Efficient network performance assessment using Inter-Packet Arrival Times

René Serral-Gracià, Eva Marin-Tordera, Marcelo Yannuzzi, Xavi Masip-Bruin, Sergi Sánchez-López,
* Advanced Network Architectures Lab (CRAAX), Technical University of Catalunya (UPC), Spain
Email:{rserral, eva, yannuzzi, xmasip, sergio}@ac.upc.edu

***Abstract*—Network performance measurements have been broadly used in order to debug and to assess the reliability of the network. In general performance measurement is a resource consuming process, which in practice derives in hardly scalable systems. However, in order to control and monitor network resource usage and to assess the quality of multimedia traffic, it is broadly accepted that efficient and scalable network performance evaluation infrastructures must be present in nowadays networks. Nevertheless, most of current research is focused on the assessment of the different performance metrics in order to determine if the quality of the delivered service is correct.**

Differently, the main contribution presented in this paper, relies on the fact that the accurate estimation of the network metrics is generally not necessary, given that it is much more efficient to directly detect service disruptions. To this end, we propose an algorithm to detect anomalies on the service level delivered to the users. Our solution is based on the distance measurement between acquired reference distributions on the Inter-Packet Arrival Times. This simple to measure time-series permit to detect very efficiently QoS disruptions, making of the solution a very good candidate to be used on Service Level Agreement assessment systems, as we prove in our experimental set up, where we evaluate the performance and accuracy of the proposal in a real scenario.

I. Introduction

The massive deployment of multimedia streaming and real-time services on the Internet has forced Network Operators (NO) to search for measurement techniques in order to assess the performance of the networks, and in particular mechanisms to assess whether the network is complying with the Service Level Agreement (SLA) issued with the customers. Hence. SLA assessment has been a deeply studied topic in the literature. The classical approach to address the problem has been through complex real-time network performance metric computation techniques, by using Passive Traffic Analysis, e.g., [1], or by means of Active Traffic Generation, e.g., [2], [3]. Both approaches aim at the accurate estimation of Available Bandwidth (BW), One Way Delay (OWD), Inter Packet Delay Variation (IPDV), Packet Loss Ratio (PLR), etc.

The main concern when performing metric estimation through Passive Traffic analysis is that real-time performance metric assessment incurs in severe scalability issues, given the computational intensive process required in high speed backbone networks. Moreover, in order to compute metrics such as OWD it is mandatory to setup several collection points in vantage locations of the network, which imposes a severe constraint in terms of privacy and willingness of the NO to deploy smart boxes on their network. Another constraint comes from the fact that performing per packet passive traffic analysis requires a fairly high amount of control traffic exchanged by the different collection points the in order to synchronize the information gathered and to allow the metric computation.

This work was partially funded by Spanish Ministry of Science and Innovation under contract TEC2009-07041, and the Catalan Government under contract 2009 SGR1508.

Regarding Active Traffic Generation, they have a two-fold issue, the first one is that they generate traffic that changes the conditions of the network it is measuring, and secondly, the network dynamics can change dramatically, specially in access networks, which questions the accuracy of the approach.

Hence, we argue that improving metric computation, even if it might lead to interesting findings, it is a fundamentally flawed approach, since the reason for the performance metrics is to assess whether the network is providing the desired quality or not. To address this issue, in our previous work [4] we proposed a solution where, by comparing Inter-Packet Arrival Time (IPAT) distributions, we were able to efficiently detect network disruptions. The nice characteristics of IPATs is that they can be easily computed at the egress node of the network by subtracting timestamps. Our solution exploits the existing correlation between changes on IPAT distribution with network congestion [5]. It is the goal of this paper, to present an extension of the above approach in order to improve even further the accuracy and to reduce the resource requirements of the network disruption detection mechanism. To this end, we use the well known Bhattacharyya Distance, as trigger for network disruption detection.

We validate our solution using a real European-wide testbed with more than 500 tests over 5 different access technologies. Our results highlight the accurate detection of the different network disruptions found in our traces, while reducing the resource usage down to a $\sim 25\%$, with an accuracy of $\geq 85\%$, compared to the perfect knowledge of the network disruptions.

The rest of the paper is structured as follows. Next section details some related work. After this, we present the main contribution and methodology used to perform the on-line network disruption detection. After this the paper describes the different testbeds and the evaluation results. Finally in Section V we conclude and explain the open issues for future work.

II. Related Work

Most of the work in SLA Assessment present in the literature is performed by means of Active Traffic generation. The most recent work in this area can be found at [3]; where the authors continue the work started at [2], [6], by presenting an active probe tool which estimates the network performance metrics (OWD, PLR, and IPDV) by using a Multi-objective and multipurpose discrete time traffic generator in a very efficient and effective way.

In the case of SLA Assessment using Passive Traffic Analysis, the first approach to propose accurate metric estimation was performed by Zseby et al. in [7], who propose a mechanism to autonomously identify packets on different collection points. The issue of this packet identification is that when gathering the per packet information on several collection nodes it requires considerable amounts of bandwidth used by this *control traffic*. To alleviate this burden in [1] we presented a dynamic adaptive sampling technique which, by using IPDV information, the system was able to determine the sampling rate dynamically.

Also in the area of passive traffic analysis, there is active research in the area in the PerfSONAR [8] project where the aim is to provide a full-fledged and scalable system for network performance assessment. Even though, not much studies of the platform have been published yet, some preliminary results can be found in [9].

Differently from the above approaches, in order to avoid the overhead associated with the metric computation, in our previous work we proposed a solution aimed at avoiding such overhead by comparing the Relative Entropy of the received Inter-Packet Arrival Time distribution with reference distributions; which allowed to assess potential changes on the congestion in the network. Such approach used the well-known Kullback-Leibler Divergence (KLD) in order to compare the relative entropy of both distributions. This together with a learning algorithm, allowed the system to detect traffic disruptions efficiently. Despite of this, KLD has the drawback of requiring a non-negligible amount of control traffic, used to select the reference distributions, and thus assessing the network quality. To overcome this limitation in this paper we extend our previous work by removing KLD and replacing it with the Bhattacharyya Distance (BD) which, as we will prove later, delivers similar results in terms of accuracy but greatly reducing the amount of required control traffic.

III. Network disruption detection

In order to reduce the computation of the network performance metrics, we propose a mechanism to detect network disruptions based on the evaluation of the differences in the IPAT at the egress nodes of the network under analysis. To this end, we periodically compare the acquired IPAT with a set of reference IPAT distributions. We obtain these reference distributions through an on-line training process which learns about new IPAT distributions in the network, and establishes a link between each distribution with its associated quality. In this section, we describe the general methodology, together with the specific learning strategy, and the distance algorithm used to perform the network disruption detection.

A. General description

To detect potential network disruptions we first collect the IPATs during a predefined time period at the egress node. We then, compare their distribution with a reference distribution set, where it is decided whether the obtained distribution is close to the reference. In the case that the difference is below a threshold we assume similar IPAT, and therefore similar network behavior in both cases. Otherwise, we query the ingress node to acquire detailed packet information, from which we compute the real performance metrics. Finally, we can assess the status of the network and report any encountered network disruptions.

The collection of IPAT distributions is performed in a per flow basis, during a predefined time interval t. The empirical distribution is computed in bins of width w (referring to IPAT ranges). Therefore, a particular IPAT i falls in bin $k = \lfloor \frac{i}{w} \rfloor$.

After the distribution acquisition, to cope with the IPAT variability, we must learn and validate the traffic profile. This implies to map the traffic to the actual metrics and to update the set of valid distributions.

B. Learning Strategy

The first critical part of the proposed algorithm is the *Learning Strategy*, which determines the complexity to issue the network disruption detection on the egress node. Before entering with the full description of the *Learning Strategy*, we need to define a few concepts.

Definition 1: A *Valid IPAT Distribution* $\mathcal{V}$ is such a distribution where a function of the real metrics (OWD, IPDV and PLR) fall above a specified threshold ν, i.e., the network is providing a performance within the established boundaries. We discuss about how $\mathcal{V}$ is computed in Section III-D.

Definition 2: Reference Distribution Set (RDS) $\mathcal{D}$, as a set of strictly different valid distributions. Where $|\mathcal{D}|$ is the cardinality of the set, $\mathcal{D}^1$ is the first element and $\mathcal{D}^{|\mathcal{D}|}$ the last one. The memory used by the RDS is upper bounded by the maximum amount of slots designated to store the reference distributions, in this work we leave this number of slots Δ as a parameter of the system.

To ease its implementation, the *Learning Strategy* only stores in the RDS distributions with proper network behavior, which implies that any successful comparison represents no disruption on the service. To correctly assess the quality of a given distribution, e.g., when the comparison exceeds a threshold, we must map the distribution to its real quality. To this end, we use the technique presented in [10], where the source measurement point (i.e., the ingress router) sends per packet information, such as transmission timestamps, which are matched on the destination measurement point (i.e., the egress router), computing the relevant performance metrics. This technique requires control traffic from source to destination to compute the metrics as discussed before. Such control traffic determines the amount of bandwidth (resources) required by the system, and for the sake of efficiency and scalability it should be minimised. Nevertheless, this method reports exact values of the network metrics that we can use to map to the IPAT distribution.

Once the *real* validity is assessed, it is registered to the RDS only in the case its validity is above ν, while in the case that the distribution is "not valid" the system will trigger a *Network Disruption* event. It is worth noticing that the *Learning* process is only invoked when the distribution comparison is not accurate —i.e., when network conditions are unknown.

C. Distribution comparison

The second critical aspect of the proposed solution is the comparison of the reference distribution with the acquired. Since this process is periodically invoked with new distributions. In our first approach [4] we used the *Kullback-Leibler Divergence* (KLD) which compares the relative entropy of both distributions, here we improve this approach by using the *Bhattacharyya Distance* (BD).

1) Bhattacharyya Distance: There are many ways to compare distributions, in our system we aim at comparing the overlap of reference IPAT distributions with ongoing ones, in order to determine if current network conditions represent any form of service disruption.

BD is a well-known mathematical method which permits to compute overlapping in the samples of a distribution. This method uses Exp. 1.

$$D_B(p,q) = -\ln\left(\sum_{i=1}^{n} p(x)q(x)\right) \tag{1}$$

where p and q are the two discrete distributions of size n bins to compare.

The above expression gives the degree of overlapping between both distributions. In our algorithm we consider p as the reference, and q, the distribution to be tested.

In this paper we use this distance as a measure of the difference on the IPAT distribution, which indicates potential changes in the network conditions.

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

Definition 3: Degree of Matching D_M, between a RDS $\mathcal{D}$ and another distribution Q is defined as:

$$D_M(\mathcal{D},Q) = \quad \min\{d(p,Q)\} \quad p = \mathcal{D}^1 \ldots \mathcal{D}^{|\mathcal{D}|} \tag{2}$$

where $d(p,Q)$ is the comparison algorithm between distributions p and Q as discussed before.

Then Q and $\mathcal{D}$ are considered similar if $D_M(\mathcal{D},Q) \leq \delta$, where δ is our distance threshold. The most important aspect of these comparisons is that different distributions do not necessarily imply different performance qualities, since the traffic profile of the applications can change unexpectedly, or we can experience different levels of congestion, all of them complying with the objective network quality. As a consequence, if we find a distribution Q which is not similar to any distribution in $\mathcal{D}$ we must learn the quality of this new distribution. Therefore, we invoke the *Learning Strategy* previously defined, which will query the ingress node in order to assess the specific network metrics for that time period. Given that the learning inherently consumes resources, in terms of control traffic, we have to consider a trade-off. Hence, having low values for δ means higher number of misses in the distance computation, forcing the system to query more often, but assuring higher accuracy in the network disruption detection than in the case of having bigger δ which makes our system more permissive to differences on the distribution.

As it can be noted, in some cases it could happen that $D_M(\mathcal{D},Q) \leq \delta$ while having a network disruption. These false negatives are possible, and they are the only source of inaccuracy of our system. On the other hand, this effect is not present in the case of false positives, i.e., detecting disruptions where they don't exist, given that our system always queries for the real metrics if $D_M(\mathcal{D},Q) > \delta$. As we experimentally prove in the evaluations section, even with some degree of inaccuracy our system is able to efficiently detect most of the network disruptions.

D. Distribution Validity $\mathcal{V}$

The final objective of our system is to decide whether the network is having performance issues or not. In particular, if it is complying with the contract with customers, that is, the validity of the traffic is above the threshold ν described previously.

As we already discussed, $\mathcal{V}$ is defined in the range $[0,1]$ indicating the quality of service experienced for the flow with respect to the agreement with the customers, we compute its value by using the usual metrics (OWD, IPDV and PLR).

Another point to consider is that, depending on the type of traffic, the QoS constraints might considerably differ. Hence, we define $\omega_O, \omega_I, \omega_P$ as weights specified for each particular metric on each monitored flow, where $\omega_O + \omega_I + \omega_P = 1$.

Expression 3 computes $\mathcal{V}$, which is the degree of validity of the time interval.

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

Where $Q^*(x)$ determines the quality degradation function. We assume that this degradation function is exponential and defined as:

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

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

Then $\mathcal{V} \geq \nu$ the network behavior is considered stable. The closer is ν to 1, the stricter our system will be to network violations.

IV. Evaluation

We validate the improvement achieved by BD in respect to the accuracy and resource consumption that we obtained with our previous KLD. In particular, we focus on the system's accuracy, that is, in the network disruption detection rate, measured in terms of false negatives (i.e., not detected disruptions). We compare BD and KLD, using as reference the case of perfect knowledge about the network disruptions, we complement the study by analyzing the amount of resources required by the system; such resources are counted in terms of reduction ratio of the required bandwidth used by the control traffic. More specifically, we compare the cost of reporting per packet information as defined in [10] with BD and KLD, which only demand information when there is a change in the traffic reception profile.

During all the analysis we use the same parameters across the tests for the estimation. In particular, we set up, as a proof of concept, the following values: distance threshold of $\delta = 1\%$, bin width of $w = 3ms$, and an acquisition time interval of $t = 175ms$.

A. Tests and Testbeds

Our validation is performed by using the European Research network. In this testbed we performed a set of more than 500 experimental tests using twelve different testbeds across Europe. The testbeds cover a total of 5 countries and 4 different access technologies (LAN, xDSL, UMTS and WiFi) with an overlay architecture over the Géant research network.

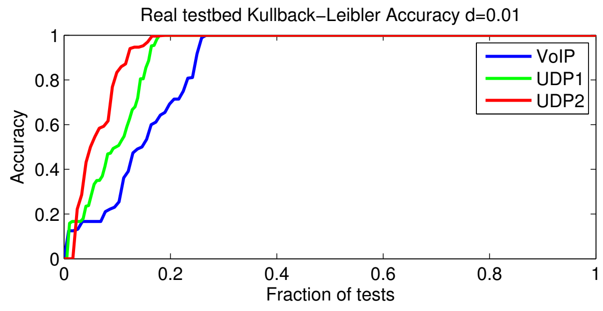
We evaluated the performance of our system by actively generating UDP traffic on the network with different properties. Specifically, we focus on three different sets of tests. The first one simulates a VoIP conversation with a used bandwidth of $64Kbps$. We label this traces as (synthetic) `VoIP`.

The second type of traffic is a periodic flow with average packet rate of ~ 96 packets per second, with MTU size packets amounting to a total of $1Mbps$ of UDP traffic. We call this trace `UDP1`. Finally, the third type of traffic is an average sized, high rate UDP flow of 900 packets per second with $\sim 1.4Mbps$. We call this test `UDP2`.

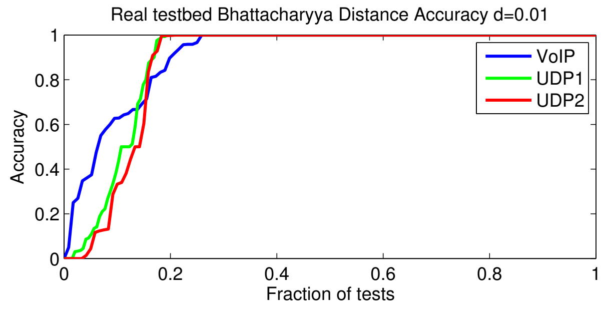
B. Methodology

In order to perform the validation we follow a structured methodology, which guarantees repeatability and automatic result extraction in the tests.

For each test we collect the full trace on both end nodes, i.e., ingress and egress. Once all the trace for the test is collected, we match the packets extracting the network performance metrics as described in [10], using them as reference quality (perfect knowledge). Afterwards, we identify the different network disruption periods with the reference results acquired above. Then, we apply our algorithm off-line, where we register the required control traffic due to *Learning*, and the estimated network disruption periods. We do this both for KLD and for BD. Finally, we match the network disruptions with the ones obtained with the perfect knowledge.



(a) Kullback-Leibler Divergence Accuracy



(b) Bhattacharyya Distance Accuracy

Fig. 1. Accuracy for Synthetic traffic over the European Network

C. Accuracy and Resources requirements

In order to study the behavior of our system, here we discuss the achieved accuracy together with the analysis of the required resources and we compare the results between KLD and BD.

In Figure 1 we show the Cumulative Density Function (CDF) for the accuracy results in the case of KLD Fig. 1(a) and BD Fig. 1(b), each result details separately each traffic profile. The X-axis of the figures holds the test number (normalized to 1) and the Y-axis the actual accuracy. The figure considers all the tests, including the ones without network disruptions.

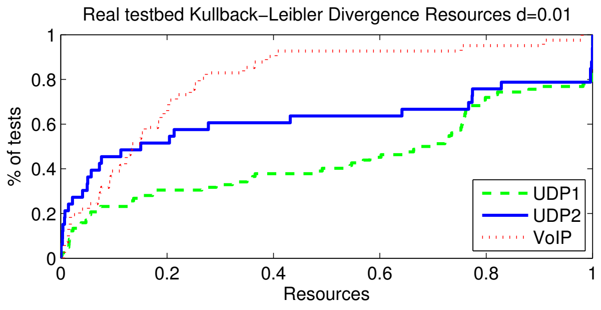
As it can be observed, in terms of accuracy both solutions have a similar performance, where KLD is marginally better except for VoIP flows where the performance is noticeably improved. In order to analyze the accuracy errors experienced in both alternatives, we manually checked that most of the failures in the disruption detection were due to isolated bins with validity very close to ν's boundary with no practical interest in a real deployment given its short duration. Further analysis is required in order to observe whether the accuracy is dependent on the number of disruptions or on the traffic profile.

Despite of this, when we study the resources consumption, in Fig. 2 we can observe that BD requires a noticeable less amount of resources, opposed to the accuracy results where the performance was comparable. This reduction in the required resources makes BD a better candidate for use in a real deployment despite the reduction in accuracy in some cases. If we observe the results in more detail, we can notice that in the case of VoIP, the resource requirements are closer for both alternatives, even if consistently lower for BD. Hence, we consider as part of our future work, to extend our design

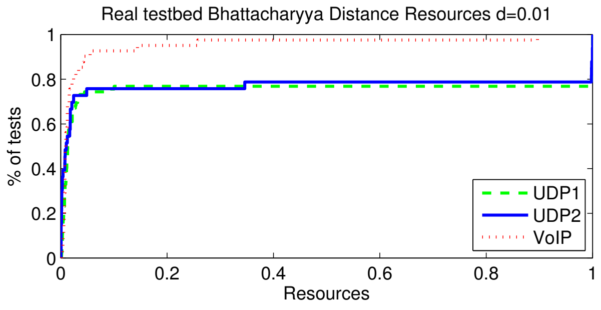
	Violations	*Detected KLD*	*Detected BD*	*Accuracy BD*	*Accuracy KLD*	*Resources KLD*	*Resources BD*
VoIP	7216	6096	4773	0.845	0.661	0.198	0.044
UDP1	62264	60108	59271	0.966	0.952	0.551	0.249
UDP2	24863	22265	23922	0.980	0.962	0.459	0.280

TABLE I
VIOLATION DETECTION, $\delta = 0.01$

and propose a dynamic adaptable algorithm in order to use different distance algorithms depending on the traffic profile in order to benefit of the higher accuracy in KLD for low rate flows, while using the very low resource consumption obtained by BD in the case of high rate traffic.



(a) Kullback-Leibler Divergence Resources



(b) Bhattacharyya Distance Resources

Fig. 2. Accuracy for Synthetic traffic over the European Network

To complement the results, in Table I we summarize the difference in terms of accuracy and resource requirements of KLD and BD. The table shows the aggregated total amount of bins with violations, together with the amount our algorithm could detect, without considering tests without violations. Next pair of columns highlight the overall accuracy. Finally, the last pair, details the average amount of resources needed for the reporting.

V. CONCLUSIONS

We have presented a network disruption detection system, which improved the accuracy of previous solutions by reducing most of the time the the performance metric computation and the synchronization between the edges of the network. The basis of the proposed system is the comparison using Bhattacharyya Distance of the distribution of Inter-Packet Arrival Time (IPAT) between real-time information, and reference distributions smartly obtained from the same network traffic through a robust *Learning Strategy*.

Our methodology was validated with a set of tests using synthetic traffic in a European-wide testbed with different access technologies. The obtained results show that we can reduce the required resources considerably, with a mild effect on the final accuracy of the network disruption detection.

We left as an important part of our future research to improve the design of the system by using a dynamic solution for distance computation, which can use the higher accuracy of Kullback-Leibler Divergence on low rate flows, with the lower resource requirements of solutions such as Bhattacharyya Distance on high rate traffic.

ACKNOWLEDGMENTS

We would like to thank Philippe Owesarski and Yann Labit for all the good advice during the initial phases of this work. We also are indebted with the EuQoS testbed owners who helped us during the project in performing the tests presented here.

REFERENCES

[1] René Serral-Gracià, Alberto Cabellos-Aparicio, and Jordi Domingo-Pascual. Network performance assessment using adaptive traffic sampling. *IFIP Networking*, pages 252–263, Singapore, May 2008.
[2] Joel Sommers, Paul Barford, Nick G. Duffield, and Amos Ron. Accurate and Efficient SLA Compliance Monitoring. In *Proceedings of ACM SIGCOMM*, pages 109–120.
[3] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Multiobjective monitoring for SLA compliance. *IEEE/ACM Transactions on Networking*, 18(2):652–665, 2010.
[4] 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.
[5] 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.
[6] Joel Sommers, Paul Barford, Nick G. Duffield, and Amos Ron. Improving Accuracy in End-to-end Packet Loss Measurement. *SIGCOMM Comput. Commun. Rev.*, 35(4):157–168, 2005.
[7] Tanja Zseby, Sebastian Zander, and Georg Carle. Evaluation of Building Blocks for Passive One-Way-Delay Measurements. In *Passive and Active Measurements Conference*, 2001.
[8] perfSONAR - http://www.perfsonar.net.
[9] J.W. Boote, A. Hanemann, L. Kudarimoti, P. Louridas, L. Marta, M. Michael, N. Simar, and I. Tsompanidis. Quality assurance in perfSONAR release management. 2007.
[10] René Serral-Gracià, Pere Barlet-Ros, and Jordi Domingo-Pascual. Coping with Distributed Monitoring of QoS-enabled Heterogeneous Networks. *4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pages 142–147.