Distributed Sampling for On-line SLA Assessment

René Serral-Gracià, Pere Barlet-Ros, Jordi Domingo-Pascual
Advanced Broadband Communications Centre, Technical University of Catalunya (UPC), Spain
{rserral, pbarlet, jordid}@ac.upc.edu

***Abstract*— New business infrastructures over the Internet pose a new set of traffic constraints. In particular, multimedia and interactive contents require guarantees of bandwidth and delivery time. The broad deployment and real-time nature of this class of applications require the provisioning of specific resources in the network to guarantee a certain level of Quality of Service (QoS). QoS techniques need ways of obtaining feedback about the status of the QoS enabled paths, for example, for checking the fulfilment of Service Level Agreements (SLA). A possible technique for obtaining such feedback is by passively monitoring the network traffic.**

The issue with traffic monitoring is the additional bandwidth needed by the control traffic generated by the different collection points in order to synchronise its acquired QoS metrics. Moreover, passive monitoring at line speed is an expensive process both in terms of resource consumption and price. However, some of these requirements can be significantly reduced by using traffic sampling.

This paper presents a novel methodology for intra-domain on-line distributed QoS monitoring, which makes an efficient use of the network resources by employing distributed sampling mechanisms.

The proposal is validated by performing real tests on an European-wide testbed. Our results show that the sampling technique can significantly reduce the traffic overhead, while obtaining very accurate estimations of the One Way Delay performance metric.

I. Introduction

As heterogeneity of services grows in the Internet new functionalities are required. Service providers and operators are offering a large amount of multimedia contents for global broadcasting (video streaming, on-line TV or videoconferencing) which produces a considerable increase of real-time traffic in the network. Some of these services usually come with an associated fee, which is tied to a proper quality of service. Given that Internet is based on a *best-effort* policy, offering these services is not always easy. In this direction, research efforts are set on efficient Quality of Service (QoS) mechanisms that focus on ensuring the contract fulfilment.

QoS mechanisms permit to reserve resources, to prioritise among different services, and even to guarantee the network parameters available for a connection. The fact that network usage is not static poses two major issues on QoS policies and Internet in general. First, not all the traffic is constrained by the same network parameters. For example, Web traffic does not have tight demands in terms of *One Way Delay* (OWD), while VoIP needs low latency and low *Packet Loss Ratio* (PLR) in order to keep the conversation within acceptable thresholds of quality. And second, network conditions are prone to change over time. In fact, the network can have high variations due to cross-traffic or congestion. Hence, even QoS policies require methods for querying the network status, both for monitoring the QoS traffic constraints and for reacting in case that compliance is going to be broken.

The most usual ways of acquiring the status of the network are: i) by using SNMP on the routers, ii) by querying the end-points of the communication, iii) by monitoring the traffic along the path.

The first option gives too coarse information, and thus it is not suitable for detailed metric reporting, because it is not possible to correlate the obtained information with specific flows or services.

The second option is a good approach for end-to-end user's perception of the QoS. The problem is the complexity and cost of deploying the system in the final users' premises.

The last option, which is the one used in this work, permits to deliver specific and precise information of the desired flows along their path. This is accomplished by placing collection points in the ingress and egress routers of each domain that report to an analyser unit on a per domain basis. We call this monitoring infrastructure the *Network Parameter Acquisition System* (NPAS) [1]. The main issue highlighted in our previous work [1] and in such systems in general, is the high bandwidth required to send the control traffic among the different entities involved in the QoS metric reporting.

This paper extends our previous research by using distributed traffic sampling techniques that significantly reduce the required reporting traffic (i.e. control traffic between the collection point and the analyser unit). Our solution operates by choosing specific packets in all the collection points of the NPAS, which are matched using hash tables to reduce the hardware resources needed by the system. NPAS is designed so all the collection points can operate independently of each other in an autonomous fashion.

The reporting architecture and the sampling optimisations proposed in this work are validated by a set of real tests performed in an European-wide overlay testbed available in the EuQoS project [2], and by analysing real traces acquired from the Catalan Academic Network. The results prove that the sampling technique can reduce the control traffic by one order of magnitude with very small errors in the OWD estimation.

The rest of the paper is structured as follows: Section

This work was partially funded by IST under contract 6FP-004503 (*IST-EuQoS*) and NoE-038423 (*Content*), MCyT (Spanish Ministry of Science and Technology) under contract TSI 2005-07520-C03-02 and the CIRIT (Catalan Research Council) under contract 2005 SGR 00481.

II overviews related work about traffic sampling in QoS scenarios. Section III presents the basic on-line reporting architecture. The paper continues in Section IV with the proposed sampling methodology. The validation along with the results are presented in Section V. In section VI we discuss about possible enhancements of the solution. Then the paper finishes with the concluding remarks and the future work.

II. Related Work

Traffic sampling for network metric estimation is a topic covered many times in the past (e.g [3]–[7]). Sampling can be applied to a broad range of fields with the goal of reducing the needed resources to compute a given metric. This paper focuses on using traffic sampling for efficient QoS metric acquisition.

Such efforts on sampling Internet traffic evolved on *Packet SAMPling* (PSAMP), an IETF Working Group in charge of defining and standardising the different sampling techniques applicable on network measurements.

In order to match each packet on each collection point, of all the different approaches proposed by PSAMP, our work requires a sampling method with a deterministic packet selection function. A possible solution was first presented by Duffield et al. [8], where the authors present a methodology for inferring the packet's trajectory by using *hash sampling* the technique was called *trajectory sampling*. The solution reduces the analysed traffic by using sampling together with a hash function over some selected fields on the packet's header. The main difference between trajectory sampling and our approach is that, while trajectory sampling uses hash sampling in order to decide which packets to select regardless of its origin, we must consider all the flows within SLA contracts, and hence we must extend the usage of hash sampling. Specifically we define a two level hash table to overcome this limitation.

In the distributed QoS metric analysis field another sampling approach is used in [9], where well-known traffic sampling methodologies are used for computing one-way delays and packet losses in ATM networks. The caveat of that approach is that it relies on the information stored in the ATM cells, not permitting its application in other technologies. Moreover, their solution only considers scenarios with two static monitoring points on the network, while our proposal goes one step further giving a generic solution for technology independent intra-domain QoS parameter acquisition.

More recently, T. Zseby in [5], [7] presents an architecture for SLA validation with static monitoring points. The author does not consider the full requirements for a large scale deployment of the architecture, while our solution instead involves an arbitrary number of collecting points and it is used for general SLA assessment.

NPAS is not the only system for SLA assessment, there is another solution, namely perfSONAR [10]. perfSONAR is a tool intended to distributively monitor any network metric. The authors present a methodology that provides meaningful network performance indicators. The main difference between this tool and our proposal is that while perfSONAR limits the study to the link status by active traffic generation, NPAS analyses directly the network metrics by passively collecting the traffic. In our opinion, using active traffic generation is less precise in general since we cannot guarantee that the network treats the active probes with the same priority as normal traffic.

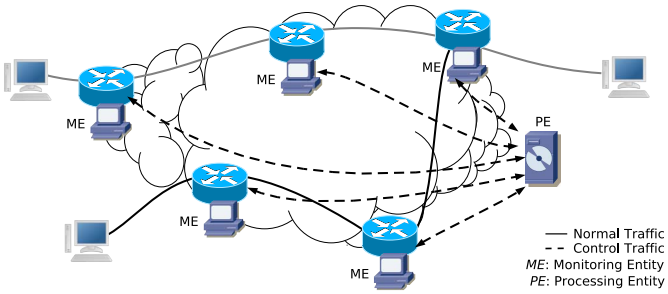

Fig. 1. Architecture deployment example

III. Network Parameter Acquisition System

On-line QoS monitoring requires an infrastructure for QoS metric gathering. This work is centred in intra-domain scenarios and extends our previous research [1] by reducing the required resources for distributed traffic monitoring.

Supporting inter-domain scenarios implies the definition of specific aggregation methods in order to minimise the inter-domain link overhead due to reporting traffic, such analysis is left as an important part of our future work.

The system collects the traffic on each ingress and egress points within the network, only including the traffic subject to QoS policies. The computed network metrics are One-Way Delay (OWD), IP Delay Variation (IPDV), Packet Loss Ratio (PLR) and used bandwidth, which are the most relevant, according to [11].

This infrastructure delivers a framework which publishes the intra-domain traffic metrics to higher layer entities in the QoS control plane, to assess whether the SLA contracts are fulfilled or not. This mechanism is used for triggering the appropriate actions if needed. Both actions and management policies are out of the scope of this paper.

Figure 1 shows a generic network scenario where the system is deployed. It is composed of the following entities:

1) The Monitoring Entity (ME): It is in charge of the traffic collection via selection filters. The traffic selection policy aggregates the data in a per flow or per Class of Service (CoS) basis, but it can be easily extended to other configurations.

The traffic information collected by MEs is sent to a per-domain analyser entity *(Processing Entity - PE)* which computes the network metrics of all the traffic under analysis.

2) Processing Entity (PE): It is the central gathering point within the domain. It performs most of the processing, identifies aggregates, matches the information of the packets coming from the MEs and computes the QoS metrics.

Further detail in the specification and bandwidth usage of this solution can be found at [1].

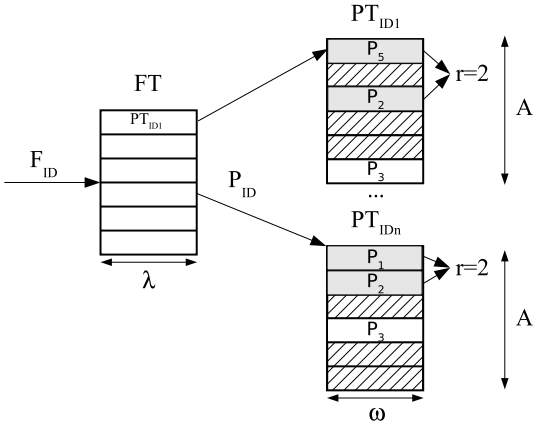

Fig. 2. Structure of Flow (FT) and Packet (PT) Hash Tables

IV. Distributed Sampling

The main limitation of the presented solution, which is addressed in this paper, is the high bandwidth requirements needed to report per packet information to the PE. This limitation is solved by using traffic sampling over the collected packets in the ME.

The complexity of applying sampling in this distributed scenario is that centralised techniques (e.g. the techniques presented in [4]) are not well suited for this task, since they do not guarantee that all MEs capture exactly the same set of packets. Thus, accurately determining packet losses or one way delays is not feasible. We solve this issue by defining a deterministic sampling technique to match exactly the same packets all over the different ME, and later computing the network metrics. This technique permits all the different ME to collect traffic independently, only the selection function at configuration time has to be shared by the MEs.

The distributed sampling method proposed in this paper is based on the hash sampling technique proposed in [8]. Hash sampling computes a hash function over a set of fields on the packet's header. Thus, a packet is only analysed if it falls within specified positions in the hash table. This permits to efficiently control the resources and the sampling rate of the solution. In order to adapt this technique to our environment, using only one hash table is not sufficient, since the QoS monitoring framework has to guarantee that all the flows under QoS contract are considered for the analysis. Thus, the sampling must be applied within the flows, not directly to all the collected traffic. Therefore we identify the packets using two different keys, the *flow identifier* and the *packet identifier*.

A. Hash functions analysis

The above requirements are implemented using a two level hash table as shown in Figure 2. The flows and the packets tables are indexed by two different hash functions.

1) Flow Identifier hash function (F_H): The flow identifier (F_{ID}) has 32 bits and it is obtained using the typical 5-tuple for flow selection: Source and Destination Addresses, Source and Destination Ports and Protocol.

The flow identifier is hashed by a randomly generated H3 hash function [12], which distributes the flows uniformly and unpredictably. This way, we guarantee minimum collisions in the first hash table. If a collision is found, a linked list of the colliding flows is created. In our context we must resolve collisions on the flow tables, since the system must consider all the flows under SLA constraints.

2) Packet Identifier hash function (P_H): The packet identifier (P_{ID}) has 32 bits, and it is generated by a CRC32 from 27 bytes of the packet's payload, as proposed in [13]. However the author states that other fields such as IP Source and Destination or Protocol are also required, but in our scenario these fields are already considered in F_{ID}, and hence ignored on the P_{ID} computation. Moreover, as proved in [8], using 27 bytes of payload minimises the probability of different packets obtaining the same identifier per collection period.

Using both functions presented above allows us to quickly insert each desired packet's QoS information. Each flow hash table has a size of A packets, from where only the first r are considered (see Figure 2 where we suppose $r = 2$ as an example). Possible collisions on the second hash table are ignored (the new packet silently overwrites the previous one). Hence, a parameter to consider is the size of the packet hash table. As we detail in the results, for fairly small hash tables collisions are rarely found.

When a new packet arrives to a ME it is classified to the flow where it belongs, the Flow hash Table (FT) indicates the specific table (PT) where the packet must be stored. Then P_H is computed and the packet information is stored into the hash table.

B. Applying the Sampling

Once we have all the information stored into the hash table, the last step is to choose which packets will be sent to the PE. In this paper we consider that the sampling rate is applied uniformly within all the ME involved on the measurement. Let ρ be the upper bound of the sampling rate. In some situations ρ might not be applicable, since the number of packets received by ME is discrete, then the minimum applicable sampling rate to flow f is $\rho_{(f,1)} = \frac{1}{R_f}$, where R_f holds the number of packets received during the time interval in flow f.

Analogously, the maximum sampling rate is achieved by $\rho_{(f,R_f)} = \frac{R_f}{R_f} = 1$. Hence all the possible sampling rates for flow f are: $\mathcal{R}_f = \{\rho_{(f,1)}, \ldots, \rho_{(f,R_f)}\}$. Then we define the applied sampling rate to f as, $\rho_{(f,i)}$ which $\forall i : 1 \leq i \leq R_f$ is closest to the decided ρ.

Hence, the total effective applied sampling ρ_e is defined as

$$\rho_e = \frac{\sum_{i=1}^{|f|} r_i}{\sum_{j=1}^{|f|} R_j} \tag{1}$$

where $|f|$ is the number of flows among all the MEs and $r_i = \rho_i R_i$, the considered packets in flow i for the study. Therefore ρ_e defines the effective sampling rate on the system, and thus the real required bandwidth for the control traffic. We must

highlight that this value depends on the traffic existing on the network, thus not known in advance.

The application of the sampling rate to the packet's hash table must be done in a deterministic manner in order to synchronise the different ME. Hence, a time window t (referred to as bin from now on) is negotiated among the ME, such t triggers the selection of the packets and flushes the hash tables. The sampling rate is applied by selecting the first r_f packets of each monitored flow on all the MEs. r_f is obtained from the applied sampling rate ρ_f and from R_f: $r_f = \rho_f R_f$.

Note that the hash tables on each ME will have exact copies for the common flows except when there are packet losses, that is when a packet appears on some ME and not on the others down the path. After t time units we transfer r_f packets of each table towards PE, which matches all the packets over all the MEs reporting the QoS metrics.

Another point to consider is when the OWD between two ME is big, it can happen that some collected packets on the first ME have still not reached the second, to avoid this, the PE holds an historic of some time windows t. Due to space constraints we do not discuss here the optimal value for this historic. The reader is referred to [1] for a detailed description of such parameter, in this work we assume that it is big enough to consider every single packet arriving to PE.

C. Memory and Bandwidth Requirements

Since we might have gaps within the packet hash table, we have to allocate memory for A entries for each P_{ID} to chose the first r_f. Assuming a flow hash table of τ active flows and λ bits per F_{ID}, with ω bits per P_{ID}, the overall memory Θ required per ME by the system is upper bounded by expression 2:

$$\Theta \leq \lambda\tau + \omega\tau A \tag{2}$$

Where $\lambda\tau$ is the size in memory of the flow hash table, $\omega\tau$ is the size of one packet hash table. In our system $\lambda = \omega = 32bits$ which produces $\Theta \leq 32\tau(A+1)$. Where Θ depends on the active flows τ and the size of the hash table A, and not on the actual packets traversing the ME.

Regarding the bandwidth, using small r_f implies less required bandwidth, lower sampling rate and less precision for the results as we have less information to estimate the real values.

Moreover, t determines the flushing period of the hash table and thus bounding its reporting interval. In the next section we discuss the appropriate value for A and t in a real scenario along with the memory and bandwidth requirements of its deployment.

V. Tests and Results

In this section we validate the proposed sampling mechanism by using an experimental testbed with a large set of tests. We detail the experimental tests, the parameter selection (A,t) and the accuracy obtained by applying several sampling rates to the traces. The analysis is completed by the study of the real memory requirements of the solution.

A. Experiments

For validating the proposed sampling mechanism we performed a set of 520 tests using the 12 different testbeds from EuQoS [2] partners, covering a total of 5 countries and 4 different access technologies (LAN, xDSL, UMTS and WiFi) with an overlay architecture on the Géant research network [14]. For this work the testbed is configured to act as a single domain, with one ME deployed on each technology, amounting to 12 ME and 1 PE.

The tests were performed by actively generating synthetic traffic, emulating different applications, with a broad set of combinations of different packet rates, packet sizes and different hours. A wide range of packet loss and delay variation conditions were encountered given the different cross-traffic found on the network at different days, hours and physical locations of the testbed. That gives good set of tests to validate the proposal.

In order to ease the validation, instead of directly sampling the traffic, we collected the complete trace and applied the different sampling rates off-line. For the sake of the exposition we only show results for the following sampling rates: $\rho =$ 0.35, 0.15 and 0.05, that correspond to effective rates (using expression 1) $\rho_e = 0.31$, 0.11 and 0.04, which in our opinion represent examples of low, medium and high sampling rates. With this methodology we were able to compute the metrics for each sampling rate and its accuracy by taking as reference the complete original trace. Therefore we can compute the accuracy of our solution.

In summary the methodology for validating the accuracy of the technique consists of the following actions:

1) Generate the test traffic and gather all the traces on MEs.
2) Apply off-line the different sampling rates ρ_f with the chosen bin size t. This is the information sent to the PE.
3) At the PE, for each bin and ρ_e the average OWD is computed.
4) Finally each sampled OWD result is compared with the original, as detailed later in this section.

As a second set of experiments, in order to assess the costs and the memory requirements of a deployment on a real network, we performed a series of collections with real traffic during several periods on the Catalan Academic Network, which is a Gigabit ethernet link with a sustained traffic of around 360Mbps, the access to the traces has been provided by the SMARTxAC project [15].

B. Selecting t

t determines the interval of time we use to fill the hash tables and send the sampled traffic information to the PE. Hence, t impacts on the system in three different aspects. First, in the reporting latency, the bigger is t the longer it will take the system to react. Second, the effects of potential collisions on the hash table, the bigger is t the more often packets enter to the hash table, hence the collisions might increase. And third, the smaller is t the more packets are being sent by the ME, thus more control traffic overhead is generated.

A	*p bin with Col.*	*Max. #Col*	*Avg. #Col*
19	0.3	1421	8.65
101	0.1	1339	4.87
1297	0.03	143	2.5
16979	~ 0	60	1.88

TABLE I
COLLISIONS FOR DIFFERENT SIZES OF *A*

In order to select an appropriate t we analysed the OWD and the IPDV of all the performed experiments as we described in [1]. In that work we show that a desirable bin size is $t = 175ms$ which is the value we chose for the validation.

C. Selecting A

One of the basis of our infrastructure is the use of a hash table. The size of the table impacts directly to the system's performance. On the one hand, the smaller is the table the less memory the system requires to operate. On the other hand, the bigger is A the lower is the collision probability. We have to point out that having collisions is not a big issue because we are using sampling. Therefore we will not consider all the packets for the analysis, as long as we have r_f packets to study per bin and per ME. So we advise using $A \gg max\{r_f\}$ for any flow on the system.

In order to evaluate the effects of such table size we used the Catalan Academic Network traces described before. There the results show that typically on a network where $t = 175ms$ we have an average of ~ 3500 flows with a total amount of $\sim$ 10000 packets per bin. It results on an average of ~ 3 packets per bin on each flow, but with a maximum packets per bin of 1440. This values are upper bounds *in our scenario*, since we consider all the traffic on the link, while for SLA assessment we would only consider part of the traffic. Anyway we think that the analysed traces are quite representative of the normal behaviour on the Internet.

We used this information to study the effects of different hash table sizes, specifically we choose different prime numbers $A = 19, 101, 1297$ and 16979. Using prime numbers together with a robust hash function is advisable in order to minimise the collisions [8]. From the tests we computed several statistics as shown in Table I. The table details the probability of having a bin with a collision, in our case it is $\sim 30\%$ for the minimum size and is rapidly reduced down to 0.01% in the case of the biggest considered A. From the bins with collision we computed its maximum and average length (columns 2 and 3 respectively), for example with size 1297 in the worst case we only have 143 collisions, while from all the bins with collision their average is only 2.5. As it can be noted the gain of further increasing A from 1297 is not worth the cost increase in terms of memory, since we have a really small amount of collisions.

In [8] the authors reccommend to use $A = 16979$, with our strategy we can reduce this value because we classify the packets into flows, reducing this way the amount of packets entering each hash table. Therefore the rest of the paper assumes $A = 1297$.

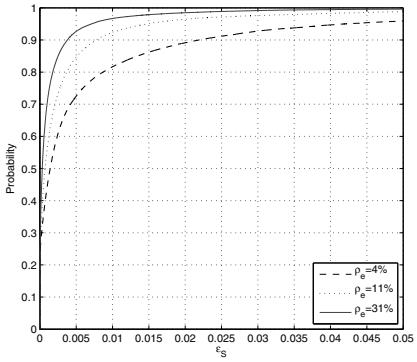

Fig. 3. CDF: relative error of the sampling estimate

ρ_e	99th prc.	95th prc.	50th prc.
0.04	0.16	0.04	0.001
0.11	0.08	0.02	~ 0
0.31	0.03	~ 0	~ 0

TABLE II
RELATIVE ERROR PERCENTILE FOR OWD

D. Validation

We focus the validation of the platform with OWD accuracy analysis. Some discussion about PLR is left for further study as detailed in Section VI.

The study of the OWD accuracy is performed by comparing the estimated values using sampling with the real results of the full trace. Therefore, all the analysis computes relative error values as detailed on equation 3:

$$\varepsilon_S = \left| 1 - \frac{\hat{x}}{X} \right| \tag{3}$$

Where X is the real value taken from the complete trace and $\hat{x}$ stands for the estimate obtained by applying sampling.

Expression 3 is applied to all the averages obtained from the actions described previously. Figure 3 presents the Cumulative Distribution Function (CDF) of the relative error of *One Way Delay*. In the figure the X-axis shows the relative OWD error and the Y-axis holds the cumulative probability. Table II complements the results with the most relevant percentiles.

As it can be noted, the more we increase the sampling rate the better is the accuracy. In the worst presented case with $\rho_e = 0.04$, the error in estimating the $95th$ percentile of OWD only shows 4% error, while for $90th$ percentile it is further reduced to 2%.

In a SLA controlled scenario admissible OWD for typical real-time applications range from 50 to 200ms (as in the case of VoIP communications [16]). Having relative errors lower than 2% give a usable estimate in order to decide whether the quality is within good thresholds.

E. Memory and Bandwidth analysis

With the proposed sampling technique, we reduce the required control traffic proportionally to ρ_e. The associated cost by using this approach is the increase in memory requirements of the solution. Each ME must hold the hash structure presented before. We studied the memory requirements of deploying this solution on the Catalan Academic Network assuming the traffic conditions detailed previously.

With 3500 flows per bin, we need $\Theta \simeq 18Mbytes$ on each ME for the hash table. This value has been computed using equation 2.

The amount of required memory on the PE depends on each r_f and on the amount of flows and ME. Assuming the above network with 12 ME with 20 packets per bin and $\rho_e = 0.11$, then $r_f = 2$ hence the PE needs $\sim 328Kbytes$.

VI. Discussion

In this paper we discussed thoroughly the application of distributed sampling in order to estimate the OWD. We left out the study about PLR.

Analysing PLR is a more complex issue since for low rate flows the system collects few packets per bin. It is well known [17] that the achieved accuracy (within a 95% confidence interval) when classifying sampled traffic into categories is bounded by $\varepsilon \leq 1.96\sqrt{\frac{1}{r}}$, where r is the amount of sampled packets within the category (packet losses in our case).

Therefore estimating PLR with the above technique leads to some inaccurate results with low rate flows. Hence this technique is not suitable for the estimation of this metric. In this regard we have performed some further research [18] with dynamic adaptive sampling which leads to much better results (i.e. our preliminary study we obtain errors below 10%).

The adaptive sampling solution instead of specifying a static sampling rate to the system, permits to evaluate where is important to focus the collection resources in order to increase dynamically the sampling rate on the flows with more probability of losses, for further details the reader is referred to [18] and [19].

VII. Conclusions and Future Work

This paper proposes a technique for reliable and efficient on-line SLA assessment. Even if this work only covers intra-domain scenarios, the architecture is ready to be extended to an inter-domain environment independently of the existing access technology.

With the aim of reducing the required bandwidth for the on-line reporting, the proposed technique combines the usage of sampling algorithms in conjunction with reliable hash functions to obtain an efficient distributed system. The whole architecture has been validated by broad deployment and testing on an European wide testbed using Géant network.

The system reduces the generated control traffic by one order of magnitude by using sampling rates down to 0.04 with a relative error smaller than 2% for 90% of the cases for One Way Delay. If more precision is required, our experiments show that increasing the sampling rate (0.11 in our experiments) reduces the estimated error to 1% for 95% of the cases, still reducing ~ 10 times the required resources compared with the per packet reporting.

We also discovered that estimating PLR is much more complex. Hence, in order to improve its estimation, we leave as an important part of our future research the proposal of a dynamic adaptive sampling technique to solve this issue.

Another point for further study is the architecture definition and the experimental study of the inter-domain scenario. This analysis would require the definition and capabilities of the inter-domain entities and the interconnection protocols with the peer domains.

References

[1] 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, 2008.

[2] [IST] EuQoS - End-to-end Quality of Service support over heterogeneous networks - http://www.euqos.eu/, September 2004.

[3] Nick Duffeld. Sampling for Passive Internet Measurement: A Review. *Statistical Science*, 19(3):472–498, 2004.

[4] C. Veciana-Nogués, A. Cabellos-Aparicio, J. Domingo-Pascual, and J. Solé-Pareta. Verifying IP Meters from Sampled Measurements. In *Kluwer Academic Publishers. Testing of Communicating Systems XIV. Application to Internet Technologies and Services. IFIP TC6/WG6.1. TestCom*, pages 39–54, 2002.

[5] Tanja Zseby. Deployment of Sampling Methods for SLA Validation with Non-Intrusive Measurements. In *Passive and Active Measurements*, 2002.

[6] Tanja Zseby. Stratification Strategies for Sampling-based Non-intrusive Measurements of One-way Delay. In *Passive and Active Measurements*, 2003.

[7] Tanja Zseby. Comparison of Sampling Methods for Non-Intrusive SLA Validation. In *E2EMon*, 2004.

[8] N. G. Duffield and Matthias Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Netw.*, 9(3):280–292, 2001.

[9] Irene Cozzani and Stefano Giordano. Traffic sampling methods for end-to-end QoS evaluation in large heterogeneous networks. In *Computer Networks and ISDN Systems 30*, pages 1697–1706, 1998.

[10] A. Hanemann, J. W. Boote, and et. al. PerfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring. In *Third International Conference on Service Oriented Computing, LNCS 3826, ACM SIGsoft and SIGweb*, pages 241–254, 2005.

[11] ITU-T Recommendation Y.1540. Internet protocol data communication service - IP packet transfer and availability performance paramteters, 12/2002.

[12] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[13] Tanja Zseby, Sebastian Zander, and Georg Carle. Evaluation of Building Blocks for Passive One-Way-Delay Measurements. In *Passive and Active Measurements Conference*, 2001.

[14] GÉANT - http://www.geant.net.

[15] P. Barlet-Ros, J. Sol?-Pareta, J. Barrantes, E. Codina, and J. Domingo-Pascual. SMARTxAC: A Passive Monitoring and Analysis System for High-Speed Networks. In *Terena conference. Journal Campus-Wide Information Systems. ISSN: 1065-0741*, volume 23 Issue 4, pages 283–296, 2006.

[16] ITU-T Recommendation G.107. The E-model, a computational model for use in transmission planning, 03/2005.

[17] Baek-Young Choi, Jaesung Park, and Zhi-Li Zhang. Adaptive Packet Sampling for Accurate and Scalable Flow Measurement. In *Proceedings of IEEE Globecom '04*, 2004.

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

[19] René Serral-Gracià, Albert Cabellos-Aparicio, and Jordi Domingo-Pascual. Packet loss estimation using distributed adaptive sampling. In *End-to-End Monitoring*, Apr 2008.