University of Zurich
Department of Informatics

Hasan
Peter Racz
Burkhard Stiller

SLO Auditing for Hosted Streaming Services

TECHNICAL REPORT – No. ifi-2008.08

July 2008

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Hasan, P. Racz, B. Stiller: SLO Auditing for Hosted Streaming Services
Technical Report No. ifi-2008.08, July 2008
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch

SLO Auditing for Hosted Streaming Services

Hasan, Peter Racz, Burkhard Stiller
University of Zurich, CSG@IFI, Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
[hasan|racz|stiller]@ifi.uzh.ch

ABSTRACT

Auditing of service level objectives (SLOs) committed by a service provider is necessary, since customers are to be reimbursed, if the provider fails to fulfil these commitments. By automating the auditing process, a timely detection of a violation is possible and potentially achieves economic gain. Thus, an SLO auditing system must be flexible in order to adapt to potential SLO changes, scalable in the processing time with respect to the amount of data, and supports multi-domain application environments. This paper presents a scenario for hosted streaming services and defines in detail relevant SLOs, which serve as examples for the visualization of steps to be undertaken to automate SLO auditing. Driven by the scenario and requirements specification a respective architecture is designed and prototypically implemented based on a generic auditing framework, which comprises out of distributed components for metering, auditing, and reimbursement. Additionally, a new reimbursement scheme is proposed that considers the degree and duration of SLO violations in calculating reimbursements. Finally, the evaluation shows that the required properties of the framework and the implemented auditing application for a bandwidth SLO are fulfilled.

***Keywords*: Service Level Agreement (SLA), Service Level Objective (SLO), SLA Compliance Auditing, Automation.**

I INTRODUCTION AND MOTIVATION

INTERNET services are used by many companies to operate their business and in doing this, they need to rely on various services offered by connectivity or network providers and other service providers. A company as a customer of a service provider may itself offer Internet services, thus also acts as a service provider. Furthermore, a service composition or an orchestration of various Internet services is made possible through the emerging Service Oriented Architecture (SOA) [7]. Thus, customers have to rely on services and their quality level offered by providers, which leads to the necessity to have a Service Level Agreement (SLA) concluded between a customer and a provider.

According to the TeleManagement Forum, an SLA is "a formal negotiated agreement between two parties, sometimes called a Service Level Guarantee, it is a contract (or part of one) that exists between the service provider and the customer, designed to create a common understanding about services, priorities, responsibilities, etc." [12]. A more detailed list of the content of an SLA is given in [14], which states that an SLA comprises in particular a service description, the expected performance level of the service, the procedure for reporting problems, the time-frame for response and problem resolution, the process for monitoring and reporting the service level, the consequences for the provider not meeting its obligations, and escape clauses and constraints.

However, besides these technical definitions an SLA has no value, if no examination is ever made to verify whether the provider meets its obligations, or if no consequences are applied, when contract violation may have happened. The performance level of a service committed is specified in a set of Service Level Objectives (SLO). Thus, SLA compliance auditing aims at verifying that these SLOs are met in a given situation. Due to the fact that hundreds of customers may use dozens of services in a very short period of time, this task must be automated in order to be effective, to be efficient, to reduce errors caused by human auditors, and to allow for timely reactions in case of an SLA violation.

The remainder of the paper is organized as follows. Section II presents an application scenario, which allows for Section III to derive requirements of an automated SLA compliance auditing infrastructure. While Section IV discusses the design of the architecture developed and its interfaces, Section V introduces the AURIC (***Au***diting F***r***amework for ***I***nternet Servi***c***es) framework and outlines a prototypical implementa-tion on top of this framework. An analytical evaluation of automated SLA compliance auditing is presented in Section VI with respect to requirements specified in Section III. Finally, Section VII concludes the work.

II APPLICATION SCENARIO

The application scenario selected shows the necessity of SLAs, a careful choice of respective SLA parameters and their definitions, precise specifications of SLOs, and appropriate reimbursements, in case of violations to those SLOs specified. Those SLOs are used to describe the process of an SLA compliance auditing and to demonstrate the ease of developing an SLA compliance auditor on top of the AURIC framework.

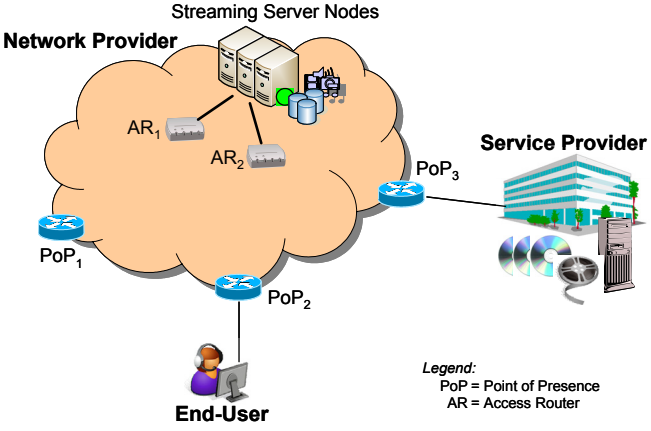


Fig. 1. Hosted Streaming Services

In this application scenario, the Service Provider (SP) offers streaming services, *e.g.*, Live TV and Video-on-Demand over the Internet. To reduce expenditures on infrastructure, its management and maintenance, the SP has its streaming server (software) hosted by a Network Provider (NP) and concludes an SLA with it, as depicted in Fig. 1. The streaming server runs on nodes provided and owned by the NP. In order to distinguish the streaming server from a node running it, the node is called a streaming server node.

A. SLA Parameters

In this case the SLA contains the following five SLA parameters, which are of key importance to the SP for offering high quality streaming services:

- Availability
- Downlink and uplink latency
- Downlink and uplink bandwidth

Availability is defined as the total time a streaming server node is reachable within a billing period from any node connected directly to any Point of Presence (PoP) of the NP. This SLA parameter covers the availability of PoPs, streaming server nodes, and network paths in between, but does not cover the availability of a streaming server, since the NP is only able to guarantee the availability of components under its control. Nevertheless, the availability SLO specified by the NP is very important, because it influences the overall availability of a streaming service of the SP.

The downlink latency is defined as the one-way delay time for transferring an IP packet received by an access router of any streaming server node through a NP's network to a node connected directly to any PoP. The value of this SLA parameter reflects the load of PoPs, access routers of streaming server nodes, and network paths in between. It affects the response time of a streaming service to a certain extent – depending on the current network load – as experienced by a user. However, since streaming servers are not owned by the NP, it cannot guarantee the latency, which covers the delay time introduced by a streaming server. In general, users of a streaming service are not willing to wait too long before being able to watch a video. Therefore, it is under the responsibility of the SP to develop and deploy a streaming service, which is scalable and supports load balancing among streaming servers, if it wants to offer a service with fast response time.

The uplink latency is defined as the one-way delay time for transferring an IP packet received by any PoP to any streaming server node. Uplink latency affects the time required to send a message and, therefore, it influences the overall response time of a streaming service.

The bandwidth parameter is defined as the number of bytes transferred between streaming servers and PoPs within a predefined time interval *T*. The downlink bandwidth is distinguished from the uplink bandwidth, since a streaming service has different requirements with respect to these two directions. More downlink bandwidth will be needed compared to uplink bandwidth. In order to be able to serve N users simultaneously with a unicast streaming rate of r kbps per user, a system must provide for a downlink bandwidth of at least N*r kbps. Therefore, this SLA parameter limits the number of concurrent users or the streaming rate to a user[1].

B. Service Level Objectives

Based on these definitions and assuming for the example a monthly billing period, the following five SLOs are specified for this application scenario, since they contain measurable parameters and determine the quality of the streaming service:

1. *Availability SLO*: Within a calendar month, any streaming server node S may only be unreachable from any node N connected to any PoP for a total duration of at most D. Availability is to be measured periodically and actively using probe packets. A streaming server node S is considered unreachable within a test cycle T_i, only, if no response is received for any probe packet sent by N to S (100% packet loss) during T_i. Thus, the existence of a single response is interpreted as availability within T_i. Therefore, the length of a test cycle needs to be restricted, *e.g.*, to 15 minutes.
2. *Downlink latency SLO (DL)*: The average value of downlink latency in each test may not exceed DL_{max} and the monthly average downlink latency may not exceed $\overline{DL}_{max}$. Note that, it only makes sense, if $\overline{DL}_{max}$ is smaller than DL_{max}. The average latency is to be measured actively and periodically every t minutes (one test cycle) and there are at least b probes to be sent in each test cycle.
3. *Uplink latency SLO (UL)*: The average value of uplink latency in each test may not exceed UL_{max} and the monthly average uplink latency may not exceed $\overline{UL}_{max}$. Note that, it only makes sense, if $\overline{UL}_{max}$ is smaller than UL_{max}. The average latency is to be measured actively and periodically every t minutes (one test cycle) and there are at least b probes to be sent in each test cycle.
4. *Downlink bandwidth SLO*: The reserved downlink capacity (DC) for SP's streaming services on an Access Router AR_j, to which streaming server nodes are connected, is $DC_{in}(AR_j)$. The total downlink data rate (DR) of packets received by AR_j from all streaming servers connected to it, $DR_{in}(AR_j)$, is therefore at most $DC_{in}(AR_j)$, which is ensured by NP through traffic shaping. The total data rate of packets leaving NP's network through PoP_k is $DR_{out}(PoP_k)$. Both $DR_{in}(AR_j)$ and $DR_{out}(PoP_k)$ are calculated periodically by measuring the number of bytes transferred over a time interval T. The downlink bandwidth SLO defines in general the minimal ratio e_d between the outgoing and the incoming data rate. Depending on where the incoming and outgoing

1. By employing P2P technology, the SP can serve more users, since a user does not have to get the stream from a server, but from other users watching the same stream. Another technology to reduce bandwidth requirements is multicasting, either on IP or application level. However, P2P mechanisms and application level multicast require resources of participating users, so the setting of service prices, if services are not free of charge, has to take this into account. With respect to IP level multicast, not many network providers support it. In any case, bandwidth is an important SLA parameter, since it is generally a limited resource of a network provider, therefore, a guarantee is required.

data rates are measured, the SLO can specify the following four different levels of granularity for the definition of the downlink bandwidth:

- *Aggregated bandwidth*: The ratio between the sum of outgoing data rates at all PoPs and the sum of incoming data rates at all ARs must exceed the threshold e_d. This condition is formulated in Eqn. (1).

$$\frac{\sum_k DR_{out}(PoP_k)}{\sum_j DR_{in}(AR_j)} > e_d \quad (1)$$

  This case defines a smaller granularity and means that the bandwidth guarantee is given for aggregated data rates entering and leaving the network regardless of the entry and exit points.

- *Bandwidth per PoP*: The ratio between the outgoing data rate at a given PoP and the sum of incoming data rates at all ARs that are destined to that PoP must exceed the threshold e_d. This condition is formulated in Eqn. (2).

$$(\forall PoP_k) \quad \frac{DR_{out}(PoP_k)}{\sum_j DR_{in}(AR_j, \text{to } PoP_k)} > e_d \quad (2)$$

  This case defines a more granular SLO, since the bandwidth guarantee is given per PoP for all incoming traffic destined to the PoP.

- *Bandwidth per AR*: The ratio between the sum of outgoing data rates at all PoPs that are originating from a given AR and the incoming data rate at that AR must exceed the threshold e_d. This condition is formulated in Eqn. (3).

$$(\forall AR_j) \quad \frac{\sum_k DR_{out}(PoP_k, \text{from } AR_j)}{DR_{in}(AR_j)} > e_d \quad (3)$$

  This case defines an SLO granularity similar to the previous one. The bandwidth guarantee is given per AR for all outgoing traffic originating from the AR.

- *Bandwidth per AR-PoP pair*: The ratio between the outgoing data rate at a given PoP that is originating from a given AR and the incoming data rate at the given AR that is destined to the given PoP must exceed the threshold e_d. This condition is formulated in Eqn. (4).

$$(\forall AR_j, \forall PoP_k) \quad \frac{DR_{out}(PoP_k, \text{from } AR_j)}{DR_{in}(AR_j, \text{to } PoP_k)} > e_d \quad (4)$$

  This case defines the best possible fine-granular SLO, since the bandwidth guarantee is given for each AR-PoP pair.

5. *Uplink bandwidth SLO*: The reserved uplink capacity (UC) on PoP_k for streaming services of the SP is $UC_{in}(PoP_k)$. The uplink data rate (UR) of packets received by PoP_k, $UR_{in}(PoP_k)$, is therefore at most $UC_{in}(PoP_k)$, which is ensured by the NP through traffic shaping. The total data rate of packets forwarded by AR_j to streaming server nodes is $UR_{out}(AR_j)$. Similar to the downlink bandwidth measurement both $UR_{in}(PoP_k)$ and $UR_{out}(AR_j)$ are calculated periodically by measuring the number of bytes transferred over a time interval T. Analog to the downlink bandwidth SLO different granularity levels can be specified, where the uplink bandwidth SLO defines in general the minimal ratio e_d between the outgoing and the incoming data rate.

Bandwidth measurements assume that the time is synchronized in metering components at ARs and PoPs. Additionally, the measurement interval T has to be selected in a way that the maximal latency L can be neglected in the measurement. The relative error caused by the latency is L/T. Thus, assuming a maximal latency of 100 ms and a measurement interval greater than 100 s, the relative error gets smaller than 0.1%. Therefore, in practical settings, where T will usually be selected in the range of several minutes, the latency can be neglected.

It is important to note for this example, that even though the NP is able to fulfil all of these SLOs, there is no guarantee of users' Quality-of-Experience (QoE). QoE of a user depends additionally on various other parameters, such as content quality, performance of streaming servers, performance of networks between the user node and a PoP, and performance of the user node and the streaming client, which the NP has no control of. The SP and users can conclude an SLA to ensure content quality and performance of streaming servers, but not the remaining parameters. They have to be taken care by users themselves.

C. SLO Violations and Reimbursements

If the NP fails to fulfil an SLO, it will, most likely, reimburse the SP a certain percentage of the total monthly charges of those streaming server nodes affected. To the best knowledge of the authors at the time of writing, current practices in industry for calculating reimbursements [1] are either based on a fix percentage of monthly charges or on the length of the time required by the provider to restore its service performance. In the latter case, customers are required to issue a trouble ticket, if they detect any problem, when consuming a service. This scheme is neither fair nor precise, since it may take some time before customers realize that there is a problem to report it. Furthermore, the use of a fixed percentage ignores degrees or durations of SLO violations. Thus, a much better scheme, resulting out of the work performed, is presented in Section IV, which considers both the degree and the duration of SLO violations.

III Requirements

The above mentioned scenario provides for a basis to derive key requirements for a technically and economically feasible SLA compliance auditing infrastructure. These requirements comprise major aspects, such as multi-domains support, load scalability, flexibility, and economic gain.

A. Multi-domains Support

In an SLA compliance auditing process, measurement data on service performance are examined to detect any violation to SLOs specified. Performance measurements, auditing, and violation handling must not necessarily be accomplished by a single administrative domain. Furthermore, it does not have to be the customer or the provider, who does performance measurements and auditing, but it can also be a Trusted Third Party (TTP), *e.g.*, if a lack of trust between customers and providers is an issue or if lower costs can be achieved through a TTP. Therefore, inter-domain interactions are required, if those functions are distributed across administrative domains.

Communications across administrative domains generally happen through the Internet and, thus, should involve security mechanisms to protect information and infrastructure from attacks. Moreover, communications via Internet are assumed to be less reliable than within an Intranet. Therefore, a multi-domains support implies requirements to cope with security and reliability.

B. Load Scalability

The resource consumption of an SLA compliance auditing process depends on the amount of data to be examined. Resources needed are processing time and memory, and they must be at least linearly scalable, if the data amount changes.

Various factors may influence the amount of performance data, amongst others, the number of SLAs, services, SLOs, and sessions. They need to be analyzed, in particular to know the relation between these factors and the growth of performance data. A linear relation is here also the worst case acceptable.

C. Flexibility

With respect to service performance, customer or application requirements change over time. This is due to the desire to have a better quality of multi-media content, and an ever faster transfer of information. Fortunately, the capability and capacity of networking infrastructure also have a rapid growth in order to fulfil those demands. This trend leads to changes in SLOs. Coping with SLO updates becomes an important task in the change management. Therefore, changes to committed values must be possible, *i.e.*, the system must be configurable. Additionally, to avoid building an auditing application from scratch for each new SLO and situation, a flexible auditing framework is necessary, based on which the necessary auditing applications can be easily derived.

D. Discussion and Economic Gain

Since new Internet services are created, providers may want to offer them over time, which require the definition of new SLA parameters and a respective SLO specifications. This leads to the need to develop an applicable SLA compliance auditing application, which suits all requirements stated above.

Operating such an automated SLA compliance auditing requires expenditures to be spent on the infrastructure and human resources. The economic gain that is achievable through this automation must be greater than those efforts to setup and operate the automated auditing infrastructure. Therefore, the AURIC approach developed addresses this optimization balance.

IV Architecture Design

The SLA compliance auditing architecture developed (cf. Fig. 2) is designed to follow those requirements stated above.

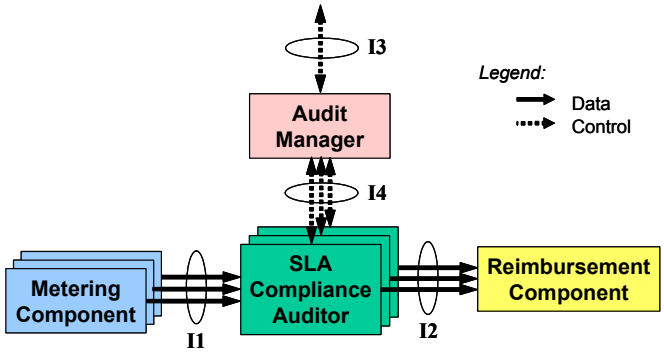


Fig. 2. SLA Compliance Auditing Architecture

A. Components

The architecture comprises out of four components (cf. below), which cover all tasks defined in an SLA with respect to its compliance auditing. Those components may be distributed within and also across an administrative domain, in case there is more than one party involved. Replication as within Fig. 2 is done for two reasons: (a) to denote that processing load is to be shared among instances of the same component; in this case, those instances share the same logic, but they process different data sets, and (b) a different SLO normally requires a different processing logic. Thus, instances of a component might not always share the same logic, although they have the same function. For example, the function of all instances of a metering component is performance measurement, but one instance is, *e.g.*, responsible for bandwidth measurements, whereas another one for delay measurements.

1) Metering Component

In general, each instance of a metering component is responsible for the measurement of a specific SLA parameter. However, metering tools do exist, which are capable of measuring some SLA parameters at the same time using the same test, such as ICMP (Internet Control Message Protocol) ping. It can be used to measure the round trip delay, jitter, and packet loss rate (thus, also availability). To measure the availability specified in the scenario using ICMP ping, each PoP is connected directly to a metering node, which pings all streaming server nodes periodically. In order to allow 100% packet loss in n test cycles, if the total duration of unavailability must be less than D, then the test cycle length must not exceed D/n.

This ICMP ping-based measurement is used mainly due to its easy deployment. For example, Verizon Business measures its network performance using data collected by ICMP pings. In 5-minute intervals, data are collected from designated routers in key network hubs world-wide. All samples from the previous month are used to calculate the monthly average round trip delay time and packet delivery statistics [13].

In order to obtain timely performance data of real-time applications, AT&T develops and deploys an active measurement method [3] using two test sequences: a Poisson

and a periodic probe sequence, according to RFC 2330 [8] and RFC 3432 [9]. AT&T divides each 24-hour day into 96 test cycles of 15 minutes. Between two measurement servers of any pair, a periodic probe sequence of UDP (User Datagram Protocol) packets is sent by each of the servers independently in each test cycle. The probe sequence has a random start time within the 15 minute cycle and lasts for 1 minute. The UDP packet size is 60 Byte and the inter-packet interval is 20 ms. The periodic probe sequence mimics a real-time VoIP application. However, the dense test of the periodic probe sequence covers only a small fraction of the test cycle. To increase the probability of detecting possible performance degradation the Poisson probe sequence is designed, which is run throughout the length of a test cycle, but at lower density.

The performance metrics aimed at by AT&T are round trip delay, round trip loss ratio, delay variation, and the extent of packet reordering. Therefore, to measure one-way delay specified in the scenario's SLOs, the AT&T measurement method needs to utilize a local high-accuracy time synchronization system and needs to be adapted accordingly. Furthermore, an experimental comparison of different Network Time Protocol (NTP) synchronization strategies to measure one-way delay is presented in [10].

To measure the one-way delay between an AR and a PoP, two metering nodes are needed: one connected to the AR and the other to the PoP. One node sends timestamped probes, while the receiving node generates measurement records. In case of downlink latency, measurement records are generated by the node connected to the PoP. They include the identifier of the PoP, a timestamp, and the measured *DL*. For uplink latency, measurement records are generated by the node connected to the AR and these contain the identifier of the AR, a timestamp, and the measured *UL*.

To measure the downlink and uplink bandwidth, metering components at ARs and PoPs periodically count the amount of data (measured in byte) that traverse the node and originate from or are addressed to a streaming server node. Depending on the granularity level of the SLO, a metering component has to meter the traffic at different granularity levels and can perform an aggregation at different levels. For the *aggregated bandwidth* SLO, a metering component has to meter the traffic originating from one of the streaming server nodes of the SP. Thus, the metering component has to differentiate packets based on the source address and count the amount of data transferred in packets that have the address of any of the streaming server nodes as the source address. The metering component sends periodically measurement records to an auditor. Measurement records from an AR contain the identifier of the AR, a timestamp, and the measured DR_{in} and UR_{out}, while measurement records from a PoP contain the identifier of the PoP, a timestamp, and the measured UR_{in} and DR_{out}.

For the other three SLO types (cf. Section II.B), a more granular metering is required. The metering component has to meter the traffic per source and destination address pairs (or at least at a granularity level that enables the differentiation of traffic from each AR and PoP). Measurement records are in this case per source and destination address pair and records from an AR contain the identifier of the AR, a timestamp, the source and destination addresses, and the measured DR_{in} and UR_{out}, while measurement records from a PoP contain the identifier of the PoP, a timestamp, the source and destination addresses, and the measured UR_{in} and DR_{out}.

The clocks of metering components both at ARs and PoPs have to be synchronized, which can be done by the NTP. NTP achieves accuracy in the order of tens of milliseconds, depending on the latency to the time server [4]. This accuracy is sufficient for bandwidth measurements, since in case of a clock skew of S seconds the relative error is S/T, where T is the measurement interval, which is typically in the range of several minutes.

2) SLA Compliance Auditor

The SLA compliance auditor (auditor in short) implements the logic to audit performance measurement records according to an SLO. It retrieves[2] measurement records related to a specific SLO from metering components, audits them, and sends violation reports, if any, to a reimbursement component. While an SLO represents a *target* (reference) performance of a system under examination, measurement records represent facts about its *actual* performance. Each test cycle generates a measurement record, and normally, a set of measurement records is needed to determine a possible SLO violation.

A Fact-List is defined as a set of measurement records in chronological order. A Fact-List is *complete,* if it allows for determining whether an SLO is violated or not. Otherwise, it is called an *open* Fact-List. An audit for an SLO can be seen as a function, mapping complete Fact-Lists to values representing degrees of compliance with (or violation to) the SLO. In an auditing process, a degree of violation c is calculated from a set of property values P_1, P_2, ..., P_n, as shown in Eqn. (5). Properties are those parameters describing an SLO. Their values have to be obtained from a complete Fact-List. For example, DR_{in} and DR_{out} are properties in a downlink bandwidth SLO. The function f_{audit} defines an algorithm to calculate a compliance value, while each function p_i defines an algorithm to calculate a specific property value.

$$c = f_{audit}(P_1, P_2, \ldots, P_n) \qquad c \in \Re \tag{5}$$
$$P_i = p_i(Factlist) \qquad 1 \le i \le n$$

Therefore, an audit task is decomposed into the following *sequence* of subtasks:

1. *Fact Filtering*: Selection of measurement records related to an SLO,
2. *Fact Grouping*: Collection of measurement records and generation of complete Fact-Lists,
3. *Property Values Calculation*: Calculation of property values from a complete Fact-List,
4. *Compliance Value Calculation*: Calculation of the degree of compliance with the SLO from property values,

2. In general, a metering component is run to obtain performance data continuously, while an auditor is run on demand and has a limited lifetime. Therefore, auditors are configured to retrieve measurement records from metering components.

5. *Violation Report Compilation*: Compilation of parameters for a violation report from available information, *i.e.*, compliance value, property values, and other information in a complete Fact-List.

Measurement records can be audited in real-time or in batch mode. The benefit of real-time auditing is a timely detection of violations and, thus, allowing for a fast reaction. However, in many cases, the real-time requirement is not hard, since the rate, at which measurement records are generated, is normally low. Furthermore, a complete Fact-List is needed before a violation can be determined.

3) Audit Manager

An audit manager controls a set of auditors. It is responsible for instantiating and configuring each auditor to conduct an audit task. A real-time auditor runs continuously as long as the SLO it is responsible for is still valid. In case of the batch processing mode, audit tasks are accomplished at the end of a billing period. In this regard, an auditor is terminated as soon as it has completed its task and has notified the audit manager of the task completion.

An audit manager is also responsible for collecting statistics' information on audit tasks, which includes the load of an auditor, the time it has spent, and information on SLO violations detected. Finally, to manage auditors appropriately, an audit manager needs to know the status of each auditor, *e.g.*, whether an auditor is idle or waiting for new measurement records, or not responding at all.

4) Reimbursement Component

The reimbursement component has to calculate reimbursements to be paid to customers as a consequence of a provider not meeting its SLOs. Calculations of reimbursements are based on SLO violation information obtained from auditors, and reimbursements are one type of inputs to a charge calculation process.

In principle, the amount of a reimbursement can be defined as a function of the degree and the duration of an SLO violation. Suppose that $R_{dgr}(v, x)$ is a function, which maps degrees of violations, x, of an SLO identified by the index v, to reimbursements in a percentage of monthly charges, assuming a single SLO and a single violation in the billing period. Suppose also that $R_{dur}(v, x)$ is defined similarly with respect to possible durations of a violation. Hence, by specifying various weighting factors, the total reimbursement, R, can be calculated for all violations of all SLOs, as given in Eqn. (6), where

v = index of an SLO
SLO_v = SLO number v
n_{SLO} = number of SLOs
u = index of an SLO violation
n_v = number of violations of SLO_v
$dgr_{u,v}$ = the degree of violation number u of SLO_v
$dur_{u,v}$ = the duration of violation number u of SLO_v
w_v = the weight of SLO_v
p_v = the weight of the function R_{dgr} for SLO_v
q_v = the weight of the function R_{dur} for SLO_v

$$R = \sum_{v=1}^{n_{SLO}} w_v \cdot \sum_{u=1}^{n_v} \frac{p_v \cdot R_{dgr}(v, dgr_{u,v}) + q_v \cdot R_{dur}(v, dur_{u,v})}{n_v} \quad (6)$$

$$\sum_{v=1}^{n_{SLO}} w_v = 1 \qquad 0 \le w_v \le 1$$

$$\forall v = 1 \ldots n_{SLO} \qquad p_v + q_v = 1 \qquad 0 \le p_v, q_v \le 1$$

The advantages of Eqn. (6) lie both in its simplicity and flexibility. Providers and customers need to define or agree on $R_{dgr}()$, $R_{dur}()$, and a set of weighting factors. In relation to a billing period, two types of SLO violations are distinguished:

1. Type 1: Detectable within a billing period
2. Type 2: Detectable only at the end of a billing period

A reimbursement scheme based on the degree of an SLO violation can be applied to both types of violations. However, a reimbursement scheme based on the duration of an SLO violation can only be applied to type 1. For example, violations to the availability SLO, as defined in this scenario, are of type 2. This is due to the fact, that in principle, an availability SLO violation can be determined only at the end of a calendar month. Thus, a reimbursement for an availability SLO violation is defined as a function of its degree, which is given by the duration of the total downtime. As an example, Table I specifies reimbursements for each unreachable streaming server node as a function of the total downtime in a calendar month.

TABLE I
Example of Reimbursement for each Unreachable Streaming Server Node

Downtime [hour]	Reimbursement [%][a]
0.5	5
1	10
2	20
4	30
8	40
16	50
24	80
> 24	100

a.Reimbursement is given in percentage of the monthly charges of a streaming server node.

In case of bandwidth SLOs, both the duration and the degree of a violation are to be determined, if one is detected in a test. The duration of a violation is the length of the test. The duration of two or more consecutive violations can be added together to count as a single violation of longer duration. The degree of a violation is either the maximum or the average deviation of the difference between incoming and outgoing rate from the committed threshold.

For the sake of simplicity in calculating reimbursements, a downlink or uplink latency SLO can be treated as two SLOs: one for the average latency in each test cycle and one for the monthly average latency. Hence, a reimbursement calculation for violations to average latency in each test cycle uses a scheme similar to a bandwidth SLO, whereas a reimbursement

calculation for violations to monthly average latency uses a scheme similar to an availability SLO.

B. Interfaces

To enable an implementation of the architecture as depicted in Fig. 2, suitable technologies are considered for all interfaces.

1) Interface I1

This interface is used to transfer measurement records from a metering component to an auditor. Since the same measurement record may be needed by different auditors, it is not deleted after its transfer to an auditor, but is kept accessible during the active billing period. In order to enable an auditor to select measurement records from a metering component, a message exchange must be defined, which allows for specifying a selection criterion. A request-answer communication pattern is used to transfer a selection criterion as well as measurement records between an auditor (A) and a metering component (M). A sends a `SelectionRequest` message to a M containing a criterion to select certain measurement records. As a response, M sends a `SelectionAnswer` message back with a result code indicating whether there was something wrong in processing the request message:

```
A => M: SelectionRequest(<SelectionCriterion>)
M => A: SelectionAnswer(<ResultCode>)
```

To transfer measurement records the following message pair is used:

```
M => A: TransferRequest(<Record> {, <Record>})
A => M: TransferAnswer(<ResultCode>)
```

To achieve this type of interaction pattern, the Diameter protocol [2] is very suitable to implement this interface, since information in a measurement record can be stored in attribute value pairs (AVP) and Diameter is applicable to inter-domain communications. The Diameter Base Accounting message pair, *i.e.*, `Accounting-Request/Accounting-Answer` (ACR/ACA), is sufficient to implement the Transfer message pair. However, to allow for the use of the Selection message pair, a new Diameter command must be defined additionally.

2) Interface I2

Each violation report as a result of an auditing process is transferred to a reimbursement component through this interface. Basically, violation reports and measurement records share the same structure. Thus, the Transfer message pair is applicable as well for the transfer of violation reports between an auditor (A) and a reimbursement component (R):

```
A => R: TransferRequest(<Report> {, <Report>})
R => A: TransferAnswer(<ResultCode>)
```

Therefore, the Diameter protocol is also suitable to implement this interface.

3) Interface I3

This interface allows the configuration and management of audit tasks. An audit task configuration specifies the meters from where measurement records are to be retrieved, the SLO to be audited, and the reimbursement component that should receive the audit results. This interface also defines message exchanges to setup and terminate an audit task, and to request statistics and status information. Processing audit tasks is the responsibility of the audit manager and therefore, this interface can be seen as an interface for offering an auditing service by the audit manager. The following message exchange between a service requestor (SR) and the audit manager (AM) is used to configure an audit task:

```
SR => AM: AuditRequest(<TaskConf>)
AM => SR: AuditAnswer(<TaskID>, <ResultCode>)
```

The parameter `TaskConf` contains addresses of the metering components, the identifier of the SLO, the address of the reimbursement component, and optionally the start time of the audit. `TaskID` is used to identify the audit task configured. An audit task being conducted can be terminated as follows:

```
SR => AM: TerminationRequest(<TaskID>)
AM => SR: TerminationAnswer(<TaskID>, <ResultCode>)
```

To request the status of an audit task, the following message pair is used. Possible status codes are *e.g.*, `Scheduled`, `InProgress`, `Completed`, and `WaitingForData`.

```
SR => AM: StatusRequest(<TaskID>)
AM => SR: StatusAnswer(<TaskID>, <StatusCode>,
                       <ResultCode>)
```

Statistics' information as mentioned in Section IV.A.3 can be queried by using the following message pair:

```
SR => AM: StatisticsRequest([<TaskID>])
AM => SR: StatisticsAnswer(<StatisticsInfo>,
                       <ResultCode>)
```

In addition to the four message types, a notification message can be sent at any time by an audit manager to the service requestor to inform about any error occurred during an audit or that an audit task has been completed:

```
AM => SR: Notification(<TaskID>,
             <NotificationCode>, [<Info>])
```

`NotificationCode` is used for error-free and erroneous situations. In case of erroneous situations, the parameter `Info` is used to give a more detail information, *e.g.*, if a metering component is unreachable, its address is given in `Info`.

This auditing service can be implemented as a stateful web service, which is due to a wide acceptance of web services for application-driven inter-domain interactions. This possibility enables the NP to outsource the auditing process and enables a third party to offer auditing as a service to any provider.

4) Interface I4

While I1, I2, and I3 are supposed to support inter-domain interactions, I4 is used for intra-domain communications. It allows the audit manager to control a set of auditors, which perform those audit tasks received by the audit manager through the interface I3. The audit manager delegates the task to audit a specific SLO to an auditor that implements the auditing logic of this SLO. Basically, this interface must map the set of messages defined for I3 to messages used for configuring and controlling an auditor.

The following messages are used to start an audit task:

```
AM => A: StartAuditRequest(<TaskConf>)
A => AM: StartAuditAnswer(<ResultCode>)
```

The following messages are used to stop an audit task:

```
AM => A: StopAuditRequest()
A => AM: StopAuditAnswer(<ResultCode>)
```

The following messages are used to request the status information of an audit task:

```
AM => A: StatusRequest()
A => AM: StatusAnswer(<StatusCode>, <ResultCode>)
```

The following messages are used to request the statistics about an audit task:

```
AM => A: StatisticsRequest()
A => AM: StatisticsAnswer(<StatisticsInfo>,
                          <ResultCode>)
```

The following message is used to notify the audit manager about state changes:

```
A => AM: Notification(<NotificationCode>, [<Info>])
```

C. Security Considerations

One of the consequences of supporting inter-domain interactions is the opening of the infrastructure to external accesses. Thus, the infrastructure with distributed components has to deal with the following threats:

- Data theft through eavesdropping or unauthorized access to measurement records and violation reports.
- Data interception and manipulation by Man-In-The-Middle attacks, which causes an auditor yielding wrong results.
- Denial-of-Service attacks to various components providing a service.

In order to protect an auditing infrastructure against the first two threats, accesses to data and a service must be controlled. In general, this can be achieved by employing an AAA infrastructure, *e.g.*, using the generic AAA approach [6]. However, the AAA infrastructure is designed to authenticate users as well as to authorize and account usage of services by users. In this respect, services are supposed to interact with a large number of users, which are joining and leaving. On the contrary, most auditing applications have a pre-defined and very small set of interacting components, which does not change during an audit. Additionally, an auditing infrastructure typically is set up for a longer time of operation. In this case, authentication and authorization are accomplished once between these components. Therefore, security associations among interacting components can be pre-established to secure communications during the auditing. Thus, on-line interactions with AAA entities are not needed.

To protect against a Denial-of-Service attack an Intrusion Protection System can be employed, which is able to identify possible attacks and to block the respective traffic.

D. Reliability Considerations

Since inter-domain communications in this approach happen through the Internet, latency and loss rates may be high. Hence, an auditor must consider missing and delayed delivery of measurement records. To cope with short-term data loss, a reliable transport protocol is required, if this cannot be solved on the application layer. Data loss over a longer period can be handled only on the application layer, where policies need to provide for decisions on how to proceed. A batch mode auditing may postpone the audit and restart it at a later time, whereas a real-time auditing may send an alarm and awaits further inputs. A similar consideration is needed for handling latency. Latency up to a certain value is tolerable, but latency above this threshold may be considered as data loss.

The impact of higher latency and loss rate to auditing results are different in real-time and batch mode auditing. Since a batch mode auditing can just be restarted with the same data again, auditing results are not affected. In real-time auditing, no result is obtained in time, where inputs are missing. Results can also be wrong, if the audit algorithm is not defined appropriately to handle delayed or missing inputs.

V Implementation

To study and show the feasibility of SLA compliance auditing a bandwidth SLO compliance auditor has been implemented prototypically on top of the auditing framework AURIC [5]. Fig. 3 depicts the respective software architecture developed. To audit a specific SLO, following modules have to be implemented: performance meter, auditing logic, and reimbursement calculator.

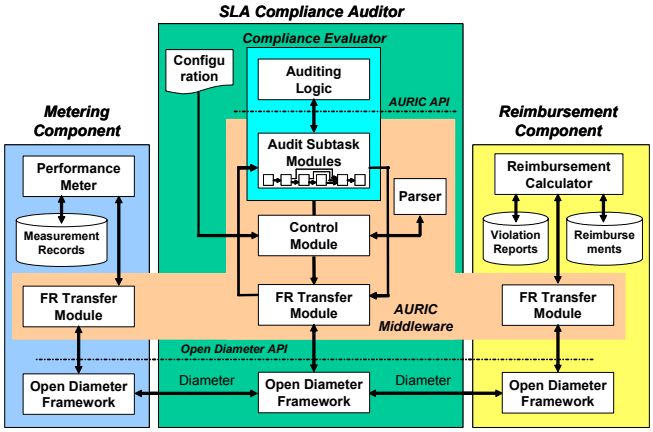


Fig. 3. Software Architecture

A. AURIC Overview

AURIC determines the auditing framework, which provides for a set of libraries to ease the development of an SLA compliance auditing application. The FR (Fact and Report) transfer module is responsible for the transfer of measurement records and violation reports, whereas audit subtask modules manage measurement records including Fact-Lists during auditing and carry out the sequence of audit subtasks. AURIC employs the Diameter Accounting protocol to transfer measurement records and violation reports. The AURIC implementation is written in C++ and it supports the use of the Open Diameter Framework.

AURIC defines the API for implementing the auditing logic in C++. The API provides five base classes (cf. Fig. 4) corresponding to those five audit subtasks described in Section IV. The parent class `SubtaskFunc` provides methods to parametrize the application specific subtask function derived. These methods are invoked by the auditing framework after the creation of the function based on the configuration file. Each base class offers a method called `Process()`, whose purpose is described in Table II, and which should be implemented by the developer of an auditing application. This method is invoked by the auditing framework each time there are data to be processed.

B. Bandwidth Usage Meter

Bandwidth usage meters are deployed at each AR and PoP: in the network of NP and they meter the traffic from and to the streaming server nodes of SP. The prototypical implementation

of the meter uses the libpcap library [11] to capture and filter packets traversing the network interface. The libpcap library supports a flexible capturing of packets based on the Berkeley Packet Filter (BPF) format. BPF enables a wide range of parameters for the definition of a filter, e.g., source and destination addresses, transport protocol, and port numbers. Additionally, it supports logical operators (i.e. and, or, and not) to build complex filter expressions. The prototypical meter enables the configuration of several filters and it counts the number of bytes for each packet that passes these filters. The meter maintains separate counters and filter expressions for the incoming and outgoing traffic.

```
class SubtaskFunc {
 public:
  virtual ~SubtaskFunc() {}
  virtual bool SetStringParm(unsigned int parmNo,
   const string& parmVal) {return false;}
  virtual bool SetNumberParm(unsigned int parmNo,
   float parmVal) {return false;}
  virtual bool SetBooleanParm(unsigned int parmNo,
   bool parmVal) {return false;}
};
class FilterFunction : public SubtaskFunc {
 public:
  virtual ~FilterFunction() {}
  virtual bool Process(const Fact& currentFact)=0;
};
class GroupingFunction : public SubtaskFunc {
 public:
  virtual ~GroupingFunction() {}
  virtual void Process(const Fact& currFact,
   OpenFactLists& ofl) = 0;
};
class PropertyFunction : public SubtaskFunc {
 public:
  virtual ~PropertyFunction() {}
  virtual prop_value_t* Process(
   FactList& currentFactList) = 0;
};
class ComplianceFunction: public SubtaskFunc {
 public:
  virtual ~ComplianceFunction() {}
  virtual float Process(
   const PropertyValues& propertyValues) = 0;
};
class AttributeFunction : public SubtaskFunc {
 public:
  virtual ~AttributeFunction() {}
  virtual void Process(string& attrValue,
   FactList& currentFactList,
   const PropertyValues& propertyValues,
   float complianceValue) = 0;
};
```

Fig. 4. AURIC API

The meter communicates with the FR transfer module via an API (see Fig. 3). The FR tranfer module retrieves measured data from the meter periodically in every measurement interval T, and transfers measurement records to the auditor via the Diameter protocol.

In case of the *aggregated bandwidth* SLO the configuration to measure DR_{in} at ARs and DR_{out} at PoPs is the following:

ip src host 192.168.1.100,

and to measure UR_{out} at ARs and UR_{in} at PoPs the following:

TABLE II
The Purpose of the API's `Process()` Methods

Class	The Purpose of `Process()` Method
`Filter-Function`	To examine the measurement record encapsulated in the `Fact` object and return true or false to denote whether the record is related to the SLO being audited.
	A `Fact` object provides for methods to get information about the measurement record encapsulated in the object, *e.g.*, the value of a particular attribute.
`Grouping-Function`	To examine the measurement record encapsulated in the `Fact` object and assign the record to one or more Fact-Lists with the help of `OpenFactLists` object.
	An `OpenFactLists` object provides for methods to manipulate open Fact-Lists managed by the auditing framework, *e.g.*, to add a Fact into an open Fact-List and to close an open Fact-List, *i.e.*, to declare it complete.
`Property-Function`	To calculate a property value from the list of related measurement records encapsulated in the `FactList` object. A `FactList` object provides for methods to manipulate and to access information about measurement records encapsulated in the object, *e.g.*, the number of records, the sum of the value of a particular field of the records.
`Compliance-Function`	To calculate a compliance value from the list of property values encapsulated in the `PropertyValues` object.
	A `PropertyValues` object provides for methods to access property values.
`Attribute-Function`	To calculate a report attribute (parameter) value from the list of related measurement records (encapsulated in `FactList` object), the list of property values (encapsulated in the `PropertyValues` object), and the compliance value.

ip dst host 192.168.1.100,

where it is assumed that the streaming server node has the IP address 192.168.1.100. If there are several streaming server nodes in place, additional filtering rules are to be configured with the IP address of each node. Additionally, depending on the SLO specification the configuration can define the transport protocol and ports as well in order to measure only TCP or UDP traffic and traffic to or from specific ports.

For the other three SLO types (see Sec. 2.B) the filter configuration has to be more fine granular, *e.g.*, in case of the *bandwidth per PoP* SLO the configuration to measure $DR_{in}(AR_j$ *to* $PoP_k)$ at AR_j is the following:

ip src host 192.168.1.100 and dst net 192.168.10.0/24,

where it is assumed that users attached to PoP_k receive an IP address from the address range 192.168.10.0/24. Analogously, a similar filter rule has to be configured for each AR and PoP.

C. Auditing Logic for Downlink Bandwidth SLO

The auditing logic of an application defines in detail those procedures applied to measurement records, which have to be achieved through the sequence of audit subtasks driven by the application scenario under consideration. In the Fact Filtering subtask the auditing framework calls the `Process()` method of a `FilterFunction` object defined by the application, if a measurement record is available. Since measurement records retrieved from a metering component are supposed to be already selected for this specific SLO, namely downlink bandwidth SLO, this method returns the boolean value true.

Suppose that an aggregated bandwidth SLO is used (cf. Section II), then in the Fact Grouping subtask the `Process()` method of a `GroupingFunction` object creates a Fact-List from all records with a timestamp difference of less than a certain threshold, in order to consider time synchronization error of those measurement points. A Fact-List is considered complete in this SLO, if a new measurement record arrives, whose timestamp differs from the timestamp of the previous record by a value greater than the threshold.

In the Property Values Calculation subtask two property values are calculated per Fact-List: $Total\_DR_{in}$ and $Total\_DR_{out}$. The value of $Total\_DR_{in}$ is the sum of DR_{in} of all measurement records in the Fact-List from all access routers, whereas the value of $Total\_DR_{out}$ is the sum of DR_{out} of all measurement records in the Fact-List from all PoPs. The `Process()` method of a `PropertyFunction` must be able to distinguish identifiers of an access router from those of a PoP.

In the Compliance Value Calculation subtask the `Process()` method of a `ComplianceFunction` object calculates and returns the ratio between $Total\_DR_{out}$ and $Total\_DR_{in}$ as the compliance value. Finally, in the Violation Report Compilation subtask each report attribute, *e.g.*, `Timestamp` or `ViolationDegree`, is determined or calculated by the `Process()` method of an `AttributeFunction` object. In case the compliance value is smaller than a pre-configured threshold, a violation report is generated and sent to the reimbursement component.

D. Reimbursement Calculator

Based on violation reports from auditors and the agreed reimbursement function, this module calculates reimbursements to be substracted from monthly charges. The reimbursement calculator is configured with the reimbursement functions $R_{dgr}(v, x)$ and $R_{dur}(v, x)$, and the wighting factors for each SLO. At the end of a billing period the reimbursement calculator reads all violation reports. It retrieves the violation degree and duration from the violation report and based on these values it calculates the reimbursement. Afterwards, it generates a reimbursement record for each violation, containing customer and provider details, the billing period, the amount of reimbursement, and references to the SLO and the violation report.

VI Evaluation

The evaluation of automated SLO auditing based on AURIC is undertaken with respect to all requirements specified in Section III.

A. Multi-domain Support

In the prototypical implementation, multi-domain support relies on the use of the Diameter protocol. This means that security and reliability of inter-domain communication depends to a great extent on the OpenDiameter implementation. The Diameter standard defines that the Diameter protocol must not be used without any security mechanism (TLS or IPsec). Furthermore, the support of SCTP by OpenDiameter provides for a reliable communication platform.

B. Load Scalability

In a load scalability evaluation, the amount of measurement records to be audited per time unit is crucial, since the processing rate of an auditor is limited. Suppose n_{AR} and n_{PoP} is the number of ARs and PoPs respectively, then Table III summarizes the amount of measurement records n_{rec} per test cycle T for various types of bandwidth SLOs. In case of latency SLO, n_{rec} equals $n_{AR} * n_{PoP}$ in each T. Since a server should not be down frequently, a metering component for availability SLO generates very few records in a normal operation, if only unavailability events are stored. In case all servers are down for a long period, n_{rec} equals $n_{Server} * n_{PoP}$ in each T.

TABLE III
Amount of Measurement Records for Bandwidth SLO

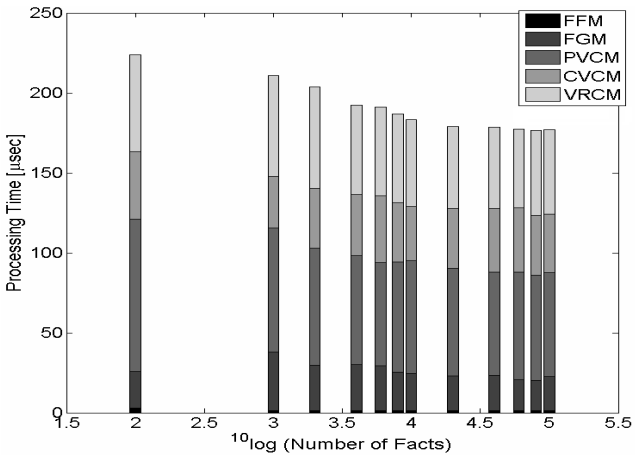
Type of Bandwidth SLO	Amount of Measurement Records per Test Cycle
Aggregated	$n_{AR} + n_{PoP}$
Per AR	$n_{AR} + n_{PoP} * n_{AR}$
Per PoP	$n_{PoP} + n_{PoP} * n_{AR}$
Per AR-PoP Pair	$2 * n_{AR} * n_{PoP}$

Assuming a test cycle length of 15 minutes and 100 streaming server nodes, 50 ARs, and 200 PoPs are in operation, the amount of measurement records generated is at most 20’000 every 15 minutes per SLO, which can be considered low. The implemented bandwidth SLO auditor is capable of processing measurement records generated at that rate as depicted in Fig. 5. The x-axis represents the number of measurement records (Facts) made available to an auditor *at once*, while the y-axis displays the average processing time per measurement record spent by each of the audit subtasks. As observed, the average processing rate is about 5’000 records per second or 4’500’000 in 15 minutes. The smaller the length of a test cycle, the higher the processing rate of an auditor is required. Choosing the proper length of a test cycle is a trade-off between accuracy and effort.

C. Flexibility

The flexibility of AURIC is evaluated by examining the way to accommodate the framework and the auditing application to a change in an SLO. Suppose that NP and SP agree to change the aggregated bandwidth SLO to bandwidth SLO per PoP, then measurement records generated by ARs must contain the destination address or the identifier of the PoP, through which a packet leaves the network. Thus, the implementation of a bandwidth usage meter must be extended to provide this level of granularity. Note that a change in the structure of a measurement record does not cause any changes to the implementation of the FR transfer module.

In order to adapt to the new level of SLO granularity, the implementation of the auditing logic needs to be changed. However, since this new SLO only changes the level of aggregation, only the `Process()` method of a subclass of the `GroupingFunction` class must be reimplemented. All other subclasses can remain the same. The change happens to the



FFM = Fact Filtering Module
FGM = Fact Grouping Module
PVCM = Property Values Calculation Module
CVCM = Compliance Value Calculation Module
VRCM = Violation Report Compilation Module

Fig. 5. Average Processing Time per Measurement Record (Fact)

way a Fact-List is created, namely all measurement records in a Fact-List must have the same identifier of a PoP in addition to the condition defined for the timestamp difference (cf. Section V). The completeness criterion for a Fact-List is however unchanged.

Finally, the implementation of a reimbursement calculator is unaffected as long as no changes are made to the reimbursement functions and weighting factors applied. Table IV summarizes changes to entities in the implementation, if the aggregated bandwidth SLO is changed to a per PoP SLO.

TABLE IV
Impact of a Change from Aggregated Bandwidth SLO to a per PoP Basis

Entity	Changes
AURIC	None
Bandwidth usage meter	Meter must also distinguish traffic based on destination IP addresses and a measurement record generated by an AR must contain the identifier of the PoP
Auditing logic	The `Process()` method of the subclass of the `GroupingFunction` must group measurement records also by PoP identifier in addition to timestamp, when it generates a Fact-List.
Reimbursement calculator	None

This example shows that no changes are needed for AURIC, while only minimal and isolated changes are required for the auditing application.

D. Economic Gain

Setting up and operating an automated auditing infrastructure involve some costs, whose amount depends on the number of SLOs to be monitored. Setup costs comprise procurement and installation costs of auditing hardware and software, while operational costs are mainly dominated by maintenance costs, *i.e.*, costs for upgrade, repair, or replacements.

However, in addition to those advantages mentioned in Section I, an operational automated auditing infrastructure also provides economic gain, which appears in forms of:

- A chance to take corrective actions at a very early stage of an SLA violation. This avoid potential greater loss caused by more customer claims and may gain customer confidence.
- Reduced efforts in Customer Relationship Management (CRM): Less customer claims lead to less CRM efforts.
- Reduced efforts of technical support team: Technical support team does not need to manually audit the performance of its service infrastructure, if there is a customer claim.

VII Summary and Conclusions

SLAs determine an important instrument to contract a provider's commitment on quality of its services. Thus, SLA compliance auditing is a must and an automated mechanism is advantageous, due to the fact that manual auditing is error-prone and inefficient for a large number of audit data. This paper presents a detailed scenario for hosted streaming services, including definitions and example specifications of five main SLOs. Bandwidth SLOs, as defined in this paper, represent a new type of SLO, which is of high importance for a provider to offer a streaming service.

Key requirements for an SLA compliance auditing are derived from the scenario, which comprise load scalability, flexibility, and multi-domains support. The prototypical implementation shows a linear scalability of processing time with respect to the number of measurement records, and the analytical evaluation shows that SLO changes can be accomodated easily. In addition, multi-domains support is given through the use of Diameter protocol. Furthermore, a new reimbursement scheme is proposed, which considers both the degree and the duration of an SLO violation in calculating reimbursements.

Concluding, automated SLA compliance auditing can potentially achieve economic gain, in particular for a large number of customers and services. Finally, the architecture designed enables a third party to offer SLO auditing as a service to any service providers, thus allows for an outsourcing of audit tasks.

Acknowledgment

The work has been performed partially in the framework of the EU IST project EMANICS (FP6-2004-IST-4).

References

[1] AT&T: *AT&T Business Service Guide: AT&T Managed Internet Service*; 2007.
[2] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko: *Diameter Base Protocol*; IETF RFC 3588, September 2003.
[3] L. Ciavattone, A. Morton, G. Ramachandran: *Standardized Active Measurements on a Tier 1 IP Backbone*; IEEE Communications, June 2003.
[4] G. Coulouris, J. Dollimore, T. Kindberg: *Distributed Systems: Concepts and Design*; Addison-Wesley Longman, 2005.
[5] Hasan, B. Stiller: *AURIC: A Scalable and Highly Reusable SLA Compliance Auditing Framework*; 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2007), San Jose, USA, October 2007.
[6] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence: *Generic AAA Architecture*; IETF RFC 2903, August 2000.

[7] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz: *OASIS Reference Model for Service Oriented Architecture 1.0*; OASIS Standard, October 2006.
[8] V. Paxson, G. Almes, J. Mahdavi, M. Mathis: *Framework for IP Performance Metrics*; IETF, RFC 2330, May 1998.
[9] V. Raisanen, G. Grotefeld, A. Morton: *Network Performance Measurement with Periodic Streams*; IETF, RFC 3432, November 2002.
[10] V. Smotlacha: *One-way Delay Measurement Using NTP Synchronisation*; TERENA Networking Conference, Zagreb, Croatia, 2003.
[11] Tcpdump/libpcap Website; http://www.tcpdump.org/, last visited: May 2008.
[12] Telemanagement Forum: *SLA Management Handbook*; V1.5. GB917, 2001.
[13] Verizon Business: *Performance Statistics*; http://www.verizonbusiness.com/about/network/latency.
[14] D. C. Verma: *Service Level Agreements on IP Networks*; Proceedings of the IEEE, Vol. 92, Issue 9, September 2004.