SLA Monitoring System of QoS Parameters to Network Performance Metrics Mapping

Aleksandar Tsenov[1] and Todor Georgiev[2]

Abstract – **Service-level-agreement (SLA) monitoring measures network Quality-of-Service (QoS) parameters to evaluate whether the service performance complies with the SLAs. It is becoming increasingly important for both Internet service providers (ISPs) and their customers. However, the rapid expansion of the Internet makes SLA monitoring a challenging task. As an efficient method to reduce both complexity and overheads for QoS measurements, sampling techniques have been used in SLA monitoring systems. In this work, using an efficient sampling strategy, which makes the measurements less intrusive and more efficient, a network performance monitoring software model is introduced, which monitors such QoS parameters as packet delay, packet loss and jitter for SLA monitoring and verification.**

Keywords – **Service-level-agreement (SLA), Quality-of-Service (QoS), SLA monitoring**

I. Introduction

Internet Service Providers (ISPs) now offer service level agreements (SLAs) routinely to their customers. Management needs contractual guarantees that business objectives are met, and end-users demand assurance that their critical network applications and services are available when needed. The availability of SLAs and a means to validate them gives management the confidence to move ahead. The wide adoption of the E-business model has made it essential that service-providers deliver on SLAs in a quantitative and qualitative manner. This has driven the service-providers to seek consistent testing and measurement methods that make real sense of customer network performance.

An SLA is defined by the International Telecommunications Union (ITU) as "a negotiated agreement between a customer and the service provider on levels of service characteristics and the associated set of metrics. The content of SLAs varies depending on the service offering and includes the attributes required for the negotiated agreement" [1]. The Internet Engineering Task Force (IETF) defines SLAs in a similar way [2]. Figure 1 shows the main features of the SLAs.

Generally speaking, a good SLA should include these three key aspects:

Service level objectives: encompass Quality-of-Service (QoS) parameters or class of service provided, service availability and reliability, authentication issues, SLA expiry date, and so on.

Service measuring components: specify the way of measuring service quality and other parameters used to assess whether the service complies with the SLA.

Financial compensation components: include billing options, penalties for breaking the contract, and so forth.

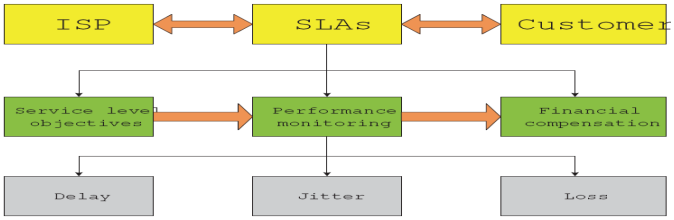


Fig. 1. Structure of service-level-agreements

II. Research Motivation and Contribution

SLA monitoring is about collecting statistical metrics about network performance to evaluate whether the provider complies with the level of QoS that the customer expects [3]. Therefore, accurate measurement and estimation of network performance becomes a key challenge in SLA monitoring. However, the implementation of measurement becomes increasingly difficult and complex due to the rapid expansion of the Internet. Traditional measurement tools, such as "ping", cannot satisfy the measurement requirements nowadays. Moreover, the dramatic increase in the speed of wide area backbones presents obstacles to complete statistics collection. The enormous amount of measurement data may significantly increase the cost and resource usage [4].

In order to solve these problems, sampling techniques are employed in SLA monitoring systems to reduce the quantity of control data and resources required to process it, and finally to reduce the measurement complexity and cost. Systematic sampling and random sampling are two widely used methods in existing monitoring systems, but both of them have severe limitations. Stratified random sampling can achieve higher estimation accuracy, but its high complexity may compromise its advantages.

The aim of this research, which has been funded through the research contract "**BY-TH 105/2005**", is to develop an efficient sampling strategy to make the measurement less intrusive and more efficient. Then a network performance monitoring software, which monitors such QoS parameters as packet delay, packet loss and jitter for SLA monitoring and verification, and which uses the proposed sampling strategy, needs to be designed. These objectives have been fully achieved. Firstly, a theoretical analysis of the performance of different sampling techniques is made [5], [6]. Secondly, a novel adaptive sampling strategy is proposed. Finally, QoS monitoring software model is proposed.

[1]Aleksandar Tsenov is with Telecom Department at Technical University of Sofia, "Kliment Ohridsky" Blvd 8, 1756 Sofia, Bulgaria, E-mail: akz@tu-sofia.bg

[2]Todor Georgiev is with the TELELINK EAD, Business Park, Building 13, Sofia E-mail: tgeorgiev@telelink.bg

III. QoS Metrics Review

A. Main Usages of Internet Measurements

As described in [5], the main usages of Internet measurements are Internet topology measurement, workload measurement, performance monitoring and routing measurement.

- Topology measurement: collects information on the network connectivity and graphical locations of network devices. With the rapid development of Internet, it becomes a challenge to track and visualise the complex Internet topology [5].
- Workload measurement: focuses on the collection of information on the resource usage of routers or switches and the link utilisation [5], [6].
- Performance measurement: is used by network users or researchers in analysing traffic behaviour on specific paths or the performance (e.g., packet delay, jitter, packet loss) associated with individual ISPs. A recent development in the industry is the monitoring of SLAs [5].
- Routing measurement: measures the dynamics of routing protocols and routing updates [6].

All of the mentioned above measurements can be performed in two ways – passive and active measurements.

The IETF's IPPM has developed series of standards called Requests For Comments (RFC) on network performance measurements. The standard metrics for measurements are defined in RFC 2330, which are listed below:

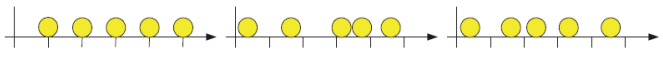
- Metric for Measuring Connectivity (RFC2678) [7];
- A One-way Delay Metric (RFC2679) [8];
- A One-way Packet Loss Metric (RFC2680) [9];
- A Round-trip Delay Metric (RFC2681) [10];
- One-way Loss Pattern Sample Metric (RFC 3357) [11];
- IP Packet Delay Variation Metric (RFC 3393) [12].

B. Sampling Techniques

In this chapter, three conventional sampling techniques, i.e., systematic sampling, random sampling and stratified sampling, and their characteristics are introduced.

Then a new sampling technique called "adaptive sampling" is presented.

Figure 2 illustrates these three sampling techniques.



a) systematic sampling b) random sampling c) stratified sampling

Fig. 2. Sampling techniques

Systematic sampling generates sampling traffic according to a deterministic function. Generation of the sampling traffic is triggered by either time (i.e., at fixed intervals) or packet count (i.e., every N-th packet). Figure 2.(a) shows periodic sampling with a period of T seconds.

Random sampling employs a random distribution function to determine when a sample should be generated. Typically the samples are generated according to a Poisson process. As shown in Figure 2.(b), random sampling may produce a varying number of samples in a given time interval. With random sampling, an unbiased estimate of the QoS metric can be achieved. However, the entirely random nature of the sampling process may also cause the undesirable effect that sampling intervals are not uniformly distributed, and therefore the network may not be sampled for a rather long time.

Stratified random sampling combines the fixed time interval used in systematic sampling with random sampling. Figure 2.(c) shows stratified random sampling with a period of T and a random sample is generated in each period.

IV. Adaptive Sampling

The proposed in this work sampling method is called adaptive sampling. In conventional sampling, the sample selection procedure does not depend on the observations made during the sampling, so that the entire samples may be selected prior to the start of the sampling process. In adaptive sampling, the procedure for selecting samples may depend on the values of the variable of interest observed during the sampling process. The primary purpose of adaptive sampling design is to take advantage of population characteristics to obtain more precise estimates, for a given sample size or cost, than is possible with conventional designs. For example, the dynamic nature of network traffic determines that sometimes the variable of interest (e.g., packet delay, packet loss, traffic quantity) may be smooth, while at another time, the variable of interest may present dramatic variations. Intuitively, given a fixed total sample size, a more accurate estimate can be obtained by changing the sampling rate adaptively such that the algorithm samples less during periods in which the variable of interest is smooth and samples more during periods in which the variable of interest varies dramatically. Figure 3 shows the adaptive sampling in two measurement intervals. In the measurement of interval i, the variable of interest presents dramatic fluctuation, so we select comparatively more samples; while in the measurement of interval i + j, the variable of interest changes smoothly, so we select comparatively fewer samples.

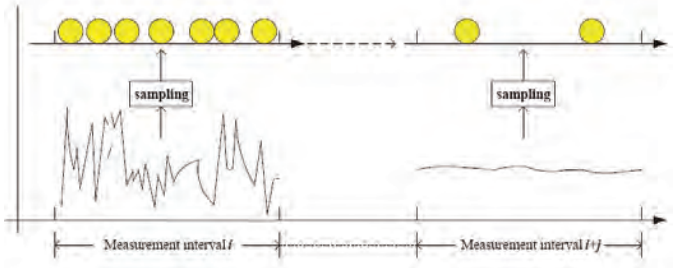


Fig. 3. Example of adaptive sampling

The following Figure 4 represents how important is the choice of the sampling period and how different the aspect of the traffic load, depending of the sampling period might be.

Despite its advantages, the real implementation of adaptive sampling may be difficult, which may compromise its advantages. For example, that for stratified sampling, the most accu-

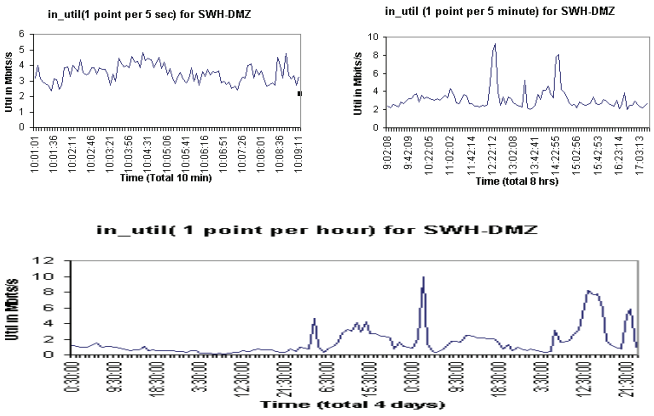


Fig. 4 Utilization with different averaging times [13]

rate estimate is obtained by allocating the number of samples in each stratum so that the number of samples in each stratum is proportional to the standard deviation of the variable of interest in the stratum. Then, to implement adaptive stratified sampling for packet delay measurements, the optimum sampling design should allocate the number of samples in each stratum to be proportional to the standard deviation of packet delay in that stratum. Therefore, to determine the optimum number of samples for the next stratum, the standard deviation of packet delay in the next stratum has to be **predicted**. In reality, the uncertainty and complexity involved in standard deviation prediction may compromise the advantage of using the adaptive stratified sampling technique.

As discussed above, adaptive sampling methods select samples adaptively according to values of the variable of interest observed during the sampling process. The key element of adaptive sampling is the prediction of future behaviour based on the observed behaviour. Hernandez et al. employee a linear prediction (LP) algorithm in their adaptive sampling method to measure the network throughput [14].

In this part, we propose an adaptive stratified sampling scheme, which employs a least-mean-square (LMS) linear prediction algorithm to predict the standard deviation of packet delay from past observations. Then the sample size for the next stratum is calculated from the predicted value of the standard deviation.

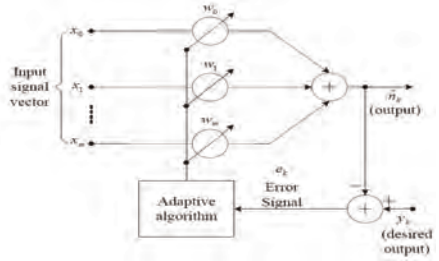


Fig. 5 Architecture of LMS algorithm [14]

The LMS algorithm is one of the most widely used adaptive linear algorithms. A significant feature of the LMS algorithm is its simplicity. It does not require measurements of the correlation function, nor does it require matrix inversion. The adaptive mechanism enables it to approximate the steepest descent algorithm automatically from sample to sample. Figure 5 shows the architecture of the LMS algorithm, where m is the order of the predictor, $y_{(k)}$ is the input vector and w_k is the prediction coefficient vector.

V. Modelling the Measurement Tool

In this part, we introduce the software design. Firstly, we introduce the software environment and the functionality of the software.

Architecture of the Measurement tool - MT System

MT should consists of two kinds of systems ; (a) Control System (CS) and (b) Measurement System (MS). Fig. 6 describes the architecture of MT.

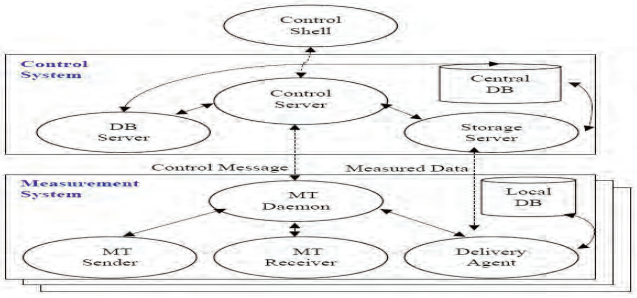


Fig. 6 Architecture of the proposed MT

Control System (CS): CS, main system of MT, receives commands sent from Control Shell (CSH), with which operator controls and manages AMT. CSH is console-based user interface. CS has three processes like Fig. 6; (a) Control Server (CSV), (b) Storage Server (SSV) and (c) DB Server (DBS). CSV receives commands from operator, parses the commands, and then processes the commands. CSV consists of three threads; (a) Main Thread (MAT), (b) Measurement Thread (MET) and (c) Polling Thread (POT). MAT receives command from CSH and processes it. MET initiates a measurement and POT checks the health of measurement systems and network. SSV collects measurement data from local database (Local DB) of each MS after the measurement and stores the data in the central database (Central DB). It is forked by CSV when preparing the collection. The collection is performed with the aid of Delivery Agent (DA) of each MS. DBS analyzes the gathered raw data and stores them into Central DB.

Measurement System (MS): MS has four processes like Fig. 6; (a) MT Daemon (MTD), (b) MT Sender (MTS), (c) MT Receiver (MTR) and (d) Delivery Agent (DA). After MTD, main process of MS, first registers itself in CS, it receives all the control messages from CSV, processes them and sends the result to CSV. For example, when CSV sends the measurement preparation message to the registered MTD of each MS, MTD receives the message to prepare measurement. It forks MT Sender (MTS) and MT Receiver (MTR) which will perform actual measurement. All the control messages from CSV to MTS or MTR of each MS are sent to MTS or MTR via MTD of the MS. The reason that we designed MT system for all the control message messages between CSV and MTS or MTR to go via MTD is that we tried to make MTS and MTR be lightweighted processes that can run stably for a long time. MTS is forked by MTD when CS starts measurement. After MTS receives a measurement start message, it generates

measurement packets. The packets are generated in Poisson process by a pseudo-random number generator. MTS sends every packet to all the MTRs which are joining in the measurement. MTR is forked by MTD when CS starts measurement. After MTR receives a measurement start message, it opens Local DB file to be ready to receive measurement packets. Whenever it receives a measurement packet, it stores the record of the packet in Local DB. The record consists of 5 fields; (a) Sequence Number, (b) Sender IP Address, (c) Sent Time, (d) Receiver IP Address, and (e) Received Time. 'Sequence Number' is 4-byte sequence number field. 'Sender IP Address' is 4-byte IP address field of MT sender that sent the packet. 'Receiver IP Address' is also 4-byte IP address field of MT receiver that received the packet. 'Sent Time' is 8-byte timestamp field in which the timestamp is written by Ethernet device driver just before packet's being sent into network interface card. 'Received Time' is also 8-byte timestamp field where the timestamp is written by Ethernet device driver just after packet's being received from network interface card. DA is forked by MTD when CS gathers measurement data from each MS. After DA receives a gather start message, it opens Local DB and delivers the measurement data stored in it to SSV of CS.

Procedure of Measurement

Step 1. Initialization of MTD for measurement: CSV sends all the MTDs that take part in measurement a 'measure-ready' message indicating that they have to prepare a measurement. The control packet including the message provides them with a system parameter and a list of IP addresses of all the participating MTDs together with the message.

Step 2. Fork of measurement processes: When MTD of MS receives the 'measure-ready' message, it makes control channels that will be used to communicate with MTS and MTR that are implemented in UNIX domain stream socket. It forks MTS and MTR and then forwards the 'measure-ready' message to them through the control channels.

Step 3. Establishment of control channel: After MTS and MTR have been forked by MTD, they establish control channel that is used to communicate with MTD. MTS and MTR obtain the system parameter such as the list of IP addresses of participants from control packet including the 'mea-sure-ready' message. When MTS and MTR are ready to mea-sure, they report the readiness to MTD through the control channel.

Step 4. Confirmation about readiness from MTD: When MTD receives the report from both MTS and MTR, MTD sends CSV a 'measure-ready-ack' message indicating that MS is ready to measure.

Step 5. Start of measurement: When CSV has received the report from all MTDs, CSV sends them a 'measure-start' message indicating that they have to start measurement.

Step 6. Start of actual measurement: When MTD receives the 'measure-start' message, it forwards the message to its child processes: MTS and MTR.

Step 7. Injection of measurement packets: MTS generates measurement packets in Poisson process. The packets are sent to all participating MTRs except MTR in the same host through UDP socket.

Step 8. Storing of measurement records: When MTR receives a measurement packet, it stores into Local DB a record that consists of the following fields; (a) Sequence Number, (b) Sender IP Address, (c) Sent Time, (d) Receiver IP Address, and (e) Received Time.

VI. Conclusion

The next steps of the development of the MT are the modeling of its primary functions and then – their implementation. The QoS monitoring software will be written in C++ language and developed with Microsoft Visual Studio 6.0. The measurements can be taken using the TCP, UDP or ICMP protocol. The expected results should be used as main part of the work according to the project mentioned below and for provisioning application-specific QoS in NGN as to [15] as well.

Acknowledgement

This work is made in connection to the **Project BY-TH 105/2005**.

References

[1] ITU-T, "Support of ip-based services using ip transfer capabilities," Tech. Rep. Rec. Y.1241, 2001.

[2] S. Blake, D. Black, M. Carlson, E. Divies, Z. Wang, W. Weiss, "An architecture for differentiated services," IETF RFC 2475, 1998.

[3] C. Molina-Jimenez, S. Shrivastava, J. Crowcroft, and P. Gevros, "On the monitoring of contractual service level agreements," in Proceedings of the (WEC'04), April, 2004.

[4] K. Claffy, G. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," ACM SIGCOMM Computer Communication Review, vol. 23, no. 4, pp. 194–203, 1993.

[5] K. Claffy,"Internet measurement and data analysis: topology, workload, performance and routing statistics", NAE'99, 1999.

[6] A. Pasztor, "Accurate active measurement in the internet and its application," Ph.D. thesis, University of Melbourne, 2003.

[7] J. Mahdavi and V. Paxson, "Ippm metrics for measuring connectivity," IETF RFC 2678, September, 1999.

[8] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for ippm," IETF RFC 2679, 1999.

[9] G. Almes, "A one-way packet loss metric for ippm," IETF RFC 2680, 1999.

[10] G. Almes, S. Kalidindi, and M. Zakauskas, "A round-trip delay metric for ippm," IETF RFC 2681, 1999.

[11] R. Koodli and R. Ravikanth, "One-way loss pattern sample metrics," IETF RFC 3357, 2002.

[12] C. Demichelis and P. Chimento, "Ip packet delay variation metric for ip performance metric (ippm)," IETF RFC 3393, 2002.

[13] Les Cottrell – SLAC, Network measurement, Lecture # 4 presented at the 26th International Nathiagali Summer College on Physics and Contemporary Needs, 25th June – 14th July, Nathiagali, Pakistan, 2005

[14] E. Hernandez, M. Chidester, and A. George, "Adaptive Sampling for NetworkManagement," Journal of Network and Systems Management, vol. 9, no. 4, 2001.

[15] Atanasov I., E. Pencheva, An Approach to Provide Network Capabilities-based Added Value, Information Technologies and Control, 3/2007, pp.27-33