An Efficient Method for Service Level Agreement Assessment

René Serral-Gracià, Marcelo Yannuzzi, Yann Labit, Philippe Owezarski, Xavi Masip-Bruin

An Efficient 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)*
[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*

Abstract

On-line end-to-end Service Level Agreement (SLA) monitoring is of key importance nowadays. For this purpose, recent researches have focused on measuring (when possible), or estimating (most of the times) network Quality of Service (QoS) and performance parameters. Up to now, all the proposed solutions have the drawback of requiring a huge amount of resources with low accuracy, generally leading to unscalable systems.

We observe, however, that the accurate estimation of network QoS parameters is not necessarily required for SLA assessment. What is required is an efficient and scalable method to directly detect SLA violations. To this end, this paper makes the following contributions. First, we introduce a polynomial-complexity algorithm based on Hausdorff Distance that efficiently detects SLA violations. Second, we propose a *Simplified Hausdorff Distance*, which provides better accuracy at lower computational cost. Our solution works on simple to measure time series—the Inter-Packet Arrival Times (IPATs) received in our case.

The validity of our proposal is confirmed by comparing with perfect knowledge of the QoS status as well as other existing alternatives in a real testbed.

Key words: SLA Assessment, Inter-Packet Arrival Times, Hausdorff Distance

☆A preliminary version of this work appeared in Proceedings of IEEE INFOCOM 2009 under the title "Towards an efficient Service Level Agreement assessment".

Email addresses: `rserral@ac.upc.edu` (René Serral-Gracià), `yannuzzi@ac.upc.edu` (Marcelo Yannuzzi), `ylabit@laas.fr` (Yann Labit), `owe@laas.fr` (Philippe Owezarski), `xmasip@ac.upc.edu` (Xavi Masip-Bruin)

1. Introduction

On-line end-to-end Service Level Agreement (SLA) monitoring is of key importance nowadays, since both Internet Service Providers (ISP) and customers want to know the quality of network services in real-time, and whether this quality complies with the contracted SLA. For this purpose, classical approaches for SLA assessment [1, 2, 3, 4, 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, attempts to provide accurate techniques for estimating such parameters have failed (e.g., the estimation accuracy of the ABW might present error rates greater than 50% in real environments [10]). This is mainly due to network traffic variability and different traffic anomalies, which make SLA assessment a challenging problem. In addition, SLA assessment methods based on the online estimation of end-to-end QoS parameters present other drawbacks. For instance, 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, usually including stations located in the core of the network). Furthermore, estimating the QoS metrics implies to gather distributed information not only about the traffic itself, but also about the synchronization status of the involved entities [11], so the control traffic generated is an important bottleneck of any solution using this approach [4].

Overall, SLA assessment systems based on the estimation of QoS metrics suffer from both lack of accuracy and large scalability problems. 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 efficient 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, this enables the design of more scalable SLA monitoring systems.

The rationale behind this is that classically ISP deal with SLA issues by overprovisioning the network [12]. This means that at the end, the complex instrumentation needed for the online 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 proposes the following approach:

1. Work as much as possible with a single point of analysis (the egress node), so as to minimize the queries and communication with the ingress node.
2. Detect and communicate SLA violations very efficiently in order to have a scalable system.
3. Exploit the existing correlation between the measured parameters and the quality in the network.

To achieve these goals, we propose to use the Inter Packet Arrival Time (IPAT). The IPATs can be easily computed at destination by getting the reception timestamps of the packets—indeed, the computation of the IPAT only involves a subtraction of two integers (the timestamps).

Previous works [13, 14], 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 the 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. This is done by comparing different IPAT distributions using a well-known algorithm, namely the *Hausdorff Distance*.

Given that this algorithm has a fairly high computational cost $\left(O(n^2)\right)$, we propose an improvement that we name *Simplified Hausdorff Distance*, which delivers better accuracy with lower computational cost $(O(n))$ for our application.

We validate our solution with different real scenarios, first over a controlled testbed with customizable network conditions, and later over an Europe-wide scenario with 12 different testbeds and 5 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 by reducing the resource usage down to $\sim 11\%$ (i.e. we achieve an 89% reduction in the network resource usage), 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 achieved and their conditions. Section 6 shows the evaluation results, while in Section 7 we experimentally study the impact of the system parameters on the behavior of the monitoring solution proposed. Finally, section 8 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 et al. in [15], 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 [16], 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 of 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 works mentioned above, we use a passive traffic analysis approach.

Regarding passive traffic analysis, some research has been done in our previous works [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 is centralized, and requires several collection points on the edges of the network that send packet information (timestamps, etc.) to a central entity, with the consequent use of bandwidth due to the so called control traffic. Such received information 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.

More related to this paper, in our previous work [19], we presented a similar technique as the one developed here, which was based on the utilization of the *Kullback-Leibler Divergence* [20]. That solution detected SLA violations with high accuracy, but at the cost of using many network resources. In this paper, we improve the resource usage by means of the *Hausdorff Distance* [21, 22], and our enhanced *Simplified Hausdorff Distance*. We show the better behavior of the latter, by comparing our results with the ones obtained previously with the *Kullback-Leibler Divergence*, in exactly the same network conditions.

3. Distribution Comparison

As described above, we plan to 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 metrics 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. in VoIP: change of the codec, silence in the conversation, etc.). Third, currently there is 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. In the next section, we shall detail the complete infrastructure using this information.

3.1. Hausdorff Distance

The *Hausdorff Distance* [23] 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:

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

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. Basically, we must obtain the minimum distance from *all* the elements of P towards *all* the elements of Q. Hence the cost is $O(nm)$ where n and m are the sizes of P and Q, respectively. In general, $n \approx m$, so the complexity of the Hausdorff Distance is $O(n^2)$.

3.2. The Simplified Hausdorff Distance

The *Hausdorff Distance* was devised for operating in the geometric plane, but in our scenario, we use distributions instead of polygons, which allows us to reduce the algorithm's complexity.

The *Hausdorff Distance* basically compares all the elements of a set with all the elements of another. However, when working with time distributions, there is a new abstraction that does not exist on the geometrical plane, namely, the bin size. As opposed to the *"all-against-all"* comparison of the original algorithm, with time distributions, we only need to compare similar bins (in position). This allows to reduce the computational cost of the algorithm.

Definition 2. *Let's define o as the bin offset threshold, and P, 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:*

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

With this enhancement, the *Simplified Hausdorff Distance* yields linear complexity for small values of o, i.e., $O(n)$ for $o \ll n$, which is in fact the case in our context.

As a side effect, this reduction in the number of comparisons permits our algorithm to increase its accuracy. The reason for this is that now the compared bins are more closely related (the impact of the bin size selection is evaluated in Section 7).

4. SLA Violation Detection

Computing the distance among distributions determines how different are the reference and the acquired traffic profiles 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 we 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. 1. The methodology is composed of three interrelated algorithms (which are detailed throughout this section), and it 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

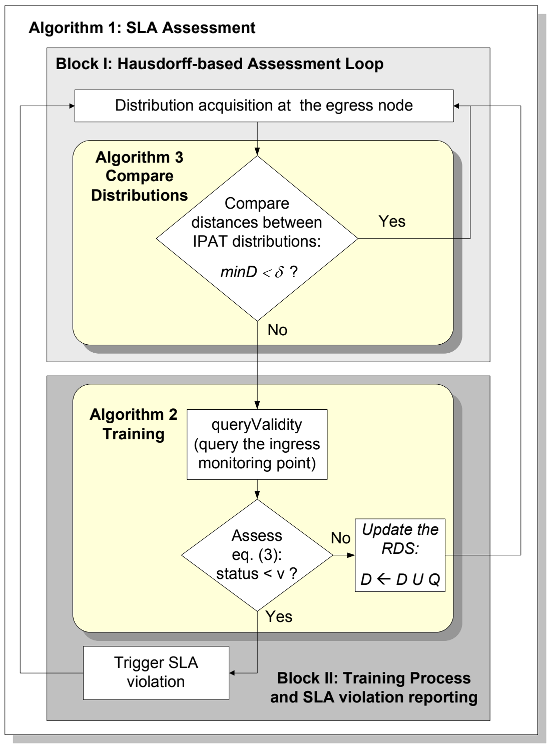

Algorithm 1: SLA Assessment
Block I: Hausdorff-based Assessment Loop
Distribution acquisition at the egress node
Algorithm 3 Compare Distributions
Compare distances between IPAT distributions: minD < δ ?
Yes
No
Algorithm 2 Training
queryValidity (query the ingress monitoring point)
Assess eq. (3): status < v ?
No
Update the RDS: D ← D U Q
Yes
Trigger SLA violation
Block II: Training Process and SLA violation reporting

Figure 1: General methodology for SLA assessment.

violations.

The strengths of this methodology are fundamentally the following:

- Once the monitoring system is appropriately trained, the assessment process will generally remain looped 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. 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, and during a configured time interval t. The empirical distribution is computed in bins of width w (referring to IPAT ranges), and therefore, a particular IPAT i falls in bin $k = \lfloor \frac{i}{w} \rfloor$ (the study of the proper t and w values is deferred to Section 7).

After the distribution acquisition in 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.
    - *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 detailing how the system learns the real quality provided by the network, and then we focus on how the system compares 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* ν.

In our model, the training process queries the source monitoring point of the flow, 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 must validate if 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 (3).

4.3. Validity Function $\mathcal{V}$

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

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

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

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

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 Alg. 2}
    else
        $status \leftarrow compareDistributions(Q, S)$ {see Alg. 3}
    end if
10: if $status < \nu$ then {Unacceptable QoS conditions}
        trigger $SLAViolation(status)$
    end if
```

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

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 $\mathcal{V} \geq \nu$ the network behavior is considered stable, where ν is the specified quality threshold of the system. The closer is ν 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 $\left\{\mathcal{D}^1, \ldots, \mathcal{D}^{|\mathcal{D}|}\right\}$ 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.

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 and resources required by the system, and for the sake of efficiency and scalability, it should be minimized. Nevertheless, this method reports reliable values of the network metrics to be mapped to the reference IPAT distribution.

Once the *real* validity is assessed, if it is below ν, the event is registered, the distribution discarded, and a *SLAViolation* 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; *ii)* it honors the fact that if a distribution has not been representative of the traffic profile for a while, it's due to changes in the network status; so this distribution is not required anymore.

Algorithm 2 Training

```
    Input: $Q,S$ {$Q$ : IPAT distribution, $S$ : Flow Source}
    Output: $status$

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

4.4. Distribution comparison

The complete pseudocode 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* is one distribution from another, or preferably, as the degree of similitude (dissimilitude) between two distributions. The higher the distance, the more different the distributions (and so does 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:*

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

where $d(p,Q)$ is the distance between the distributions p and Q, such as the ones presented in Section 3.

Algorithm 3 compareDistributions

```
Input: $Q, S$ {$Q$ :Acquired Distribution, $S$ : Flow's Source}
Output: $status$

$minD \leftarrow \infty$
for all $D \leftarrow \mathcal{D}$ do
    $d \leftarrow computeDistance(D, Q)$
    if $d < minD$ then
        $minD \leftarrow d$
        $P \leftarrow D$
    end if
end for
if $minD \geq \delta$ then
    $status \leftarrow Training(P, S)$ {Does metric computation}
else
    $updateUse(P)$ {Renew usage of the distribution to prevent expiration when $|\mathcal{D}| = \Delta$}
    $status \leftarrow getValidity(P)$ {Use value of $queryValidity$}
end if
```

Then, Q and $\mathcal{D}$ are considered similar if $D_M(\mathcal{D}, Q) \leq \delta$, 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. The *Training* procedure is then invoked with Q (see the transition from Block I to Block II in Fig. 1). As described previously, training consumes system resources, and therefore, there is a trade-off since the lower δ, the more resources (queries) will be needed. On the other hand, the higher δ, the lower amount of resources will be required, though at the cost of losing some accuracy on 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 have 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 Fast Ethernet without cross traffic. 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 leads 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 on the test.

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

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* [24] traffic profiles with all the above network conditions. Using these traffic profiles we have from predictable packet rates *(Periodic)* to unpredictable 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 twelve 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 5 countries and 4 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 $64Kbps$ 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 $1Mbps$ of UDP traffic. We call this test `UDP1`. Finally, the third kind of traffic is an average sized, high rate UDP flow with $\sim 1.4Mbps$, we call this test `UDP2`.

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 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., not detected SLA violations). It is important to notice that false positives cannot happen in this environment, since we always query network performance related to unknown IPAT distribution.

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 how many resources are required by the system; such resources are counted in terms of reduction ratio of the required bandwidth used by the control traffic. Therefore, we compare the cost of reporting information in a per packet basis, against our solution, which only demands information when there is a change in the traffic reception profile.

All the issued analysis uses the same set of parameters. In particular, we set up, as a proof of concept, the following values: distance threshold of $\delta = 3\%$, bin width of $w = 3ms$, and an acquisition time interval of $t = 175ms$. 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. For each test we collect the full trace on both end nodes.

2. We extract the network performance metrics as described in [4] from per packet information, using them as reference quality (perfect knowledge), since the process gives exact values for the metrics.
3. We identify the different SLA violation periods with the reference results acquired above.
4. We apply off-line our algorithm (by using *Kullback-Leibler*, *Hausdorff* and *Simplified Hausdorff*). Here we record: *i)* required control traffic due to *Training*. *ii)* estimated SLA violation periods.
5. Finally, we evaluate the matching between the SLA violations and the ones obtained in Step 3.

We apply our system off-line with the goal of comparing the results obtained by the perfect knowledge and our system. Nevertheless, in a real deployment our technique will be applied on-line.

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.

6.2.1. Synthetic traffic with controlled network

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

We analyze in Table 1 the *Accuracy* and the *Resource* utilization for the different generated traffic. The accuracy is computed for the overall test duration, counting the ratio of detected SLA violations over the total. While the required resources are computed by the ratio of the actual number of queries over the maximum possible queries per test. 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

(a) Periodic

	Accuracy				*Resources*			
	Good	Mild	Medium	Severe	Good	Mild	Medium	Severe
KL	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
S. Haus.	1.000	1.000	0.868	1.000	0.001	0.130	0.267	0.394

(b) Poisson

	Accuracy				*Resources*			
	Good	Mild	Medium	Severe	Good	Mild	Medium	Severe
KL	1.000	0.250	0.940	0.893	0.562	0.572	0.657	0.671
Hausdorff	1.000	0.750	0.067	0.225	0.122	0.143	0.132	0.190
S. Haus.	1.000	1.000	0.994	0.999	0.464	0.601	0.699	0.721

(c) Synthetic Real Traffic

	Accuracy				*Resources*			
	Good	Mild	Medium	Severe	Good	Mild	Medium	Severe
KL	1.000	0.667	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
S. Haus.	1.000	0.667	1.000	1.000	0.002	0.002	0.397	0.397

Table 1: Accuracy and Resources for $\delta = 0.03$

conditions, where our contribution is more useful, we can detect with very good accuracy the SLA disruption periods. On the other hand, in the fuzzy case when there are few SLA violations, the accuracy of the system drops sensibly for some algorithms. The cause of this is the statistical resolution achieved when there are few SLA violations: in this case missing a single violation period is statistically significant. Moreover, in a real deployment, such SLA violations are of no practical interest since they represent very short, sporadic, periods of light congestion, with no humanly noticeable impact on the final network behavior.

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.

The second consideration is the 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 used algorithm. 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 bad network conditions are encountered. Hence it forces the system to query for exact metrics, the minimum number of queries is bounded by the amount of SLA violations. In our experimental case this is 0.397 as shown in the 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 implies not detecting some SLA violations.

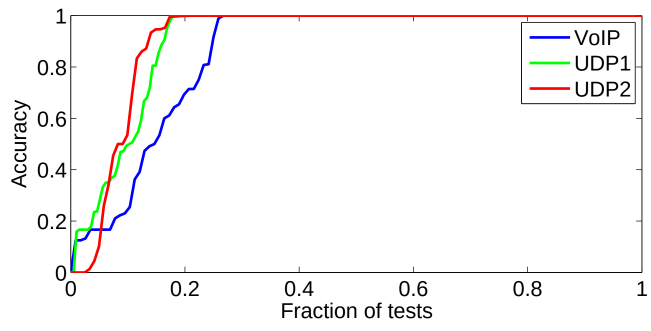
6.2.2. Synthetic traffic over the European Research network

In this testbed we plan to show the proper accuracy of our proposal in a real network with random quality, unexpected events and unknown cross traffic with different multi-hop paths.

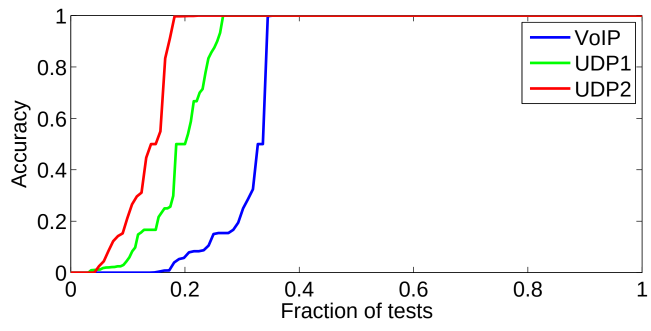
In Figure 2 we show the different accuracy results for each algorithm and traffic profile. The X-axis of the figure represents the test number (normalized to 1) and the Y-axis the accuracy. The figure considers all the tests, including the ones without SLA violations. The results show that for both *Kullback-Leibler* and *Simplified Hausdorff* we get perfect accuracy for more than 70% of the tests.

We complement the figure with Table 2, which summarizes the results of our experiments. We show the aggregated total amount of periods with violations, together with the amount of such periods our algorithm could 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.

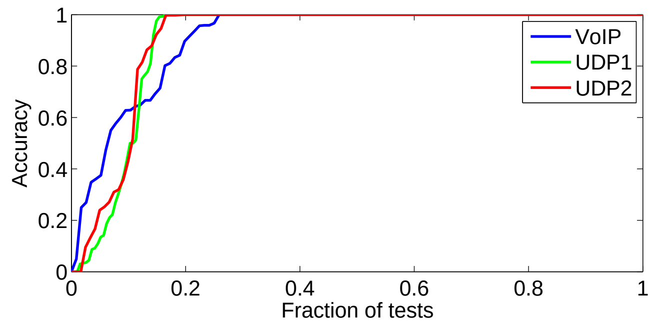
It is important to notice that most of the failures in the SLA estimation are due to isolated violations, which are very close to the SLA agreement boundary with no



(a) Kullback-Leibler



(b) Hausdorff



(c) Simplified Hausdorff

Figure 2: Accuracy for Synthetic traffic over the European Network

(a) Kullback-Leibler

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	6096	0.845	0.198
UDP1	62264	60108	0.965	0.551
UDP2	24863	22265	0.896	0.338

(b) Hausdorff

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	3163	0.438	0.024
UDP1	62264	58003	0.932	0.237
UDP2	24863	21384	0.860	0.221

(c) Simplified Hausdorff

	Violations	*Detected*	*Accuracy*	*Resources*
VoIP	7216	4765	0.660	0.044
UDP1	62264	59265	0.952	0.246
UDP2	24863	22345	0.899	0.232

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

practical interest.

In this set of tests, the best performing algorithm is *Kullback-Leibler*, but at the expenses of using more network resources than the other alternatives. As in the previous case, *Hausdorff* falls behind in terms of accuracy, which added to its higher computational complexity, makes it the worst presented algorithm.

In terms of resources, the average resource usage of the whole system is below 25%. It is lower when considering `VoIP` traffic (i.e., around 4%), while the minimum amount of resources required is $\sim 5.8 \cdot 10^{-4}$, and the maximum is 1 (meaning no reduction in control traffic compared to the per packet reporting is achieved). Further investigating the tests causing more resource usage, we found 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

(a) Kullback-Leibler

	Accuracy				*Resources*			
	Good	*Mild*	*Medium*	*Severe*	*Good*	*Mild*	*Medium*	*Severe*
Audio	1.000	1.000	0.999	0.997	0.962	0.966	0.947	0.728
Video	1.000	0.450	0.621	0.874	0.061	0.074	0.085	0.569

(b) Simplified Hausdorff

	Accuracy				*Resources*			
	Good	*Mild*	*Medium*	*Severe*	*Good*	*Mild*	*Medium*	*Severe*
Audio	1.000	0.999	0.979	0.975	0.114	0.114	0.144	0.397
Video	1.000	0.450	0.677	0.848	0.032	0.035	0.050	0.407

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

expected.

6.2.3. *Real traffic over a controlled testbed*

In this last set of tests, it is intended to observe the performance of our algorithms under real traffic with controlled network behavior. 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 for *Kullback-Leibler* and *Simplified Hausdorff*, omitting *Hausdorff* given its bad accuracy.

The accuracy in detecting the studied SLA violations is higher than 99% for the audio flows, dropping sensibly in the case of video flows. The causes of such difference in accuracy are unknown yet and subject to further study. Nevertheless, in the case of *Severe* network conditions the accuracy is still higher than $\sim 85\%$ with both algorithms.

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 this two cases. Again, using *Kullback-Leibler*, despite of having slightly better accuracy, requires $\sim 20\%$ more resources than *Simplified Hausdorff*.

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 of the system, namely, the time interval t, the bin size w and the threshold δ. The full analytical study of the impact of such parameters on the system performance can be found in [25].

7.1. The acquisition interval t

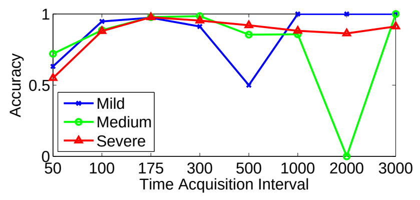
As we discussed previously, increasing the acquisition time interval gives our system more statistical soundness, due to the presence of more samples in the acquired distribution. But at the same time it gives less responsiveness to our system.

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 milliseconds. Then we estimated the SLA violations by using our three algorithms.

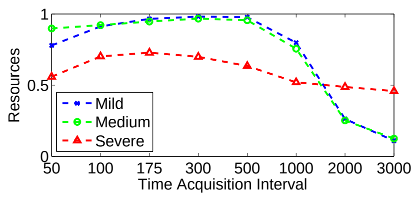
Figure 3 summarizes the obtained results for all these values. Among the results we omit the *Good* quality tests since its accuracy is always 1 and the resource consumption is in all cases lower than the other levels of disruption.

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. The main reason for this is the increasing amount of information that improves the statistical significance.

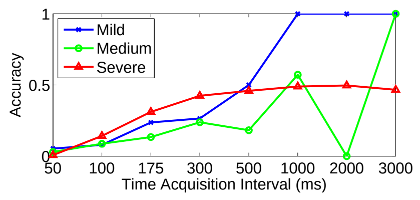
As it can be noted there is a considerable drop in accuracy for audio traffic in the case of *Simplified Hausdorff* and in *Kullback-Leibler* for *Mild* traffic conditions (in fact it drops to 50%, in the specific case of $500ms$). The cause for this is that when



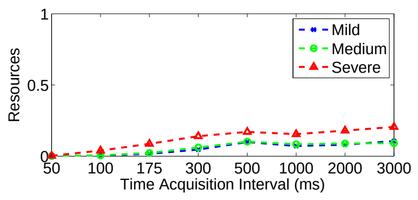
(a) Kullback-Leibler Accuracy



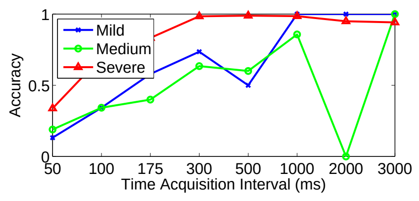
(b) Kullback-Leibler Resources



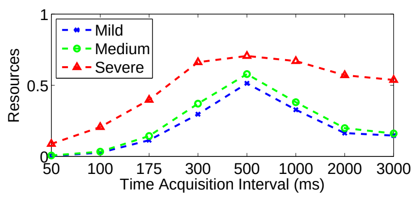
(c) Hausdorff Accuracy



(d) Hausdorff Resources



(e) Simpl. Hausdorff Accuracy



(f) Simpl. Hausdorff Resources

Figure 3: Time Acquisition interval effect over VLC Audio Traffic

aggregating more packets into the bin, with few packet losses, the aggregated quality for the period gets *"better"* in the sense that the packet loss ratio is reduced. Therefore the number of total bins with violations drops (in our case to 2 for the whole test), and the three algorithms have more difficulty to detect them (in our case only one is correctly detected). Also as expected, this behavior is repeated for the case of *Medium* traffic disruptions for higher bin sizes.

In the case of video flows, the results differ considerably. This is because this traffic uses higher packet rates, which at the end, implies that more packets get dropped. So, increasing the time interval helps for a better estimation.

As we already detected in the evaluation section both *Simplified Hausdorff* and *Kullback-Leibler* have better accuracy than *Hausdorff*. By comparing both algorithms, we noticed that, in general, *Kullback-Leibler* has better accuracy, but requires too much resources in terms of bandwidth. In any case for high values of t, *Simplified Hausdorff*, while using a fairly reasonable amount of resources, accomplishes a similar degree of accuracy.

Summarizing the findings about the accuracy, we can see that in general having large bin sizes helps the estimation. But this could be misleading, because for large timescales, small disruptions get undetected, while being relevant for shorter timescales. This is not really a fault of our system but the actual reduction in the packet loss ratio per time interval (therefore that traffic has a validity higher than ν).

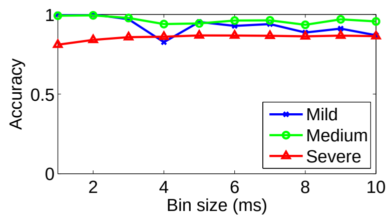
7.2. Impact of the bin size w

Changing the bin size 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 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.

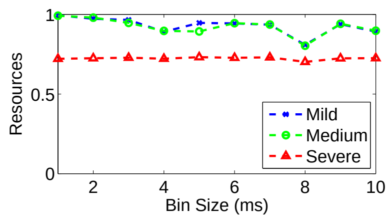
The analysis is performed by using w values from 1 to 10 milliseconds of IPAT. In the rest of the section we study these different values with the three distance algorithms, to see the effect on the accuracy and used resources.

In Figure 4 we show the results with values of $t = 175ms$ and $\delta = 3\%$, for the different w. The results highlight only the accuracy and resources of the audio flows.

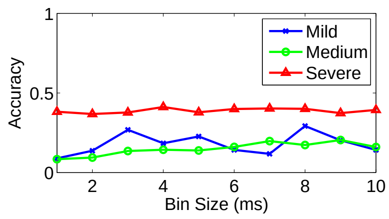
Similar results are obtained with the video stream.



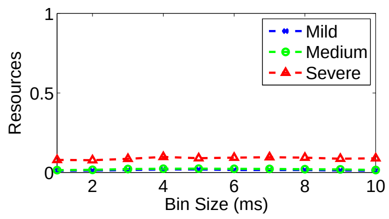
(a) Kullback-Leibler Accuracy



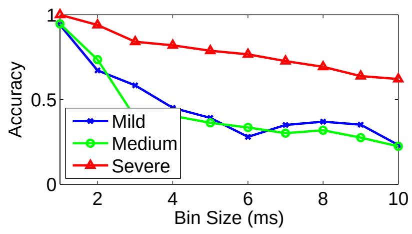
(b) Kullback-Leibler Resources



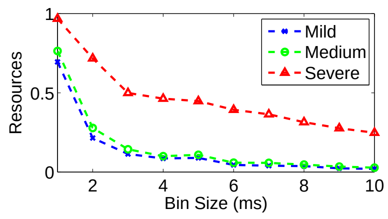
(c) Hausdorff Accuracy



(d) Hausdorff Resources



(e) Simpl. Hausdorff Accuracy



(f) Simpl. Hausdorff Resources

Figure 4: Bin Size effect over VLC Audio Traffic

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 size 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 sizes but its accuracy drops very fast as the bin size gets bigger. This highlights the point that increasing the bin size helps reducing the resolution. That is, the system considers similar groups of IPAT which, in reality, are very different. Moreover here we can see in more details the lack of accuracy provided by the *"all-against-all"* mechanism used by *Hausdorff*.

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

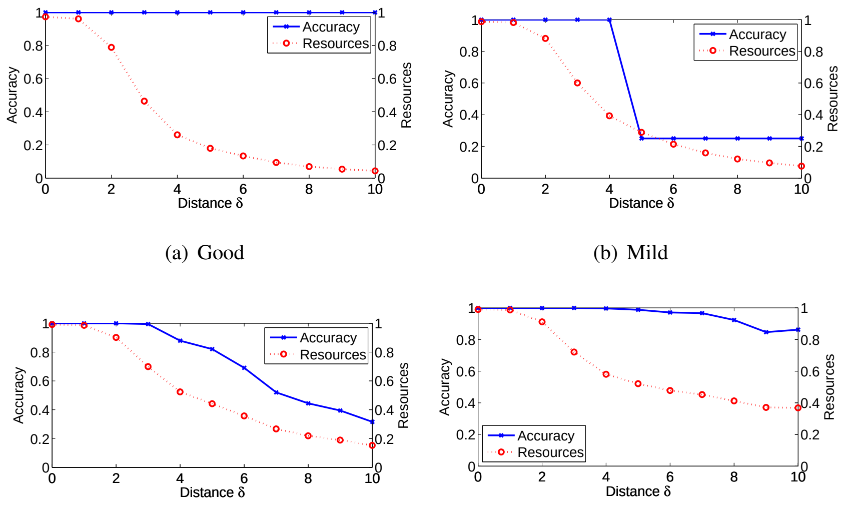
Finally in the case of *Simplified Hausdorff* the obtained accuracy is reduced as the w increases.

One interesting point to notice is that the bin size impacts more strongly *Simplified Hausdorff* than *Kullback-Leibler* algorithms. The reason is that geometric distances are more deterministic (they are a real metric indeed) than the relative entropy used by *Kullback-Leibler*, which, being dimensionless, do not have this spatiality 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 querying is triggered when $D(\mathcal{D},Q) \geq \delta$ (Step 7 of Algorithm 2). Hence δ and the traffic itself determine the amount of queries of the system. Figure 5 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 solid line, and the Resources needed on the right Y-Axis with 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



(a) Good (b) Mild

(c) Medium (d) Severe

Figure 5: Distance Effect Poisson traffic. Accuracy and Resources are on the Y-Axis and each considered distance (in %) is on the X-Axis

the estimation is always correct. Again the important trade-off is on the fuzzy cases for which not much 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. Conclusions

We have presented a novel approach for on-line SLA assessment, where differently of previous research, our work separates and reduces the performance metric computation and the interaction between the edge nodes of the network. This is accomplished by: *i)* a smart algorithm for gathering the distribution of Inter-Packet Arrival Time (IPAT); *ii)* distance algorithms to compare the distributions; and *iii)* a robust Training methodology that delivers a very competitive solution regarding SLA violation detection.

As an additional contribution, we improved *Hausdorff Distance*, by using the knowledge about the data we are dealing with. With this improved version we can efficiently infer the network quality with very good accuracy and a very low amount of resources.

We validated our methodology with a set of different tests, which involved a controlled and European-wide testbeds, using synthetic and real traffic. The experimental results show that we can reduce the required resources considerably, with a low effect on the final accuracy of the system. Though not proved formally, we also validate the strong relationship that exists between IPAT distributions and network performances.

As lines left for further research, an interesting upgrade of the system would be to infer the real metrics of the network by comparing ingress and egress packet arrival times (Inter Packet Emission Time with Inter Packet Arrival Time).

Acknowledgments

The authors would like to thank Jordi Domingo-Pascual for the support in the initial contribution of the paper.

We also thank the partners of the EuQoS project for permitting us to use their testbeds in order to perform part of the tests presented in this work.

References

[1] Tanja Zseby. ”Deployment of Sampling Methods for SLA Validation with Non-Intrusive Measurements”. *Proceedings of Passive and Active Measurement Workshop (PAM)*, pages 25–26, Colorado, USA, March 2002.

[2] Tanja Zseby. ”Comparison of Sampling Methods for Non-Intrusive SLA Validation”. In *2nd Workshop on End-to-End Monitoring Techniques and Services (E2EMon)*, October 2004.

[3] Joel Sommers, Paul Barford, Nick G. Duffeld, and Amos Ron. ”Accurate and Efficient SLA Compliance Monitoring”. In *Proceedings of ACM SIGCOMM*, pages 109–120, Kyoto, Japan, August 2007.

[4] René Serral-Gracià, Pere Barlet-Ros, and Jordi Domingo-Pascual. “Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks”. In *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pages 142–147, Venice, Italy, February 2008.

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

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

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

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

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

[10] Yann Labit, Philippe Owezarski, and Nicolas Larrieu. ”Evaluation of active measurement tools for bandwidth estimation in real environment”. *3rd IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON’05)*, pages 71–85, Nice, France, May 2005.

[11] Vern Paxson. ”Strategies for Sound Internet Measurement”. In *IMC ’04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 263–271, New York, NY, USA, 2004. ACM.

[12] Youcef Khene, Mauro Fonseca, Nazim Agoulmine, and Guy Pujoll. ”Internet service pricing based on user and service profiles”. In *11th International Conference on Telecommunications (ICT)*, pages 1359–1368, Fortaleza, Brazil, August 1-6 2004.

[13] Daniel A. Vivanco and Anura P. Jayasumana. ”A Measurement-Based Modeling Approach for Network-Induced Packet Delay”. *Local Computer Networks (LCN). 32nd IEEE Conference on*, pages 175–182, 2007.

[14] Nischal M. Piratla, Anura P. Jayasumana, and Hugh Smith. ”Overcoming the effects of correlation in packet delay measurements using inter-packet gaps”. *Proceedings. 12th IEEE International Conference on Networks (ICON)*, 1:233–238, 2004.

[15] Paul Barford and Joel Sommers. ”Comparing Probe -and Router- based Methods for Measuring Packet Loss”. *IEEE Internet Computing - Special Issue on Measuring the Internet*, 8 Issue 5:50–56, 2004.

[16] Joel Sommers, Paul Barford, Nick G. Duffeld, and Amos Ron. ”Improving Accuracy in End-to-end Packet Loss Measurement”. volume 35, pages 157–168, New York, NY, USA, 2005. ACM.

[17] René Serral-Gracià, Pere Barlet-Ros, and Jordi Domingo-Pascual. ”Distributed Sampling for On-line QoS Reporting”. pages 55–60, Sept. 2008.

[18] René Serral-Gracià, Albert Cabellos-Aparicio, and Jordi Domingo-Pascual. ”Packet loss estimation using distributed adaptive sampling”. pages 124–131, April 2008.

[19] René Serral-Gracià, Yann Labit, Jordi Domingo-Pascual, and Philippe Owezarski. ”Towards End-to-End SLA Assessment”. In *INFOCOM 2009. The 28th Conference on Computer Communications. IEEE*, pages 2581–2585, Apr 2009.

[20] Andrew McCallum, Don Towsley, and Yu Gu. ”Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation”. In *Internet Measurement Conference (IMC)*, pages 345–350, 2005.

[21] Oliver Jesorsky, Klaus J. Kirchberg, and Robert Frischholz. ”Robust face detection using the hausdorff distance”. *Proceedings of Audio and Video based Person Authentication*, pages 90–95, 2001.

[22] Yann Labit and J. Mazel. HIDDeN: Hausdorff distance based Intrusion Detection approach DEdicated to Networks. In *The Third International Conference on Internet Monitoring and Protection (ICIMP 2008), Bucharest (Romania)*, pages 11–16, 29 June - 5 July 2008.

[23] Mikhail J. Atallah. ”Linear time algorithm for the Hausdorff distance between convex polygons”. *Information processing letters*, 17(4):207–209, 1983.

[24] Philippe Owezarski, Pascal Berthou, Yann Labit, and David Gauchard. ”LaasNetExp: a generic polymorphic platform for network emulation and experiments”. 24, Innsbruck (Austria), March 2008.

[25] René Serral-Gracià. *”Towards End-to-End SLA Assessment”*. PhD thesis, Universitat Politècnica de Catalunya, 2009.