QoS parameters, we propose an effective and lightweight solution for directly detecting SLA violations. Our solution monitors the Inter-Packet Arrival Time (IPAT) at an end-point, wherein current IPAT distributions are periodically compared with a set of reference IPAT distributions as the main basis for detecting SLA violations. A mapping of the IPAT distribution with the current network conditions is derived, and a training algorithm that dynamically acquires the set of reference IPAT distributions is designed. For the comparison of the IPAT distributions, we propose a variant of the Hausdorff Distance algorithm. Our variant provides a better accuracy than the traditional Hausdorff Distance, while presenting linear complexity. Our proposal is validated in a real testbed, by comparing the SLA violations detected and the resources required in terms of bandwidth, with other existing alternatives as well as with perfect knowledge of current network QoS status.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

With the increasing usage of real-time and multimedia applications, on-line end-to-end Service Level Agreement (SLA) monitoring becomes a required service, since both Internet Service Providers (ISP) and customers want to receive information about the quality of network services in real-time, and more importantly, to check whether this quality complies with the contracted SLA. For this purpose, classical approaches for SLA assessment [1–5] have focused on accurately measuring (when possible) or estimating (most of the times) network QoS parameters such as One Way Delay (OWD) [6], Inter-Packet Delay Variation (IPDV) [7], Available BandWidth (ABW) [8], Packet Loss Ratio (PLR) [9], etc. Up to now, the attempts made to provide correct SLA assessment have a strong dependency on the accuracy of the estimation of QoS metrics (see, e.g., [3,10]). However, the accurate estimation of QoS metrics is a complex task; for instance, Labit et al. [11] show that the estimation accuracy of the ABW might present error rates greater than 50% in real environments. This is mainly due to network traffic variability and different traffic anomalies, which make SLA assessment based on the

☆ A preliminary version of this work appeared in Proceedings of IEEE INFOCOM 2009 under the title "Towards an efficient Service Level Agreement assessment". This work was partially funded by Spanish Ministry of Science and Innovation under contract TEC2009-07041, and the Catalan Government under contract 2009 SGR1508.

* Corresponding author. Tel.: +34 934056041.
E-mail addresses: rserral@ac.upc.edu (R. Serral-Gracià), yannuzzi@ac.upc.edu (M. Yannuzzi), ylabit@laas.fr (Y. Labit), owe@laas.fr (P. Owezarski), xmasip@ac.upc.edu (X. Masip-Bruin).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.06.005

estimation of QoS metrics a challenging problem. In addition, SLA assessment methods based on the on-line estimation of end-to-end QoS parameters present other drawbacks. In most cases, multiple entities need to be deployed and managed in the network in order to estimate the QoS metrics (e.g., multiple points of capture and points of traffic analysis are required). Furthermore, for the estimation of some QoS metrics such as the OWD, it is necessary to gather distributed information not only about the traffic itself, but also about the synchronization status of the involved entities [12], so the control traffic generated is an important bottleneck for solutions based on this approach [4].

In light of this, our work does not focus on the estimation of QoS metrics. The originality of the contribution presented in this paper is that we focus *directly* on the effective detection and reporting of SLA violations. More specifically, we focus on the traffic profile rather than on the estimation of QoS metrics, and as we shall show, our solution uses, in most cases, a single point of traffic analysis, so we consider that our approach might help designing simpler and more scalable SLA monitoring systems.

The rationale behind our approach is that, in practice, ISPs deal with SLA issues by overprovisioning their networks [13]. This means that, in the end, the complex instrumentation needed for the on-line assessment of QoS metrics ends up estimating values that are, most of the time, within acceptable bounds. We argue that this is inefficient for SLA assessment, since in practice SLAs are typically violated during service disruptions or severe congestion, i.e., when the QoS provided by the network collapses. Based on this observation, this paper makes the following contributions:

(1) We propose to use the Inter-Packet Arrival Time (IPAT), and derive a training algorithm that periodically compares the current IPAT distribution with a set of reference IPAT distributions that are dynamically acquired, as the basis for detecting SLA violations. Previous works [14,15] have shown that the IPATs are tightly correlated with network congestion. In this paper, we bring this correlation one step further by mapping the IPAT distribution with current network conditions. In particular, our proposal performs statistical analysis of the IPATs, with the goal of detecting changes in the status of the network. The advantage of this approach is that the IPATs can be easily computed at the destination by getting the reception timestamps of the packets – indeed, the computation of the IPAT only involves a subtraction of two integers (the timestamps).
(2) In order to compare the IPAT distributions, we propose an algorithm inspired in the well-known *Hausdorff Distance*. Our algorithm adapts the geometric Hausdorff Distance to handle distances among IPAT distributions. We show that, our approach substantially improves the SLA violation detection accuracy in our context, while, in contrast with the traditional Hausdorff Distance algorithm which has quadratic complexity, our algorithm presents linear complexity.

Our solution is tested and validated using different real scenarios; first over a controlled testbed with customizable network conditions, and later over an European-wide scenario with 12 different testbeds and five different access technologies. In both scenarios we performed a broad range of tests, including synthetically generated UDP traffic, as well as audio and video flows generated by a real video streaming application. Our results confirm the adaptability and accurate detection of the different SLA violations found in our traces, all accomplished with significant reductions in the bandwidth resource usage (we achieve an 89% reduction in the network resource usage due to control traffic), which is a promising result.

The rest of the paper is structured as follows. Next section details relevant related work. In Section 3, we outline the main contribution of this paper, and we also detail the distance algorithms considered in this work. Section 4 presents the methodology used for performing the on-line SLA assessment. The validation methodology is described in Section 5, including details about the different tests performed and their conditions. Section 6 shows the evaluation results, while in Section 7 we study the impact of the system parameters on the behavior of the monitoring solution proposed. In Section 8, we discuss the scalability of the system, and finally, Section 9 concludes the paper and presents the open issues left for future work.

2. Related work

In general, network measurement techniques can be split into two different groups, active and passive network measurements, each one with its own advantages over the other.

SLA assessment has been previously studied using active traffic analysis. Some important work has been performed by Barford and Sommers in [16], where the authors highlight the limitations of packet loss estimation using active probes, compared to the ones found via SNMP in commercial routers. This work was continued by Sommers et al. in [10], where the authors improved the loss estimation of classical Poisson-modulated probing mechanisms by presenting *Badabing*, a dynamic active tool that improves the accuracy depending on the resources used for the estimation. More recently, in [3], Sommers et al. gathered together all the above knowledge, and presented *SLAm*, another active probing tool that implements innovative packet loss, delay, and delay variation estimation techniques for SLA assessment. All mentioned publications stress the need for proper metric estimation in order to lead to correct SLA assessment.

Our work differs from these proposals, in the sense that our methodology does not focus on accurate metric estimation, but rather in the search for relevant packet information to efficiently infer whether the network quality violates an SLA. Moreover, in contrast to the active solutions adopted by the work mentioned above, we use a passive traffic analysis approach.

Regarding passive traffic analysis, some research has been done in our previous work [4,5,17,18], in which we proposed a distributed infrastructure for inferring the network quality by extracting the performance metrics from detailed packet information. The metric computation was centralized, and required several collection points on the edges of the network to send packet information (timestamps, etc.) to a central entity, with the consequent use of bandwidth due to the so-called control traffic. The information received is used by the central entity to compute the flow's network metrics (OWD, IPDV and PLR). With a different approach, in this paper, we use passive monitoring as a method for training our SLA assessment algorithm, which needs such network performance information for associating the IPAT distributions to the corresponding network performance status.

In [19], we present a similar technique to the one described here, which is based on the utilization of the *Kullback–Leibler Divergence* [20]. The solution proposed in [19] is able to detect SLA violations with high accuracy, though at the cost of requiring a significant amount of network resources.

In this paper, we propose a solution that offers a good trade-off between accuracy and the amount of resources needed to achieve such accuracy. We show the better behavior of the *Simplified Hausdorff* Distance algorithm by comparing our results with the ones obtained previously with the *Kullback–Leibler Divergence* in exactly the same network conditions.

3. IPAT distributions comparison

As described above, we use the IPATs in order to detect violations to the SLA. Even with their good characteristics in terms of computational efficiency, just gathering IPATs is not enough to provide SLA assessment. First, IPATs do not contain useful information about the performance of the network. Second, IPATs might change unexpectedly due to changes on the network conditions, or due to changes in the traffic profile (e.g., codec change in VoIP, or a silence period in the conversation, etc.). Third, there is currently no direct mapping between the IPATs and SLA violations.

To tackle these challenges, we propose to periodically compare the current IPAT distribution with a set of reference IPAT distributions. Clearly, getting this set of reference distributions means to integrate an on-line training process that records all new IPAT distributions observed on the network. This training process needs to link each of these new IPAT distributions to the current QoS status, and thereby, associate each reference distribution to a particular QoS level of the network.

In this section, we focus on the general description of the distance algorithms that we used for comparing the IPAT distributions, while we leave for the next section the description of the complete infrastructure using these algorithms.

3.1. Hausdorff Distance

The Hausdorff Distance [21] is mostly used in image recognition and object location, and it is known for its good versatility in measuring the distance between two different geometrical objects.

Definition 1. The Hausdorff Distance is the maximum of all the minimum distances between two sets.

Formally, the Hausdorff Distance gets a finite set of points $P = \{p_1,\ldots,p_n\}$ representing a reference, and compares it to a probe $Q = \{q_1,\ldots,q_m\}$ using:

$$h(P,Q) = \max_{p\in P}\left\{\min_{q\in Q}\{g(p,q)\}\right\}, \tag{1}$$

where $g(p,q)$ stands for the geometrical distance between p and q. Its efficiency – taken as the computational cost – is normally not suitable for on-line operations. In particular, to compute the Hausdorff Distance, the minimum distance from *all* the elements of P towards *all* the elements of Q must be obtained. Hence, the algorithmic cost is $O(nm)$ where n and m are the sizes of P and Q, respectively. In many practical problems $n \approx m$, thus the complexity of the Hausdorff Distance is often $O(n^2)$.

3.2. The Simplified Hausdorff Distance

The Hausdorff Distance was originally devised for the geometric plane, but in our scenario we use distributions instead of polygons. The traditional Hausdorff Distance would compute the distance between all the elements of a set against all the elements of another, which in our context introduces inaccuracies in the SLA assessment. This is because in our case, we operate over discrete distributions and thus there is a dimension not present in the geometrical plane, namely, the *bin* width. A bin width is the IPAT interval where the IPATs are classified into the same position within the discrete distribution. For example, for a bin width of $w = 3$ ms, all the IPAT within the range $[0\ldots3)$ ms fall into bin 0, while an IPAT of 10 ms belongs to bin 3. In general an IPAT of i *ms* belongs to bin $k = \lfloor \frac{i}{w} \rfloor$.

Instead of considering all the distance combinations as in the original Hausdorff Distance algorithm, we propose to limit the distance computation to similar bins (in position) for the case of time distributions, which in practice represents comparing similar or equal IPAT values. The following definition formalizes the *Simplified Hausdorff* Distance.

Definition 2. Let us define σ as the bin offset and P, and Q two distributions, where P is the reference and Q the acquired distribution, each with n and m elements, respectively. We define the "*Simplified Hausdorff Distance*" as:

$$h_D(P,Q) = \max_{i=1,\ldots,n}\left\{\min_{j=i-\sigma}^{i+\sigma}\{g(P_i,Q_j)\}\right\}. \tag{2}$$

In Fig. 1, we clarify the above definition by describing the first iteration of the performed operations by *Hausdorff Distance* and *Simplified Hausdorff* Distance with a $\sigma = 2$. As it can be noted, our approach simplifies the computational complexity required to compute the distance.

As we shall show later in the evaluation section, this variant of the Hausdorff Distance greatly improves the accuracy in the detection of SLA violations, while yielding linear algorithmic complexity for small values of σ, i.e., $h_D(P,Q)$ is $O(n)$ for $\sigma \ll n$. We anticipate that $\sigma \ll n$ is precisely the case in our context.

4. SLA violation detection

Computing the distance among distributions determines how different the reference and the acquired traffic profiles are at the egress node. However, this information is not sufficient to perform the SLA assessment. In this section, we present the methodology used for detecting SLA violations, and how the latter is supported by the distance information described above. We start by presenting a general view of the methodology, and then provide a detailed description of each of its components.

4.1. General methodology

A simplified view of the methodology for detecting and reporting SLA violations is shown in Fig. 2. The methodology is composed of three interrelated algorithms (which are detailed throughout this section), where the main algorithm is divided into two blocks, namely, Block I and Block II.

In general terms, Block I computes the IPATs of a flow during a time period at the egress node, and compares their distribution with a *Reference Distribution Set* (RDS). If the distributions are *similar*, we assume that the network status is comparable, and therefore, no SLA violations are reported. Otherwise, Block II is activated, where we query the ingress node to acquire detailed packet information. More precisely, we use passive measurements to compute the real performance metrics on the network. Based on this, we assess the current status of the network and report any encountered SLA violations.

The strengths of this methodology are the following:

- Once the monitoring system is appropriately trained, the assessment process will generally remain in Block I. This means that the SLA assessment is performed integrally at the egress node, and therefore, neither the assessment of QoS metrics nor the exchange of control traffic is required.
- The transition from Block I to Block II is only invoked when the comparison of the current IPAT distribution with the RDS reveals significant differences, meaning that the network conditions are unknown. When this occurs, the SLA is accurately assessed by explicitly measuring the QoS metrics between the source and destination – notice that this approach is different from solutions that constantly estimate QoS metrics (e.g., using sampling techniques). These measurements are in turn used to dynamically retrain the system, and thereby, to update the RDS in case a valid IPAT distribution is found – the *validity* of a distribution is formally defined later in Definition 3, but informally, it refers to a distribution of IPATs mapping a QoS state that respects the SLA.

A more specific description of the general logic to detect SLA violations is shown in Algorithm 1. The detection works on a per flow basis *f*, by continuously monitoring the status of the SLA at predefined time intervals *t*. The empirical distribution is acquired in bins of width *w* (referring to IPAT ranges), the study of the proper *t* and *w* values is deferred to Section 7.

After the distribution acquisition during the time interval *t*, Algorithm 1 considers the following actions in order to cope with the potential variability of the IPATs:

- *Training and updating* the IPAT distribution set.
  - *Map* this set to the current network status.

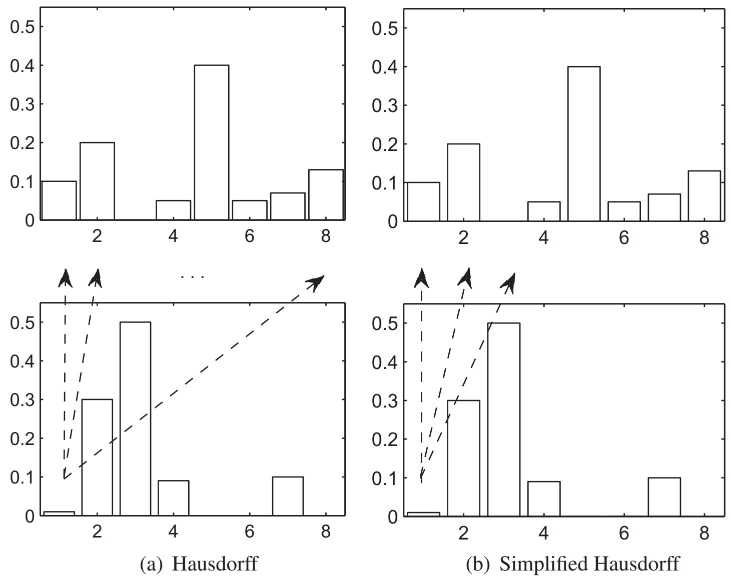


Fig. 1. *Hausdorff* and *Simplified Hausdorff* Distance comparison operations performed.

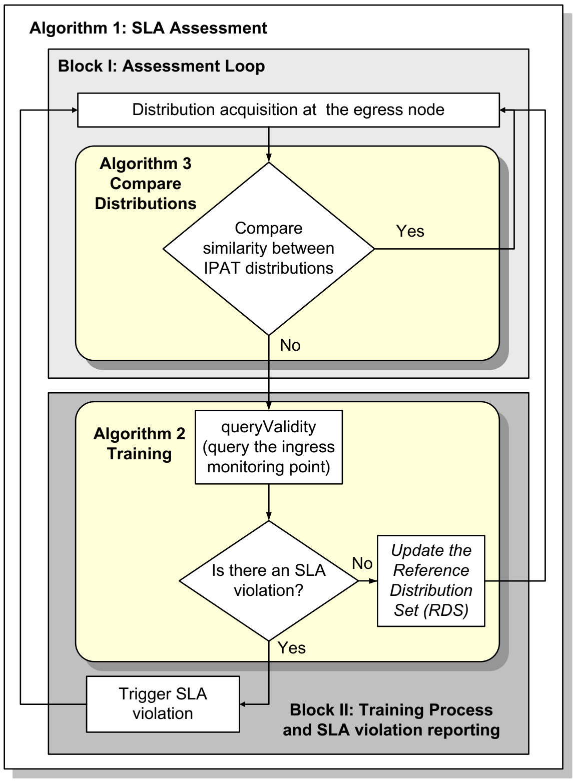

Algorithm 1: SLA Assessment
Block I: Assessment Loop
Distribution acquisition at the egress node
Algorithm 3 Compare Distributions
Compare similarity between IPAT distributions
Yes
No
Algorithm 2 Training
queryValidity (query the ingress monitoring point)
Is there an SLA violation?
No
Update the Reference Distribution Set (RDS)
Yes
Trigger SLA violation
Block II: Training Process and SLA violation reporting

Fig. 2. General methodology for SLA assessment.

  - *Update* the set of valid IPAT distributions if needed.
- *Comparing* the current profile with the learned status.
  - *Decide* whether the traffic conditions have changed or not.

The rest of this section performs a step-by-step description of the algorithms. We start by describing how the system learns the real quality provided by the network, and then we focus on how the system compares the IPAT distributions.

4.2. Training and update strategy

Any system with adaptability requirements must have a robust training mechanism. We start by introducing the concept of a valid IPAT distribution.

Definition 3. A *Valid IPAT Distribution* (VID) is a distribution for which a function $\mathcal{V}$ of the real metrics (OWD, IPDV, and PLR) falls above a specified SLA threshold v.

In our model, the training process queries the ingress monitoring point of the flow under analysis, and by means of passive measurements gets the *real* QoS parameters at the specified time interval t. As shown in line 3 of Algorithm 2, this is done by the *queryValidity*() procedure. Once the real metrics have been acquired, the destination assesses whether the SLA is respected or not. In our work, and as a proof of concept, we assume the simple and linear SLA compliance policy shown in (Eq. (3)).

4.3. Validity function $\mathcal{V}$

The QoS constraints might considerably differ depending on the type of traffic. For example, videostreaming is tolerant to large OWDs and high IPDVs, but not to packet losses, while videoconferencing is sensitive to all the metrics. Hence, we define ω_O, ω_I, ω_P as weights specified for each particular metric, where $\omega_O + \omega_I + \omega_P = 1$.

Algorithm 1. SLA assessment

```
    Input: $f$, $\mathcal{D}$ {$f$: Current Flow, $\mathcal{D}$: Global RDS}
    Output:status
    $S \leftarrow getFlowSourceMP(f)$ {Source Monitoring Point}
    $Q \leftarrow acquireDistribution(f, t)$
5:  if $\mathcal{D} = \varnothing$ then
      $status \leftarrow Training(Q, S)$ {see Algorithm 2}
    else
      $status \leftarrow compareDistributions(Q, S)$ {see Algorithm 3}
    end if
10: if $status < v$ then {Unacceptable QoS conditions}
      trigger $SLAViolation(status)$
    end if
```

Expression (3) represents the degree of validity of the QoS for a time interval t.

$$\mathcal{V} = \mathcal{Q}^O\left(\overline{OWD}\right) \cdot \omega_O + \mathcal{Q}^I\left(\overline{|IPDV|}\right) \cdot \omega_I + \mathcal{Q}^P(PLR) \cdot \omega_P. \quad (3)$$

In (3), $\mathcal{Q}^*(x)$ determines the quality degrading function, which we define as:

$$\mathcal{Q}^*(x) = \begin{cases} 1, & x \leqslant \mathcal{X}, \\ \lambda e^{-\lambda(x-\mathcal{X})}, & \mathcal{X} < x < \mathcal{M}, \\ 0, & \text{otherwise}, \end{cases} \quad (4)$$

where $\mathcal{X}$ is the metric dependent threshold of quality degradation specified by the SLA, and $\mathcal{M}$ the upper feasible bound for the quality of that particular metric. Finally, λ in $(0,1)$ is the decaying factor for the exponential quality degradation function.

In (3), $\mathcal{V}$ is defined in the range $[0,1]$ indicating the quality of service experienced by the flow with respect to the SLA. In the range above, 1 stands for perfect quality, while 0 is absolute lack of it. To compute this value we consider a linear combination of the usual metrics (OWD, IPDV, and PLR), but clearly, any other function $\mathcal{V}$ can be applied seamlessly.

When $\mathcal{V} \geqslant v$ the network behavior is considered stable, where v is the specified quality threshold of the system. The closer v is to 1, the stricter our system will be to SLA violations.

In summary, the performance metrics are used to map the IPAT distribution to the real network status, and we use such distribution as a reference to infer the SLA compliance.

Definition 4. A *Reference Distribution Set* (RDS) $\mathcal{D}$ is a set of strictly different VID distributions, where $|\mathcal{D}|$ is the cardinality of the set, and $\{\mathcal{D}^1, \ldots, \mathcal{D}^{|\mathcal{D}|}\}$ are the elements of the set. The size of the RDS is bounded by a predefined Δ, which limits its maximum memory usage.

The *Training and Update Strategy* is in charge of keeping an updated and valid version of the RDS – the details are shown in Algorithm 2.

Algorithm 2. Training

```
    Input: Q,S {Q: IPAT distribution, S: Flow Source}
    Output: status
    status ← queryValidity(Q,S) {Computes Eq. (3) and generates control traffic by acquiring the real metrics}
    if status < $v$ then
5:  return status {Do not keep Q in RDS, SLA violation}
    end if
    if $|\mathcal{D}| \geqslant \Delta$ then
      expireLatestNotUsed($\mathcal{D}$) {Expiration policy}
    end if
10: $\mathcal{D} \leftarrow \mathcal{D} \cup Q$
```

A prerequisite of the RDS is that all the stored distributions must represent good reference traffic conditions to be compared with. Therefore, our system must have some means for correctly assessing them. In order to do so, we use the technique presented in [4], where the source measurement point (i.e., the ingress router) sends per packet information, such as transmission timestamps; the timestamps are matched on the destination measurement point (i.e., the egress router), for computing the relevant performance metrics. This technique requires to generate some control traffic from source to destination to compute the metrics, such control traffic determines the amount of bandwidth resources required by the system, and for the sake of efficiency and scalability, it should be minimized. Nevertheless, we use this method because it reports reliable values of the network metrics to be mapped to the reference IPAT distribution.

Algorithm 3. compareDistributions

```
    Input: Q, S {Q: Acquired Distribution, S: Flow's Source}
    Output: status
    minD ← ∞
    for all D ← $\mathcal{D}$ Do
5:  d ← computeDistance(D,Q)
    if d < minD then
      minD ← d
      P ← D
    end if
    end for
10: if min D $\geqslant \delta$ then
      status ← Training(P,S) {Does metric computation}
    else
      updateUse(P) {Renew usage of the distribution to prevent expiration when $|\mathcal{D}| = \Delta$}
15: status ← getValidity(P) {Use value of queryValidity}
    end if
```

Once the *real* validity is assessed, if it is below v, the event is registered, the distribution discarded, and a *SLA-Violation* event is triggered (see Step 11 in Algorithm 1). Otherwise we insert the distribution into the RDS. Algorithm 2 shows that when $\mathcal{D}$ is full, we discard the oldest unused distribution. We propose this simple replacement algorithm for two different reasons: (i) it is very efficient and easy to implement; and (ii) it honors the fact that if a distribution has not been representative of the traffic profile for a while, it is due to changes in the network status; so this distribution is not required anymore.

4.4. Distribution comparison

The complete pseudo-code for the comparison between distributions is detailed in Algorithm 3. As mentioned above, the metric we chose for comparing two distributions is a distance, which can be described as *how far* one distribution is from another, or preferably, as the degree of similitude (dissimilitude) between two distributions. The bigger the distance, the more different are the distributions (and so the traffic profile, and the network performance).

Since the RDS is composed by a set of distributions, the distance cannot be directly computed. Hence, we define:

Definition 5. *The degree of Matching D_M* between a RDS $\mathcal{D}$ and another distribution Q is defined as:

$$D_M(\mathcal{D}, Q) = \min\{d(p, Q)\}p = \mathcal{D}^1 \cdots \mathcal{D}^{|\mathcal{D}|}, \tag{5}$$

where $d(p,Q)$ is the distance between the distributions p and Q.

Notice that $d(p,Q)$ can be computed using the distances defined in Definitions 1 and 2, or any other distance metric.

Then, Q and $\mathcal{D}$ are considered similar if $D_M(\mathcal{D}, Q) \leqslant \delta$, where δ is our distance threshold. The critical point here is that different distributions do not mean different qualities, since the traffic profile can change over time, even with the same quality level. Therefore, when the distributions are considered different, the system must learn the new degree of performance of the network. Thus the *Training* procedure is then invoked with Q (see the transition from Block I to Block II in Fig. 2). As described previously, *Training* consumes system resources, and therefore, there is a trade-off since the lower the value of δ, the more resources (queries) will be needed. On the other hand, the higher the value of δ, the lower amount of resources will be required, though at the expense of losing some accuracy in the SLA assessment. Section 7.3 provides an extensive discussion about the effects of changing δ.

5. Tests and testbeds

In order to validate our proposal we set up three different testbeds: (i) synthetic traffic under controlled testbed conditions, (ii) synthetic traffic over the European Research Network Géant, and (iii) real traffic over a controlled testbed.

5.1. Synthetic traffic under controlled testbed conditions

The first set of tests has been performed under a tightly controlled environment. We configured two end nodes with Linux Debian in order to generate and collect traffic. On the core of the testbed we installed two Linux Debian servers, with Traffic Control and `NetEM`[1] emulation capabilities. This configuration allows us to change the network conditions according to our needs, and experience a wide range of controlled network disruptions.

All the links on the testbed were dedicated Fast Ethernet. Furthermore, all the Linux boxes were synchronized by GPS with the NTP and PPS patches on the kernel for accurate timestamping.

In this testbed, the set of emulated network conditions are:

(1) *Good Network Conditions:* no SLA disruptions and good network behavior all over the test.
(2) *Mild Network Disruptions:* moderated increase of OWD with periods of high IPDV and some packet losses. Some traffic disruptions, but only with a few SLA violations per test.
(3) *Medium Network Disruptions:* similar to the mild network disruptions but with limited buffers on the routers, which lead to moderate periods of packet losses. Some SLA violations in many intervals during the test.
(4) *Severe Network Disruptions:* random losses from 1% to 10% with variable OWD. Severe SLA violations in periodic intervals over the test.

All the tests performed have in common that the SLA disruptions are applied at regular time intervals, combining periods of good behavior with others with disruptions.

We performed tests with *Periodic*, *Poissonian* and *Synthetic Real Traffic* [22] traffic profiles with all the above network conditions. Using these traffic profiles we have from predictable packet rates (*Periodic*) to unpredictable and realistic profiles (*Synthetic Real Traffic*).

5.2. Synthetic traffic over the European Research Network

In this testbed, we performed more than 500 experimental tests using 12 different testbeds across Europe. We performed the tests at different hours, including weekends, to have a good diversity of cross traffic and congestion levels. The testbeds were provided by the IST-EuQoS (http://www.euqos.eu) partners, covering a total of five countries and four different access technologies (LAN, xDSL, UMTS and WiFi), supported by an overlay architecture over the Géant European Network.

We evaluated the performance of our monitoring system by actively generating UDP traffic on the network with different properties. Specifically, we generated periodic flows, with varying packet rates, from 16 to 900 packets per second among all the involved nodes in the testbed. We used different packet sizes ranging from 80 to 1500 bytes per packet. More specifically, we focus on three different sets of tests. The first one simulates a low rate communication, with small size packets using 64 Kbps of bandwidth. We label this tests as (synthetic) `VoIP`. The second type of traffic is a periodic flow with average packet rate of ~96 packets per second, with MTU sized packets amounting to a total of 1 Mbps of UDP traffic. We call this test `UDP-1`. Finally, the third kind of traffic is an average sized, high rate UDP flow with ~1.4 Mbps, we call this test `UDP-2`.

5.3. Real traffic over a controlled testbed

Generating synthetic traffic gives tight control over the different characteristics of the traffic: rate, packet size, etc., but on the other hand, it does not reflect how a real application performs. Therefore, in order to have insights about the behavior of our system with real applications, we used the local testbed described in Section 5.1 with a video streaming application, namely VLC, transmitting high quality video with variable bit rate over the network. In the same way as before, we inserted various levels of disruptions to analyze the accuracy of our SLA assessment system.

6. Evaluation

The validation of the proposal focuses on the assessment of the system's accuracy, i.e., the SLA violation detection rate, measured in terms of false negatives (i.e., undetected SLA violations). It is important to notice that false positives cannot happen in our design, since we always query the network performance related to unknown IPAT distribution, verifying all the detected SLA violations against the network ingress point.

We compare the accuracy of the two presented distance algorithms (i.e., *Hausdorff* and *Simplified Hausdorff* Distance) and *Kullback–Leibler Divergence* presented in [19], against the case for which we have perfect knowledge about the SLA violations. In particular, we analyze the ratio of detected SLA violations against their total number. We also analyze the amount of resources required by the system; such resources are counted in terms of the reduction ratio of the bandwidth used by the control traffic. In particular, we compare the cost of reporting information on a per packet basis, against our solution, which only demands information when there is a change in the profile of the received traffic.

All the analysis in this section uses the same set of parameters. Specifically, we set up, as a proof of concept, the following values: distance threshold of $\delta = 3\%$, bin width of $w = 3$ ms, and an acquisition time interval of $t = 175$ ms. Section 7 includes an experimental analysis about the effects of changing these parameters.

6.1. Methodology

Analyzing all the information obtained from the tests is complex. To ease the comprehension of the validation process, we unify the evaluation for all the tests and testbeds under the same methodology as follows:

[1] More information about `NetEM` can be found at http://tcn.hypert.net/tcmanual.pdf.

(1) For each test we collect the full trace on both end nodes.
(2) We extract the network performance metrics as described in [4] on a per packet basis, using them as a reference of the quality (perfect knowledge), since the process gives exact values for the metrics, and by extension to the SLA violations through Eq. (3).
(3) We identify the different SLA violation periods with the reference results acquired above.
(4) In addition we apply the *Kullback–Leibler*, *Hausdorff* and *Simplified Hausdorff* algorithms, where we record: (i) required control traffic due to *Training* (caused by the queries to the ingress node); (ii) estimated SLA violation periods.
(5) Finally, we evaluate the matching between the SLA violations and the ones obtained in Step 3.

6.2. Accuracy and resources requirements

In order to study the behavior of our system, we now discuss the achieved accuracy together with the analysis of the required resources for each algorithm in the different testbeds. We defer to Section 6.3 the study of the statistical validity of our results.

6.2.1. Synthetic traffic with controlled network

The goal of this synthetic traffic generation is to evaluate the performance of each algorithm in a controlled environment with the different traffic profiles.

In Table 1, we analyze the accuracy and the resource utilization for the different traffic traces. The accuracy is computed for the overall test duration, counting the ratio of detected SLA violations over the total. On the other hand, the required resources are computed by the ratio of the actual number of queries over the maximum possible queries per test, caused by the per packet reporting. Our goal is to achieve high accuracy with low resource consumption.

As it can be observed in the table, the accuracy of the solution is higher for the extreme cases. When there are *Good* network conditions in the network we always estimate correctly, and with low resource consumption in general. This is because our algorithm assumes correct network behavior by design. In the case of *Severe* network conditions, where our contribution is more useful, we can detect with very good accuracy the SLA disruption periods. On the other hand, when studying the tests with few SLA violations we found out that most of the disruption periods were shorter than one bin (i.e., shorter than 175 ms). In Table 1, we show the results after removing these outliers, since in a real deployment, such SLA violations are of no practical interest. They represent very short and sporadic periods of light congestion, which are not only humanly imperceptible, but also do not reflect the nominal quality offered by the network. Nevertheless, in Section 6.3, we study the statistical confidence intervals of the system's accuracy considering all the outliers, this way we analyze the overall statistical validity of our proposal.

Comparing the various distance algorithms we can notice some general properties: first, the better accuracy in most cases is achieved by the *Simplified Hausdorff* Distance proposed as an extension in this paper, with similar results for *Kullback–Leibler*. It is also interesting to highlight the comparatively poor performance obtained with *Hausdorff*, except in the case of synthetic real traffic with *Mild* SLA violations. This is caused by the "*all-against-all*" comparison we pointed out previously as the source of inaccuracy of the algorithm.

The second observation is regarding the amount of resources needed by the *Severe*, and some *Medium* network conditions. As it can be noted, for some traffic profiles, the results are exactly the same regardless of the algorithm used. This is because the algorithms always query the ingress node when an unknown IPAT distribution is found; this is a common behavior for all algorithms when network disruptions are encountered. Hence, it forces the system to query for the network metrics (in this case, the minimum number of these queries is bounded by the amount of SLA violations). In our experimental case this is 0.397 as shown in Table 1. In the specific case of Poissonian Traffic, we need more resources than this lower bound for *Kullback–Leibler* and *Simplified Hausdorff*. Notice though, that requiring less resources than that would imply not detecting some SLA violations.

In absolute terms, for each one of the generated flows in the tests, the use of bandwidth resources in the case of per packet reporting implies an average of 57.6 Kbps in control

Table 1
Accuracy and resources for $\delta = 0.03$.

	Accuracy				Resources			
	Good	Mild	Medium	Severe	Good	Mild	Medium	Severe
(a) Periodic								
Kullback–Leibler	1.000	1.000	0.987	1.000	0.001	0.256	0.385	0.394
Hausdorff	1.000	1.000	0.088	1.000	0.001	0.020	0.019	0.394
Simplified Hausdorff	1.000	1.000	0.868	1.000	0.001	0.130	0.267	0.394
(b) Poisson								
Kullback–Leibler	1.000	1.000	0.940	0.893	0.562	0.572	0.657	0.671
Hausdorff	1.000	1.000	0.067	0.225	0.122	0.143	0.132	0.190
Simplified Hausdorff	1.000	1.000	0.994	0.999	0.464	0.601	0.699	0.721
(c) Synthetic real traffic								
Kullback–Leibler	1.000	1.000	1.000	1.000	0.002	0.002	0.397	0.397
Hausdorff	1.000	1.000	1.000	1.000	0.002	0.003	0.397	0.397
Simplified Hausdorff	1.000	1.000	1.000	1.000	0.002	0.002	0.397	0.397

traffic per flow (having an average of 200 packets per second), while in the case of using *Simplified Hausdorff* Distance, the required bandwidth is 9.7 Kbps in average (considering the average resource consumption over all the tests), which is a gain of ∼80%. It is worth noting that this use of resources occurs during the *Training* periods, which in general are more intensive during the first seconds of the test.

6.2.2. Synthetic traffic over the European Research Network

By means of this testbed, we evaluate the accuracy of our proposal in a real network with random quality, unexpected events, and unknown cross traffic with different multi-hop paths.

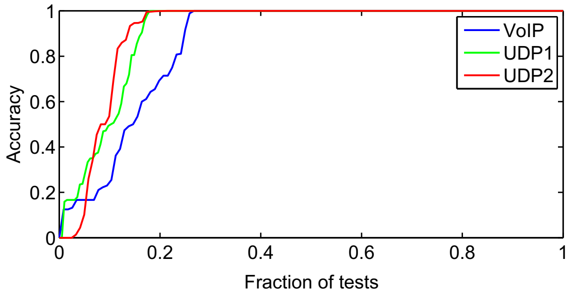
In Fig. 3, we show the Cumulative Density Function (CDF) of the accuracy for each algorithm and traffic profile. The *Y*-axis of the figure represents the accuracy, and the *X*-axis shows the test number, normalized to 1, showing the amount of tests with a given accuracy. The figure considers all the tests, including the ones without SLA violations. The results show that for both *Kullback–Leibler* and *Simplified Hausdorff* we get perfect accuracy for more than 70% of the tests. For the rest of the tests, it can be observed that `UDP-2` tests achieve better accuracy in general given that they have higher packet rate.

We complement the figure with Table 2, which summarizes the numerical values of these experiments. We show the aggregated total amount of periods with violations, together with the amount of such periods that our algorithm was able to detect. In the third column, we highlight the overall accuracy, and finally, the last column, details the average amount of resources needed for the reporting.

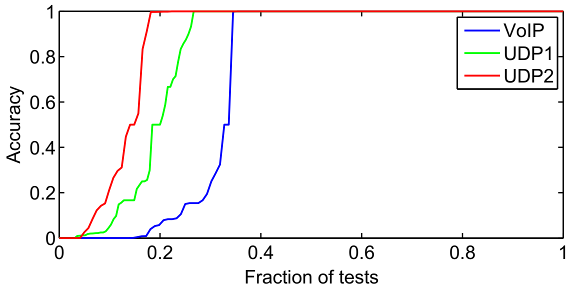
Once again we must notice that most of the failures in the SLA estimation are due to isolated violations, which we consider as outliers and remove them from our study, given that they are very close to the SLA agreement boundary with no perceptible effects. In fact, these not detected violations occurred only for a single bin (i.e., 175 ms in our studies), and most specially to the flows with low packet rate (i.e., `VoIP`), with little impact to the end-user perception. We complement the accuracy study in Section 6.3, where we discuss the statistical confidence bounds of the accuracy obtained by the system.

In this set of tests, the best trade-off is achieved by *Simplified Hausdorff* as it provides almost the same accuracy levels as *Kullback–Leibler* for `UDP-1` and `UDP-2`, with much better accuracy for `VoIP`, but also using significantly lower resources, comparable to *Hausdorff* in all the cases assessed.

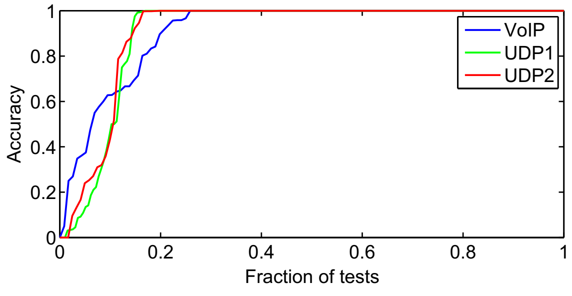
In terms of resources, the average resource usage of the whole system is below 25%. It is lower when considering `VoIP` traffic (around 4%), while the minimum amount of resources required is $\sim 5.8 \times 10^{-4}$, and the maximum is 1 (meaning no reduction in control traffic compared to the per packet reporting is achieved). Further investigating the tests causing such resource usage, we found out that they are the ones representing highly congested links (in particular xDSL), with very high loss ratios and large amount of SLA violations (higher than 90% of the bins were severely affected by packet losses). This forced a large increase in the resource requirements in these specific cases as expected.



(a) Kullback-Leibler



(b) Hausdorff



(c) Simplified Hausdorff

Fig. 3. Accuracy for synthetic traffic over the European Network.

Table 2
Violation detection under a real network, $\delta = 0.03$.

	Violations	Detected	Accuracy	Resources
(a) Kullback–Leibler				
VoIP	5036	3930	0.780	0.175
UDP-1	62,264	60,108	0.965	0.551
UDP-2	22,540	22,030	0.977	0.335
(b) Hausdorff				
VoIP	5036	3163	0.628	0.024
UDP-1	62,264	58,003	0.932	0.237
UDP-2	22,540	21,189	0.940	0.219
(c) Simplified Hausdorff				
VoIP	5036	4668	0.927	0.043
UDP-1	62,264	59,265	0.952	0.246
UDP-2	22,540	21,513	0.954	0.224

Table 3
Overall detection and resources for VLC traffic, $\delta = 3\%$.

	Overall detection				Resources			
	Good	Mild	Medium	Severe	Good	Mild	Medium	Severe
(a) Kullback–Leibler								
Audio	1.000	1.000	0.999	0.997	0.962	0.966	0.947	0.728
Video	1.000	0.750	0.703	0.909	0.061	0.074	0.085	0.569
(b) Simplified Hausdorff								
Audio	1.000	0.999	0.979	0.975	0.114	0.114	0.144	0.397
Video	1.000	0.950	0.844	0.871	0.032	0.035	0.050	0.407

6.2.3. Real traffic over a controlled testbed

In this last set of tests, we observe the performance of our algorithms under real traffic with controlled network conditions. In fact, the forced SLA violations on the testbed follow the same patterns as we introduced in the synthetic traffic case (i.e., *Good*, *Mild*, *Medium* and *Severe*).

We used VLC to perform the tests. Since the application generates two flows, one for audio and the other for video, we show them separately in Table 3. There, we can see the overall accuracy and the resource usage for *Kullback–Leibler* and *Simplified Hausdorff*, omitting *Hausdorff* given its lower accuracy.

The accuracy for *Simplified Hausdorff* in detecting the studied SLA violations is higher than 97% for the audio flows, dropping sensibly in the case of video flows, but always with accuracy ratios higher than 84%. The causes of such difference in accuracy are yet unknown and subject to further study. However, it is worth noticing that, similarly to the experiments performed in the European Research Network, our methodology shows consistent results across all the tests.

Regarding the resources, for audio flows, in the case of *Simplified Hausdorff*, they range from $\sim$11% for *Good* network behavior to $\sim$40% in the case of *Severe* SLA disruptions. For the video flows it ranges from $\sim$3% to $\sim$40%. It can be noted that the cases for which the maximum resources are required are consistent with the ones found on the Synthetic Traffic with Controlled Network: the packet losses and delay constraints were similar in these two cases. Again, using *Kullback–Leibler*, despite of having slightly better accuracy, requires $\sim$20% more resources than *Simplified Hausdorff*.

For the case of absolute bandwidth usage, in this case, per packet reporting implies an average of 147.3 Kbps in control traffic. Analogously to the previous set of tests, this usage is centered around the *Training* periods, while using our system the requirements fall down to 38.7 Kbps (obtained from averaging the resources in all the flows).

6.3. Statistical confidence of the accuracy

So far we have highlighted the accuracy and the reduction in resource requirements of our framework, but as we discussed before our results present some outliers. To prove the validity of our methodology, in this section, we obtain statistical confidence intervals to the accuracy of the results in our experimental analysis. In order to simplify the description, we limit our study to *Simplified Hausdorff* Distance.

In order to provide confidence intervals of the average accuracy obtained by our system we use the well-known bootstrapping technique presented in [23]. To this end, let $\hat{\mu} = \frac{X_1+\cdots+X_n}{n}$ be the mean accuracy obtained by our system, the confidence interval is then given by $Pr(\hat{\mu}_L \leqslant \mu \leqslant \hat{\mu}_U) = 1 - \alpha$, where $\hat{\mu}_L$ and $\hat{\mu}_U$ are the lower and upper bounds of μ with probability $1 - \alpha$, normally $\alpha = 0.05$, providing a confidence interval for the mean of 95%. In a bootstrapping procedure, to determine these bounds we must decide the bootstrapping parameters, namely, we define the resampling value A, i.e., the number of randomly chosen tests, and B the bootstrap iterations.

In our practical case, of all the performed tests we only select the ones having SLA violations, which represents the lower accuracy achieved by our system; recall that if there are no SLA violations our quality estimation is always correct. Therefore, our samples $\mathcal{S} = \{X_1, \ldots, X_n\}$ are the computed accuracies for each test, the resampling value is $A = 300$, which means that we draw A bootstrap samples with replacement from $\mathcal{S}$ obtaining $\mathcal{S}_1^* = \{X_1^*, \ldots, X_S^*\}$. Then we compute the mean of $\mathcal{S}_1^*$, i.e., $\hat{\mu}_1^*$. We repeat this bootstrap procedure $B = 10{,}000$ times, leading to $\hat{\mu}_1^*, \ldots, \hat{\mu}_B^*$. Therefore, the approximation of the distribution is obtained by sorting the different $\hat{\mu}_n^*$, obtaining $\hat{\mu}_{(1)}^* \leqslant \hat{\mu}_{(2)}^* \leqslant \hat{\mu}_{(B)}^*$. Then our confidence interval is $(\hat{\mu}_L, \hat{\mu}_H)$, where $\hat{\mu}_L = \lfloor B\alpha/2 \rfloor$, and $\hat{\mu}_H = N - \hat{\mu}_L + 1$. Leading to $\hat{\mu} = 0.88$ and $(0.85, 0.90)$ accuracy bounds, with a 95% confidence interval.

To complete the study, we reinserted all the tests without SLA violations to the bootstrap process and recomputed the accuracy and its confidence levels, leading to $\hat{\mu} = 0.94$ and a confidence interval of $(0.92, 0.95)$, which is the expected accuracy on networks where service disruptions are mixed with proper service delivery.

Even if the obtained average accuracy can be improved, as we argued before, most of the results reducing the accuracy represent mild service disruptions very close to the quality threshold which are of no practical interest in a QoS assessment deployment.

7. Parameter selection analysis

In this section, we introduce an experimental analysis of the effects of varying the various parameters of the system. We perform tests by changing the three main parameters, namely, the time interval t, the bin width w and the threshold δ. The full analytical study of the impact of such parameters on the system performance can be found in [24].

7.1. The acquisition interval *t*

As we discussed previously, increasing the acquisition time interval gives our system more statistical soundness, due to the presence of more samples in the acquired distribution, though at the expense of reducing the responsiveness of our system.

We performed the analysis by doing tests with several values of *t*. In particular we set up our VLC testbed to use the following values of *t*: 50, 100, 175, 300, 500, 1000, 2000 and 3000 ms. Then we estimated the SLA violations by using our three algorithms.

Fig. 4 summarizes the obtained results for all these values. Among the results, we omit the *Good* quality tests, since the accuracy is always 1 and the resource consumption is in all cases lower than the other levels of disruption.

The results show the expected behavior in all the cases. The trend is a clear improvement in the accuracy of the system as the time interval increases.

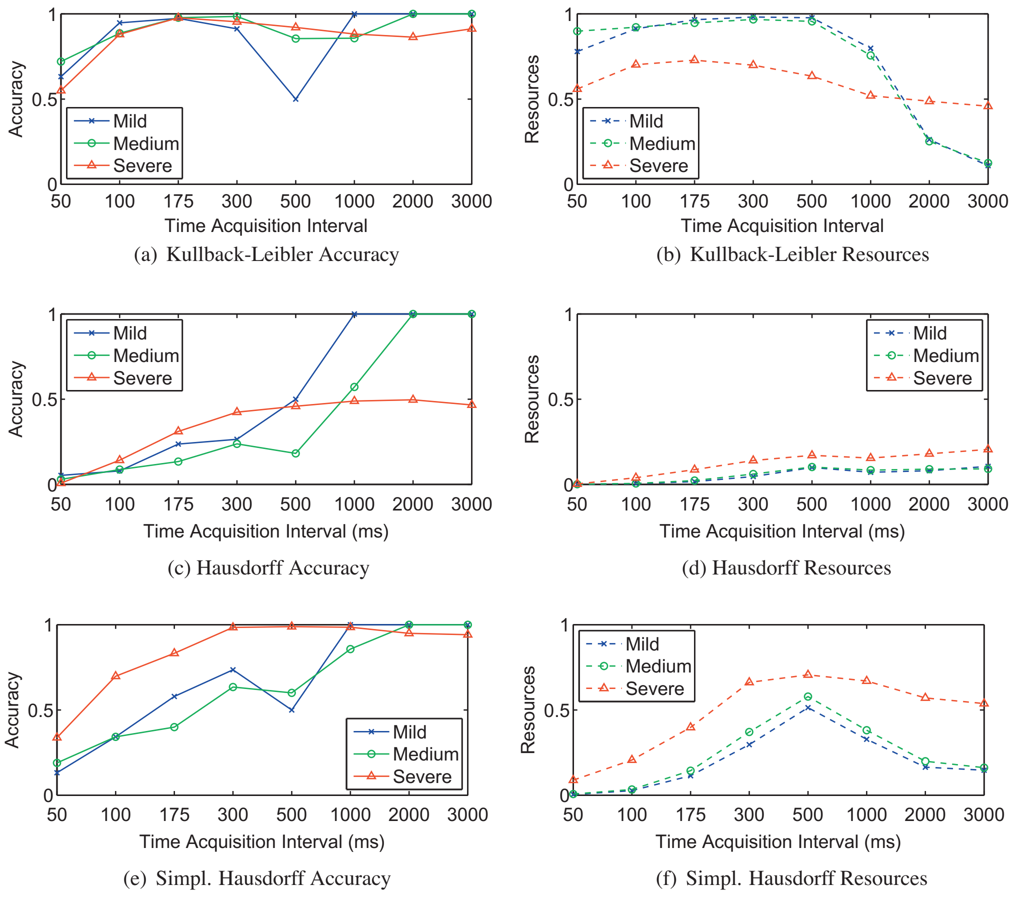
As it can be noted there is a considerable drop in accuracy for audio traffic in the case of *Simplified Hausdorff* and in *Kullback–Leibler* for *Mild* traffic conditions (in fact it drops to 50%, in the specific case of 500 ms). The cause for this is that when having longer bins, the offered service is in average very close to the quality threshold, reducing the number of bins with violations.

As we already shown in Section 6, both *Simplified Hausdorff* and *Kullback–Leibler* have better accuracy than *Hausdorff*. By comparing both algorithms, we noticed that, in general, *Kullback–Leibler* has, in some cases, slightly better accuracy, but requires too many resources in terms of bandwidth. In any case for high values of *t*, *Simplified Hausdorff* uses a fairly reasonable amount of resources, while accomplishing a similar degree of accuracy.

Summarizing the findings about the accuracy, we can see that in general having large bin widths helps improving the estimation. However, this is rather misleading, since for large timescales, short-duration disruptions remain undetected, while they might be relevant for end-user applications. This is not a weakness of our monitoring system but of the actual reduction in the packet loss ratio per time interval (therefore that bin has a validity higher than *v*).

7.2. Impact of the bin width *w*

Changing the bin width is critical to the accuracy of the system. Now we study its effects from the experimental point of view. As a proof of concept we focus our study



(a) Kullback-Leibler Accuracy (b) Kullback-Leibler Resources
(c) Hausdorff Accuracy (d) Hausdorff Resources
(e) Simpl. Hausdorff Accuracy (f) Simpl. Hausdorff Resources

Fig. 4. Time acquisition interval effect over VLC audio traffic.

on the *Real Traffic over a controlled testbed*, but our analysis could be extended to any other type of traffic in our system with similar results.

The analysis is performed by using w values from 1 to 10 ms of IPAT. In the rest of the section, we analyze these different values with the three distance algorithms, to study the effect on the accuracy and used resources.

In Fig. 5, we show the results with values of $t = 175$ ms and $\delta = 3\%$, for the different w. The results highlight only the accuracy and resources of the audio flows. Similar results are obtained with the video stream.

It can be noted that the intuitive idea of resource consumption and accuracy is held in all the cases in general, i.e., the bigger the bin width the lower the accuracy with less resource consumption, specially in *Simplified Hausdorff*.

For *Kullback–Leibler*, in the worst case the accuracy in the SLA Violation Detection drops from 1 to 0.89 for the *Mild* case in audio flows. In the results obtained for *Medium* and *Severe*, the accuracy is constant in all the cases.

As expected the resource consumption for *Kullback–Leibler* with audio flows is consistent, reducing in ∼0.10 units from $w = 1$ to $w = 10$.

In the case of *Hausdorff*, the decrease in accuracy is more noticeable in the *Severe* case. It is fairly well estimated for small bin widths but its accuracy drops very fast as the bin width increases. This highlights the point that increasing the bin width helps reducing the resolution, i.e., the system considers similar groups of IPAT which, in reality, are very different. Moreover, here we can see with more detail the lack of accuracy provided by the "*all-against-all*" comparisons used by *Hausdorff*.

Regarding the used resources, their consumptions are slightly reduced due to the increase of the bin width, but it is far less noticeable than in the case of *Kullback–Leibler*. This is mostly caused by the fact that the bin width increase favors a reduction of the resources, while the casual increase in accuracy favors the use of more resources as more queries are issued.

Finally, in the case of *Simplified Hausdorff* the obtained accuracy is reduced as the w increases. One interesting point to notice is that the bin width impacts more strongly the *Simplified Hausdorff* than the *Kullback–Leibler* algorithms. The reason is that geometric distances are more deterministic (they are a real metric indeed) than the rela-

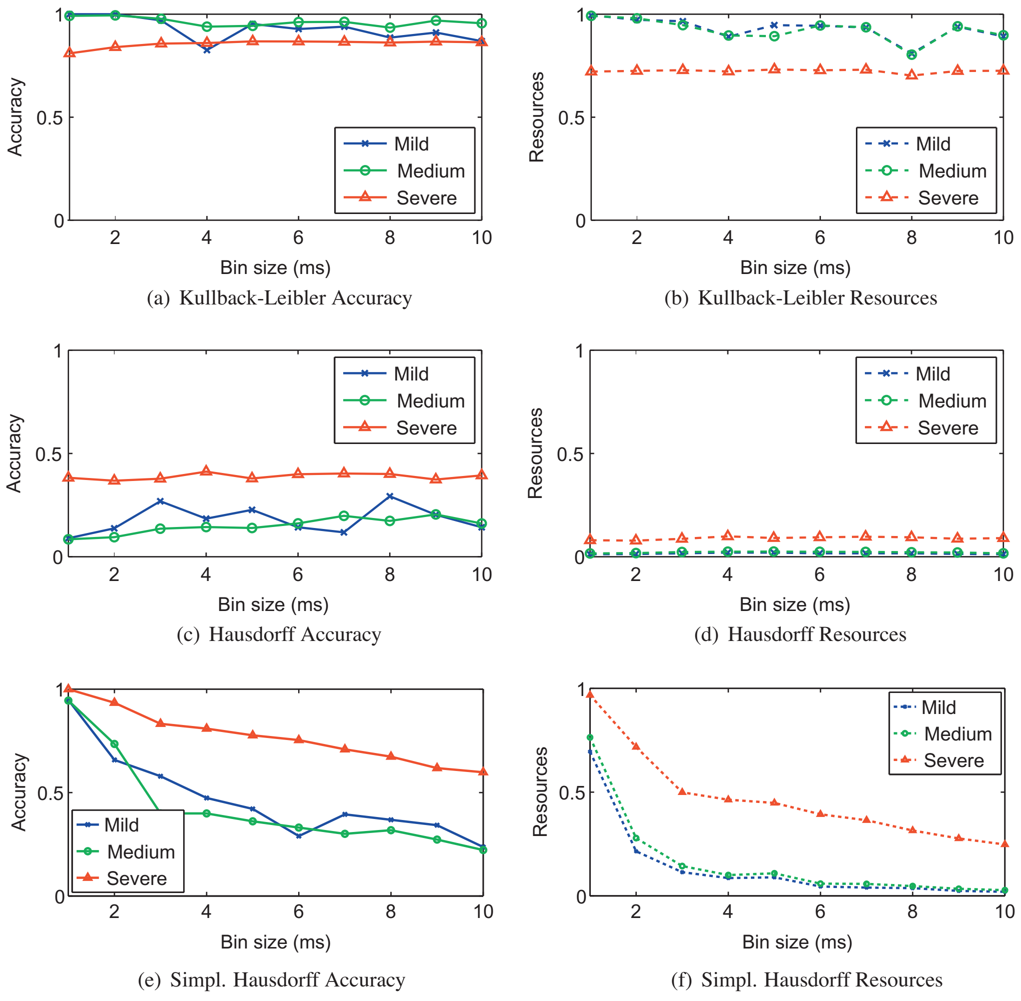


Fig. 5. Bin size effect over VLC audio traffic.

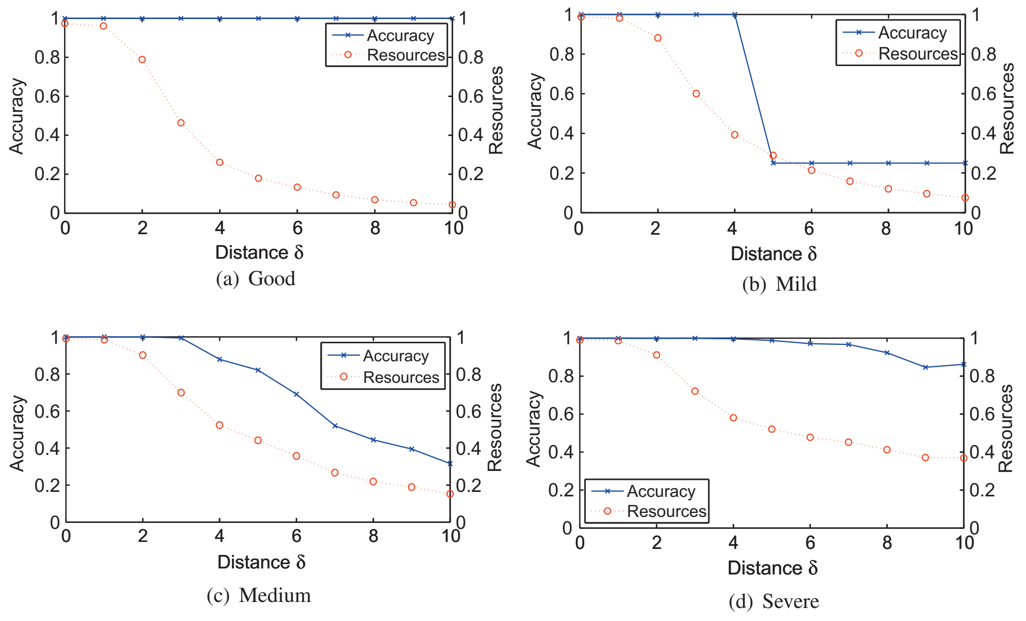


Fig. 6. Distance effect Poisson traffic. Accuracy and resources are on the *Y*-axis and each considered distance (in %) is on the *X*-axis.

tive entropy used by *Kullback–Leibler*, which, being dimensionless, does not have this spatial difference found on physical metrics.

7.3. Sensitivity analysis for δ

As we already discussed, querying the other end-point to acquire the network status is one of the bottlenecks of the system. The query is triggered when $D(\mathcal{D}, Q) \geqslant \delta$ (Step 11 of Algorithm 3). Hence, δ and the traffic itself determine the amount of queries of the system. Fig. 6 details the effects of changing δ between 0% and 10% for controlled traffic generation with Poissonian Traffic using our *Simplified Hausdorff* Distance based algorithm (the study would be similar for the other algorithms).

All the subfigures contain the Accuracy in the left *Y*-axis with a solid line, and the Resources needed on the right *Y*-axis with a dotted line. The *X*-axis contains the different values for δ. As it can be observed, in the case of Good and Severe traffic conditions, the effects of increasing the thresholds permits the reduction of the required resources given the predictability of the outcome, especially in the Good case where the estimation is always correct. Again the important trade-off is on the fuzzy cases, for which, not many SLA violations occur on the network. In that case, increasing the thresholds has a very noticeable effect on the final accuracy of the system, due to the statistical error incurred when having a small number of samples. It becomes more noticeable when there are fewer disruptions (*Mild* case in the figure).

8. Discussion

In this section, we complete our analysis by studying the cost of the solution in terms of computational demands. We also discuss the required steps to deploy our system in a real network, and estimate the bandwidth requirements of deploying our system in such scenario.

8.1. Overall computational cost of the Simplified Hausdorff distance algorithm

To assess the feasibility of deploying the system we perform an analysis of the cost of the overall algorithm. All the expressions in this section are considered with a time interval resolution t.

The function *acquireDistribution* has a linear cost over the number of received packets n per flow in t, so *acquireDistribution* is $O(n)$.

The critical path for the *Training* function is querying the validity of a distribution. This implies computing the *OWD* and the *IPDV* of the time interval, which has linear complexity over the number of packets evaluated. Thus, *Training* is also $O(n)$.

For the function *compareDistributions*, the cost consists of:

- Linear cost over $|\mathcal{D}|$ for the iterator of RDS.
- The cost of computing the distance. As described before, it is linear over the number of bins in the distribution ($|P|$).
- *Training*, which has linear cost as described before.

The total computational cost is thus upper bounded by $O(|\mathcal{D}| \cdot |P| + n)$, knowing that it is not possible to have two *Training* periods in one round of the algorithm.

8.2. Deploying our solution in a real setting

All the basic steps of the algorithm are memory efficient and easy to implement in edge nodes of the network. They involve computing IPAT distances between distributions. The most complex task is to provide feedback to the system about the real metrics of the network, since it requires to match packets for the flows under analysis at the ingress and egress nodes. This can be performed by using different techniques as described in our previous work [4,5], e.g., through dedicated boxes forming an overlay above the network under analysis.

The scalability improvement of our system is twofold. First, because of the reduction in terms of bandwidth, and second because most of the time we avoid computing QoS metrics. Regarding the cost in terms of bandwidth required to perform the assessment, we have collected traces of real traffic on the Catalan Academic Network, which has a Gigabit Ethernet link with sustained traffic of around 360 Mbps, and a peak at 483 Mbps. These traces contain an average of $\sim$3500 flows per bin interval. These values can be considered as upper bounds because, in general, not all the traffic on a link is constrained by SLA agreements. In this test, the original per packet reporting mechanism requires an average of 8.47 Mbps, while under normal network conditions our *Simplified Hausdorff* Distance algorithm requires only 1.35 Mbps – as described earlier in Section 6, we consider normal network conditions the case where the bandwidth usage is reduced to a 15%.

In the case of the second scalability improvement obtained by our approach, i.e., most of the time we avoid completely the computation of QoS metrics. This derives in considerable reductions in terms of resource usage and power consumption in the monitoring nodes.

9. Conclusions

This work presented a novel approach for on-line SLA assessment, which significantly reduces both the need for measuring QoS metrics as well as the interactions between the ingress and egress nodes in the network. We accomplished this goal by: (i) proposing a novel SLA assessment mechanism based on the acquisition of Inter-Packet Arrival Times (IPATs) distributions; (ii) using effective distribution comparison algorithms; and (iii) a robust training methodology offering a competitive and lightweight solution for detecting SLA violations.

During the design and testing phases of our SLA assessment method we also proposed a variant of the *Hausdorff* Distance algorithm, i.e., *Simplified Hausdorff* Distance, specially crafted for distance computation among time distributions. Our variant is able to effectively detect traffic compliance with the SLA, offering good accuracy with very low resource requirements.

We have validated our methodology both in controlled and European-wide testbeds, using synthetic as well as real traffic. The experimental results show that we can considerably reduce the required resources without significantly affecting the accuracy of the detection system. Though not formally proved, we have also shown the strong relationship that exists between the IPAT distributions and the performance of the network.

Our future research includes developing a dynamic algorithm to fine tune the system parameters depending on the network conditions and the traffic profile, since we believe that it can further improve the accuracy in the detection of SLA violations, while reducing the consumption of network resources. We also noticed that each presented distance algorithm performs differently depending on the traffic conditions. Hence, we plan to explore ways of dynamically change the distance algorithm used to further improve the system behavior. Moreover, in our opinion, the computational cost of our solution can be further reduced by considering Inter-Packet Emission Times and their relationship with IPAT, this would allow to further reduce the metric computation in the end nodes. Finally, in this work we used a Reference Distribution Set (RDS) that referred to distributions fulfilling the SLA, enhancing the RDS to consider situations with network disruption could reduce the generation of control traffic when the network is congested, even if this opens new challenges such as the apparition of false positives, which we avoid all together in the approach presented in this paper.

Acknowledgments

The authors thank Jordi Domingo-Pascual for his support in the initial phase of this work. We also thank the partners of the European FP6 EuQoS project for letting us use their testbeds in order to perform part of the tests presented in this paper.

References

[1] Tanja Zseby. Deployment of sampling methods for SLA validation with non-intrusive measurements, in: Proceedings of the Passive and Active Measurement Workshop (PAM), Colorado, USA, March 2002, pp. 25–26.

[2] Tanja Zseby. Comparison of sampling methods for non-intrusive SLA validation, in: Proceedings of the Second Workshop on End-to-End Monitoring Techniques and Services (E2EMon), October 2004.

[3] Joel Sommers, Paul Barford, Nick G. Duffeld, Amos Ron. Accurate and efficient SLA compliance monitoring, in: Proceedings of ACM SIGCOMM, Kyoto, Japan, August 2007, pp. 109–120.

[4] René Serral-Gracià, Pere Barlet-Ros, Jordi Domingo-Pascual. Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks, in: Proceedings of the Fourth International Telecommunication Networking Workshop on QoS in Multiservice IP Networks, Venice, Italy, February 2008, pp. 142–147.

[5] René Serral-Gracià, Albert Cabellos-Aparicio, Jordi Domingo-Pascual, Network performance assessment using adaptive traffic sampling, in: IFIP Networking, LNCS, vol. 4982, Singapore, May 2008, pp. 252–263.

[6] Guy Almes, Sunil Kalidindi, Matthew Zekauskas. A One-way Delay Metric for IPPM. RFC 2679, September 1999.

[7] Carlo Demichelis, Philip Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393, November 2002.

[8] Jacob Strauss, Dina Katabi, Frans Kaashoek, A measurement study of available bandwidth estimation tools, in: Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement, 2003, pp. 39–44.

[9] Guy Almes, Sunil Kalidindi, Matthew Zekauskas. A One-way Packet Loss Metric for IPPM. RFC 2680, September 1999.

[10] Joel Sommers, Paul Barford, Nick G. Duffeld, Amos Ron, Improving accuracy in end-to-end packet loss measurement, SIGCOMM Computer Communication Review 35 (4) (2005) 157–168.

[11] Yann Labit, Philippe Owezarski, Nicolas Larrieu. Evaluation of active measurement tools for bandwidth estimation in real environment, in: Proceedings of the Third IEEE/IFIP Workshop on End-to-End

Monitoring Techniques and Services (E2EMON'05), Nice, France, May 2005, pp. 71–85.
[12] Vern Paxson, Strategies for sound internet measurement, in: IMC'04: Proceedings of the Fourth ACM SIGCOMM Conference on Internet Measurement, ACM, New York, NY, 2004, pp. 263–271.
[13] Youcef Khene, Mauro Fonseca, Nazim Agoulmine, Guy Pujoll, Internet service pricing based on user and service profiles, in: Proceedings of the 11th International Conference on Telecommunications (ICT), Fortaleza, Brazil, August 1–6, 2004, pp. 1359–1368.
[14] Daniel A. Vivanco, Anura P. Jayasumana, A measurement-based modeling approach for network-induced packet delay, in: Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN), 2007, pp. 175–182.
[15] Nischal M. Piratla, Anura P. Jayasumana, Hugh Smith, Overcoming the effects of correlation in packet delay measurements using inter-packet gaps, in: Proceedings of the 12th IEEE International Conference on Networks (ICON), vol. 1, 2004, pp. 233–238.
[16] Paul Barford, Joel Sommers, Comparing probe- and router-based methods for measuring packet loss, IEEE Internet Computing – Special Issue on Measuring the Internet 8 (5) (2004) 50–56.
[17] René Serral-Gracià, Pere Barlet-Ros, Jordi Domingo-Pascual, Distributed sampling for on-line QoS reporting, in: Proceedings of the 16th IEEE Workshop on Local and Metropolitan Area Networks, 2008, LANMAN 2008, September 2008, pp. 55–60.
[18] René Serral-Gracià, Albert Cabellos-Aparicio, Jordi Domingo-Pascual, Packet loss estimation using distributed adaptive sampling, in: IEEE on Network Operations and Management Symposium Workshops, 2008, NOMS Workshops 2008, April 2008, pp. 124–131.
[19] René Serral-Gracià, Yann Labit, Jordi Domingo-Pascual, Philippe Owezarski, Towards end-to-end SLA assessment, in: IEEE on 28th Conference on Computer Communications, INFOCOM 2009, April 2009, pp. 2581–2585.
[20] Andrew McCallum, Don Towsley, Yu Gu. Detecting anomalies in network traffic using maximum entropy estimation, in: Internet Measurement Conference (IMC), 2005, pp. 345–350.
[21] Mikhail J. Atallah, Linear time algorithm for the Hausdorff Distance between convex polygons, Information Processing Letters 17 (4) (1983) 207–209.
[22] Philippe Owezarski, Pascal Berthou, Yann Labit, David Gauchard, LaasNetExp: a generic polymorphic platform for network emulation and experiments, in: Proceedings of the Fourth International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, 24, Innsbruck (Austria), March 2008.
[23] Anthony Davidson, David Hinkley, Bootstrap methods and their application, Cambridge Series in Statistical and Probabilistic Mathematics (1997).
[24] René Serral-Gracià, Towards end-to-end SLA assessment, Ph.D. Thesis, Universitat Politècnica de Catalunya, 2009.

René Serral-Gracià received a degree, and recently his Ph.D. in Computer Science from the Technical University of Catalunya (UPC) in the Department of Computer Architecture. Currently he is assistant professor in the same University, where his teaching activities are focused in Networking and Operating Systems Administration. From 2003 his research has been focused in topics such as QoS management and provision, traffic engineering, and IP traffic analysis and characterization. More specifically he actively worked in European projects such as Laboratories Over Next Generation Networks (LONG), End-to-end Quality of Service support over heterogeneous networks (EuQoS).

Marcelo Yannuzzi is the R&D head of the Advanced Network Architectures Lab (CRAAX) at the Technical University of Catalunya. He leads several research initiatives at CRAAX, including projects with Cisco Systems and European projects in collaboration with Telefonica R&D, ADVA Optical Networking, and Juniper. In the past, he worked for the national Telco in Uruguay and teached at the Physics and the Electrical Engineering Departments of the School of Engineering, University of the Republic, Uruguay.

Yann Labit is an Assistant Professor at the University of Toulouse in the Physic Department in Toulouse. He defended his PhD in October 2002 in Automatic Control at LAAS-CNRS (Laboratory for Analysis and Architecture of Systems), in Toulouse, supported by INSA (Institut National des Sciences Appliquées) of Toulouse. His main research interests include monitoring and modeling of Internet traffic to improve QoS and to prevent from DoS/DDoS attacks. He also is interested in designing new robust AQM (Active Queue Management) based on the modern Control Theory, and Networked Control Systems (NCS).

Philippe Owezarski is a full time researcher of CNRS (the French center for scientific research), working at LAAS (Laboratory for Analysis and Architecture of Systems), in Toulouse, France. He got a PhD in computer science in 1996 from Paul Sabatier University, Toulouse III. His main interests deal with high speed and multimedia networking and more specifically on IP networks monitoring, and Quality of Service enforcement based on measurements. Philippe Owezarski has been one of the main contributors of a monitoring project in France – METROPOLIS, has been leading a French steering group on IP networks monitoring, and has been leading the French MetroSec project aiming at increasing the robustness of the Internet against DoS and DDoS attacks. Now, he is contributing to the European COST-TMA and ECODE which propose to use monitoring as the main support for enforcing QoS optimization and security mechanisms in networks.

Xavi Masip-Bruin MSc (1997), PhD (2003) in telecommunications engineering both from the Technical University of Catalonia. He is an associate professor of Computer Science and Communications at the Technical University of Catalunya. Since 1996, he has participated in many national and European research projects. In 2006, he co-founded the Advanced Networks Architectures Lab (CRAAX) being currently the CRAAX leader.