Magneto Approach to QoS Monitoring

Sidath Handurukande[1], Szymon Fedor[1], Stefan Wallin[2], Martin Zach [3]
[sidath.handurukande | szymon.fedor]@ericsson.com, stefan.wallin@ltu.se, martin.zach@siemens.com
[1] Network Management Lab, LM Ericsson, Athlone, Ireland
[2] Luleå University of Technology LTU, Mobile Systems, Skelleftea, Sweden
[3] Siemens AG Austria, Vienna, Austria

Abstract

Quality of Service (QoS) monitoring of end-user services is an integral and indispensable part of service management. However in large, heterogeneous and complex networks where there are many services, many types of end-user devices, and huge numbers of subscribers, it is not trivial to monitor QoS and estimate the status of Service Level Agreements (SLAs). Furthermore, the overwhelming majority of end-terminals do not provide precise information about QoS which aggravates the difficulty of keeping track of SLAs. In this paper, we describe a solution that combines a number of techniques in a novel and unique way to overcome the complexity and difficulty of QoS monitoring. Our solution uses a model driven approach to service modeling, data mining techniques on small sample sets of terminal QoS reports (from "smarter" end-user devices), and network level key performance indicators (N-KPIs) from probes to address this problem. Service modeling techniques empowered with a modeling engine and a purpose-built language hide the complexity of SLA status monitoring. The data mining technique uses its own engine and learnt data models to estimate QoS values based on N-KPIs, and feeds the estimated values to the modeling engine to calculate SLAs. We describe our solution, the prototype and experimental results in the paper.

Keywords; QoS, SLA, service-modeling, data-mining, terminal-reports, network-KPI, IPTV

I. INTRODUCTION

Currently telecom operators and Internet Service Providers (ISPs) offer and manage large numbers of services. These include triple play and quadruple play services (voice, data, TV, wireless) among other services. In a very competitive market it is important to monitor the service quality that is delivered to subscribers and if the Quality of the Service (QoS) delivered to subscribers is below acceptable levels, corrective measures should be taken. Otherwise, unsatisfied customers will churn resulting in loss of business, revenue and reputation.

Service quality monitoring is a very important part of service management; without knowing the QoS value of a particular service it is difficult to take the corrective actions that are necessary to improve the service. However, service quality monitoring in a large network is a daunting task because of (1) the large number of subscribers and many services (2) the lack of precise information about QoS that is necessary (for example from end-user terminals) (3) the heterogeneity of services, end-user equipments and network devices that are used to deliver the service.

In this paper we describe, a QoS monitoring solution that combines a number of techniques to address the problem of SLA status monitoring. We use a small sample set of terminal QoS reports, lower level network performance details and data mining to find data models which map lower level network performance values to the QoS values obtained with QoS reports. Then we implement service modeling based SLA status calculations. For users who are not sending QoS reports, QoS values are estimated from lower level network performance measurements using constructed data models. This approach is novel and unique in the way we orchestrate these techniques. In short, in our solution, we use a service model based approach in tandem with a data mining approach to monitor QoS. The model based approach uses a modeling engine to calculate the status of Service Level Agreements based on the information it receives from a data mining engine and other components. The data mining engine uses a small sample set of QoS values it receives from "smarter" end-user terminals and network probes to automatically build a so-called data model which represents relationship between QoS values and information from the network probes. The data mining engine then uses this data model to estimate the QoS values for the other end-user devices that are incapable of producing QoS values[1]. These estimates are then fed into the service modeling engine to calculate time indexed SLAs. The QoS value reports are created using software components that run in end-user terminals but it is unrealistic to expect that all terminals can or would be allowed to create QoS reports in a large heterogeneous network. The estimation method mentioned above circumvents this problem. Estimation is achieved by using network probes to extract specific details from the network that are particularly relevant for particular service traffic; more specifically we use probes to find values of network level key performance indicators (N-KPIs). We describe the details of our solution in next sections.

This work is done as part of the Magneto Celtic Project [1] of which one of the main research strands is QoS monitoring using the approach described in this paper. In the Magneto project context we have developed novel QoS monitoring solution for IPTV service delivered over RTSP [2] to Home Area Networks (HANs). Our QoS monitoring solution is developed and tested in the context of IPTV service in HANs, but the concept and the solution is general enough to be used in other contexts. In this paper we show one specific application of our solution in the domain of IPTV service quality monitoring that is delivered to HANs.

The structure of the paper is as follows: Section II describes the context in which we developed our solution -namely the use of our approach for IPTV service delivery and performance report collection in Home Area Networks (HANs). Section III describes a service modeling based approach to monitor and

[1] as of now overwhelming majority of end-user devices are not capable of producing QoS reports

calculate SLAs based on the underlying framework that provides QoS reports/estimates and N-KPIs. Section IV describes a data mining approach that uses a sample set of terminal reports and automatically estimates QoS for the rest of the users and feeds these estimates into modeling engine. In Section V we describe our prototype and explain the performance results. The related work is described in Section VI and Section VII concludes the paper.

II. HOME AREA NETWORK (HAN) MANAGEMENT AND PERFORMANCE REPORT COLLECTION

A. HAN Management

We have developed and evaluated our solution in the context of IPTV service that is delivered over RTSP [2] to Home Area Networks (HANs). Given that IPTV end-user services are delivered to HANs (that are not directly under the control of operators' network) it is important to monitor and manage the service adequately; otherwise due to network issues, end-user service performance could be impacted negatively creating unsatisfied end-users. In the Magneto project we aim to manage the outer-edge of the network so that:

1) Magneto Enabled Home Gateways (HGs, home gateway is the intermediate device between the operators network and HAN such as ADSL router, Cable modem etc.) and other end-user devices (such as Set-top-boxes and laptops) can send performance reports (e.g., IPTV QoS related and network performance related as described later in this section) to QoS monitoring system of the operator's network. From the QoS monitoring perspective this functionality is important and we describe these functionalities in this paper. Devices that run "Magneto" software components are known as Magneto enabled devices in contrast to other traditional HAN devices.

2) To manage the HAN and devices within the HAN (with user's consent), Magneto enabled devices including HGs can be automatically configured based on information from operator's network management system. These functionalities of Magneto network are outside the scope of this paper and they are described in [3].

3) Magneto enabled Home Gateways are smart in the sense, they can help to manage legacy, less capable devices in HANs (e.g. simple VoIP phones). We do not describe these functionalities in this paper.

B. Terminal Reports on QoS

In general, terminal reports (from end-user devices Set-top-boxes and laptops) are used to report various details about service performance metrics, user behaviour and any other details that could be useful in managing end-user terminal and service management. In our solution we use terminal reports (from "smarter" end-user devices that are capable of creating and sending such reports) to report the service performance of IPTV; more specifically we consider computers with installed the Video LAN Client (VLC) [4] as a video terminal and Set-Top-Boxes (STB) that are capable of creating terminal reports. In Magneto project, we have extended and adapted VLC client and a STB platform so that they can create terminal QoS reports about IPTV service. In our solution, the terminal reports can include:

(i) High level service impairment, more specifically video impairments. These impairment issues are detected by IPTV player (application) or video CODEC. In the Magneto solution, the following video impairments can be detected: (1) Choppiness (missing frames resulting in a sudden "jump" of video) (2) frame freezes (picture freezes for a longer time) (3) AV out of sync issues (audio, video synchronization issues resulting, for example, in the speaker's lips not being synchronized with the voice)

Information related to such problems is available to IPTV players (applications) or CODECs (e.g., in STB). It is very difficult or sometimes impossible to detect such problems without the involvement of the player/application or the CODEC. Since these impairments are related to the quality of the end-user service (e.g., IPTV end-user service) sent by terminals (e.g., computing device playing the video or STB), we refer to them as *terminal QoS reports*. Although information related to QoS is always available to the application/CODEC, it is not always possible to get this information. Not all terminals, applications or CODECs are capable of sending this information or APIs to access this information are not implemented. Though it is a small percentage, some terminals, such as "Magneto enabled" (that run Magneto software components) are capable of providing this information. The VLC extension and the extended STB platform developed in the Magneto project can detect above mentioned video impairments; once these impairments are detected QoS reports are sent to operators' QoS monitoring system. More details about these mechanisms is given in Section V.

(ii) In addition to the above QoS reports, terminals (end-user equipment) can report various network level performance reports. Such network level reports can also be generated from other network nodes such as Magneto enabled Home gateways (or even at aggregation nodes in the access networks) that are capable of collecting and sending such network level performance reports. We call these network level performance reports as N-KPI reports and they are described in the following section.

C. N-KPIs and Collection Mechanisms

Network KPIs (N-KPIs) are Key Performance Indicators that indicate the performance of network links and nodes. We specifically consider the performance of IP level links. In the Magneto solution we consider a set of N-KPIs that include: (1) packet loss (2) jitter (3) delay. These network level performance values can be observed at the: (1) end-user device (2) intermediate devices such as Home Gateways, intermediate routers and other aggregator devices in the access networks. In Magneto project, we have developed "Probes" software components that can be deployed in Home Gateways or end-user devices (such as laptops) to collect above mentioned N-KPIs for certain traffic classes such as IPTV. More details about these "Probes" are given in Section V. In IPTV service a majority of video service impairment is caused by network performance problems such as packet loss, jitter and delay.

Since operators are more concerned about the QoS issues caused by these network problems, in this paper we consider QoS impairments caused by network issues.

We assume there is a small percentage of devices that send QoS terminal reports (for example devices provided by the operator as opposed to user purchased ones or Magneto enabled devices). We use this small percentage of devices as a "sample set" in our solution. More precisely we use this sample set to build a correlation between QoS for a specific service (e.g., IPTV) and N-KPIs that are collected from various network equipment (such as HG, intermediate routers, aggregation nodes etc.). Based on this correlation we estimate QoS levels of end-user service and then calculate the Service Level Agreements (SLAs). Next, we briefly describe how we use the QoS reports, N-KPIs together with a data-mining engine (for QoS estimation) and modeling engine (for SLA calculation based on service modeling).

D. *Overall Architecture*

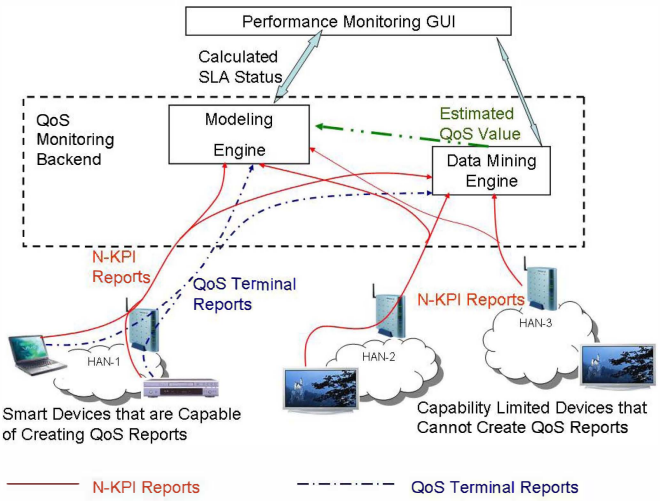

Figure 1: Overall Architecture of QoS Monitoring Solution

In Figure 1, we show the overall architecture of the QoS monitoring solution. The above mentioned QoS terminal reports (when available) and N-KPIs are fed into the Modeling Engine and Data-mining engine. The SLA status is calculated for those terminals that can send QoS reports, based on the QoS terminal reports from those terminals; these SLAs are useful in evaluating the overall end-user service quality delivered to users. The Modeling Engine also calculates the SLAs for network performance based on the N-KPI it receives; these SLAs are network connectivity metrics that are useful for evaluating performance of the network. The N-KPIs are sent from intermediate nodes such as the HG or routers.

The QoS values and the N-KPIs are forwarded to the Data-Mining Engine and Data-Mining Engine which builds a data model to map N-KPIs to QoS values. This data model is periodically updated. The Data-Mining Engine uses the data model to estimate the QoS values for terminals that cannot send QoS value reports. These estimates are fed into the Modeling Engine so that it can calculate the SLA status for terminals that cannot send QoS reports. Both SLA status and estimated QoS values are sent to the Performance Monitoring Graphical User Interface (GUI) so that operator can observe the SLA status and QoS estimates.

In the next section we describe how SLAs are calculated using a service modeling approach and subsequently we describe how QoS values are estimated using N-KPIs based on the Data-Mining approach.

Assumptions and Limitations. When estimating QoS values of IPTV service, our solution can only estimate QoS values for impairments that are caused by network level problems but not errors caused at encoding stage for example audio/video synchronization errors. In addition, there would be a certain overhead to collect N-KPI values. Another limitation is that if certain system settings are changed, for example the protocol used for IPTV (from RTP to some other protocol), the Data-Mining Engine needs to build the data model again.

III. SERVICE MODELLING FOR SLA CALCULATION

In general, a service model is a formal high level specification of a service, the relations between a service and the resources (network and other) on which the service is built and the way in which the service is made available to the users of that service [5]. A form of service model, known as service topology, represents dependencies between resources and services. In addition, these service models include (1) higher level service QoS which represents how a higher level service (e.g., IPTV) perform (2) lower level network/resource KPIs which indicate how a lower level network service (e.g. IP delivery service) performs (3) service levels (service goals and objectives) (4) Service Level Agreements (SLAs) and (5) relationships between network resources, KPIs, end-user service, QoS, users and SLAs. This form of service modeling captures the basic details of the technical implementation of a service and the requirements of that service in terms of resources.

We use these service models to monitor and analyse the quality of a service, its overall performance, and to calculate SLAs based on the modeling framework that is described later in this chapter. SLAs can be calculated for:

- **End-user Services**: for operators it is important to monitor the performance of delivered service to individual users as well as determining the overall service delivery quality averaged over all users. In the Magneto solution SLAs for the overall IPTV service are calculated and monitored using service modeling and a modeling engine (the SALmon engine, described below).

- **Network Services**: in addition to the end-user service QoS, the modeling engine can also calculate the SLAs for networks, (e.g., for IP links). This is useful when monitoring of a particular network segments is required (e.g., access network) to determine if that network segment is performing as expected.

A. *Service Modeling Framework*

We express the service model in the SALmon environment [11], which is a tailor-made language for expressing service models. SALmon combines object-oriented

structuring for service model decomposition and functional expressions for status calculations. Due to the nature of service modelling, the programming language must have the capability to treat time as part of the normal syntax: all variables are seen as arrays indexed by a time stamp. It is possible to use the time-index syntax to retrospectively change the value of variables. This is important in many scenarios, for example late arrival of probe data. This enables SALmon to recalculate SLA status whenever KPIs are reported.

List comprehension and an extensive set of built-in functions provide the power needed to express complex models. The language has two fundamental layers: the *Definition Layer* and the *Instantiation Layer.* The definition layer defines the classes and calculations in the model. Core concepts that we want to represent as classes are Services, Service Levels and SLAs. Classes have inputs, anchors, attributes and properties. The instantiation layer creates instances of the service classes, assigns properties and establishes connections between instances through anchors.
Our model is based on a number of layers that include:

1. *Service Layer:* The layer exposed to the user, where Quality of Service (QoS) is measured.
2. *Transport Layer:* where network impairments such as loss, delay, and jitter may occur.

Figure 2 illustrates the classes in our IPTV model. We have two major classes in the Service Layer: `IPTVUserService` and `IPTVProviderService`. The `IPTVUserService` represents the status and QoS for an individual end- user and the `IPTVProviderService` aggregates user services into an overall IPTV service quality from a virtual provider perspective. The `IPTVUserService` depends on the application and network components: `RTSPSession`, (Realtime Streaming Protocol [2]), and `IPConnectivity`. Service levels verify that each service performs within configured thresholds. SLAs finally represent the agreement regarding provided service levels between the provider and the user.

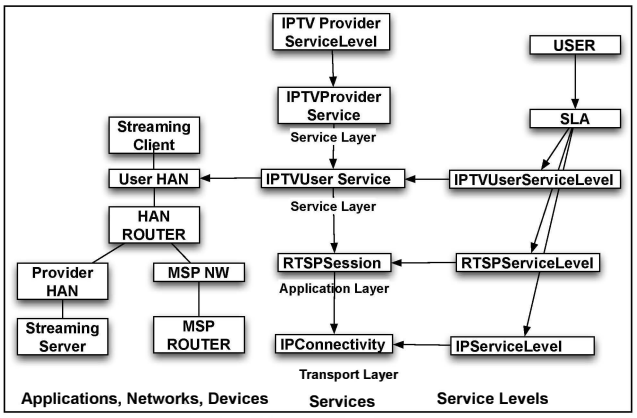

Figure 2 : Service Model

The QoS reports with end-user service performance metrics (more specifically IPTV service KPIs) are sent to the modelling engine using the SOAP protocol. In other words, terminal QoS reports are fed into the modelling engine to calculate the status of end-user SLAs. Due to the time-awareness of the SALmon engine it is capable of recalculating SLA status whenever reports arrive. More comprehensive details about our IPTV service modelling approach are described in [12]. Since it is not always feasible to get terminal reports with QoS status, we use data mining techniques to estimate the QoS values for users whose devices cannot send terminal reports. These estimates are then sent into the modeling engine to calculate the SLAs of end-user services. The next chapter describes how these estimates are performed using data mining techniques.

Similar to QoS reports, N-KPIs are fed into the modeling engine by the N-KPI probes in a manner similar to that described for QoS reports.

IV. QoS Estimation through Data Mining

Our system uses Data Mining (DM) techniques to automatically learn and apply the functional relationship between N-KPIs and QoS values. In the learning phase, the system monitors N-KPI values for cases where the QoS values are known and learns the functional relationship between those N-KPI and QoS values. Once the functional relationship is learnt, it can be applied to estimate QoS values from N-KPIs in cases where the QoS values are not known.

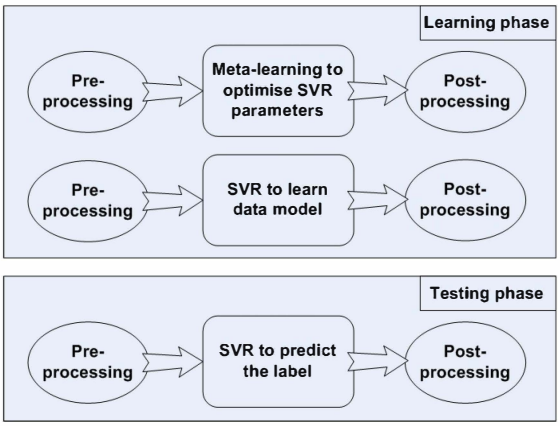

Figure 3 Learning and testing phases of the DM predictive task.

Our system uses DM methods for "predictive task" (based on the data-mining terminology). That is, forecasting [2] the values of QoS values based on the values of N-KPI measurements. A predictive process is composed of two phases; learning and testing phase (see Figure 3). During the learning phase the system is automatically trained using a learning algorithm. We are using a supervised learning method i.e. the system is provided with data examples, each composed of several attributes and a label which we aim to predict in the testing phase. The output of the learning phase is a so-called data model which specifies dependencies between values of attributes and corresponding ranges of the label. In our system

[2] We are also using the term "predicting" according to the DM terminology interchangeably with "estimating" in this paper.

attributes are N-KPI values and the label corresponds to QoS values with both, N-KPIs and QoS values being numeric. There are numerous DM algorithms to estimate relationship between quantitative attributes and numeric label [6] e.g. polynomial fitting, linear regression, Naïve Bayes.

In our experiments we use a Support Vector Regression (SVR) method which has many desirable qualities that make it one of the most widely used regression algorithms [6]. The SVR algorithm takes as an input a number of training samples, each characterized by a number of attributes (N-KPIs) and labels (QoS values).

It produces a function (i.e. a data model) which only depends on a subset of the training data, because the cost function for building the model ignores any training data that is close (within a predefined threshold) to the model prediction. More details about SVR can be found in [7].

As explained earlier, there are several parameters that influence performance of the SVR algorithm applied to a specific data set. Moreover SVR can use different kernel functions for computation of non-linear data models. These parameters can be tuned in the first part of learning phase using a so-called meta-learning algorithm applied to the training samples. We use a cross-validation for meta-learning [6]. With this method each record of the training set is used the same number of times for building a preliminary data model and estimating its performance. For example, we use a 3-fold cross-validation method in which training samples are segmented into 3 equal-sized partitions. The method then builds a model of the data 3 times and tests its performance. During each of 3 runs, one of the partitions is chosen for testing the performance while the rest of them are used for training the model. This procedure is repeated 3 times so that each partition is used for testing exactly once. The method uses a root mean square metric to estimate the performance of each round i.e. it compares root mean square of the distance between measured QoS value of samples from the testing partitions and their corresponding predicted value.

The cross-validation method outputs the expected performance of the data model built using SVR algorithm with specific values assigned to its parameters. Therefore, we run the cross-validation method with different configurations of the SVR algorithm and we estimate the optimal configuration to be used during the learning phase to build the data model of the training set. In the final stage of the learning phase, the system uses the SVR algorithm with the optimized parameters and builds the definitive data model which will be applied in the testing phase.

During the testing phase, the system is only provided with attributes (N-KPIs in our scenario) and it applies the previously build data model to predict label values (QoS). We expect that the data provided for the testing phase follows the previously build data model and therefore the predicted QoS value corresponds to the real value.

The DM algorithms are applied as a part of the Knowledge Discovery Process (KDP) composed of Data pre-processing, DM algorithm and Data post-processing components. Therefore, each instance of the DM algorithm (i.e. meta-learning, learning and testing phases) is preceded by the pre-processing and followed by a post-processing step.

The aim of the pre-processing step is to prepare the input raw data for the DM algorithm. Collected QoS values and N-KPI samples need to be time correlated and their ranges and types must be defined. Also, N-KPIs which do not have an impact on QoS are excluded from the input to the DM algorithm. Moreover, the system needs to have a so-called label which is an attribute that will be predicted with the DM algorithm: the QoS attribute in our case. More details about our implemented pre-processing stage are described in Section V.

The post-processing step ensures that only valid and useful results of the DM algorithm are preserved for further analysis. Therefore for meta-learning the optimal parameters of the SVR algorithm are stored in a format that can be easily used in the learning phase. The post-processing step of the learning phase transforms the data model into the format that can be applied during the testing phase. Finally, the post-processing step which follows the testing phase consists mainly of presentation of the results to the analyst using different visualization formats.

V. PROTOTYPE SETUP AND RESULTS

A. *The Test-bed*

To evaluate our solution we built a test-bed that can stream IPTV content over RTSP and also built other components in our solution. This test-bed setup is shown in Figure 4. In our set-up we have SALmon engine with IPTV service model (described in Section III) running in one computer and the Rapid Miner with developed data mining processes running in another computer. A MySql database is used to store collected QoS values and N-KPIs. A streaming server that can stream over RTSP protocol is also used. These components are attached to an Ethernet switch as depicted in Figure 4. In addition to the above components, a Linux server with multiple Ethernet cards is used as a network emulator with IPTables [14] and Netem [15] tools installed to emulate a network link. These tools can introduce network impairments such as packet loss, jitter, delay etc. In our experiments we used these tools to introduce network impairments so that there will be various changes in N-KPIs. The test-bed also contains an emulated home gateway and a laptop running VLC client that is extended (described later in this section) such a way so that it can send QoS values to the database. The N-KPI probe (described later in the chapter) observes the IPTV traffic and sends N-KPIs to the database. In our experiment, we introduce network impairments such as packet loss when the IPTV content is streamed to the laptop. The extended VLC client and adaptor send the detected QoS values for different packet loss levels and at the same time the network probe sends the detected N-KPIs values. Using these data, initially, the data-mining engine builds the data model (the relationship between QoS values and N-KPIs). Once this initial phase is over we change the packet loss arbitrarily and estimates the QoS values based on N-KPIs.

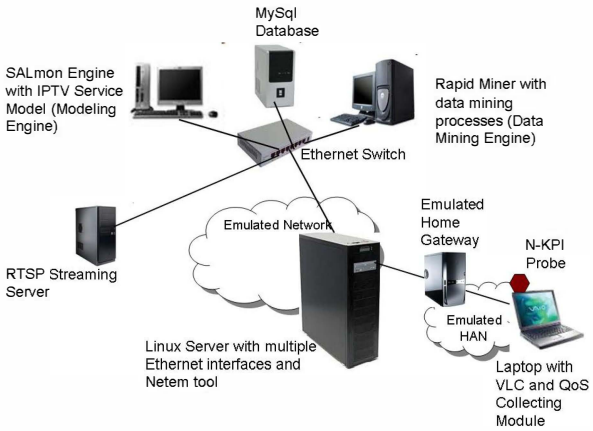

Figure 4 Testbed architecture

B. QoS Value Collection from Terminals

We developed a Magneto event processing module (based on open source Complex Event Processing engine ESPER [16]) on top of OSGI to process different events at the HG. The processing module can be easily deployed on any type of HG capable of running OSGI framework. Moreover, thanks to its modular architecture it can be easily expanded and adapted to various HAN architectures. The processing engine consists of several OSGI bundles realizing different objectives. The VLC adapter bundle handles log messages from VLC client and sends them to the Event Processing bundle which processes them according to the predefined rules. The result of this processing is sent to the Modeling adapter bundle and Data Mining adapter. The Modeling adapter bundle feeds the Modeling Engine with the QoS measurements whereas Data Mining adapter sends QoS measurements to the Data Mining Engine for further analysis.

C. N-KPI Collection of IPTV Traffic

The N-KPI probe is built using wrapper application on TShark [17]. The wrapper application is built using Java with an interface to native code. The probe is then configured in such a way it monitors IPTV traffic (which is based on RTSP which in turn use RTP) for a configurable period. During this period, the probe collects data related to IPTV/RTSP/RTP traffic [18]. More precisely, it monitors the packet loss, max jitter, mean jitter and max delta [18] that is available from TShark tool. In parallel with monitoring thread, at the end of each period, the collected data is analysed in a different concurrent thread, to find the N-KPI values. Once the analysis is finished, the calculated N-KPIs values are sent to the database together with the timestamp. While this thread analyses the collected data, the initial thread continues to monitor the IPTV traffic collecting data for the next period.

D. QoS Value Estimation using Rapid Miner

Data Mining Engine (DME) is based on Rapid Miner [8] which allows an easy composition of the full chain for the KDP. We implemented two phases of the predictive process to forecast QoS values on the basis of N-KPI measurements. DME receives QoS and N-KPI values from different sources and in the pre-processing task prepares them for the DM algorithm. In our experiments we used QoS choppiness (QoS_C) events from VLC but our technique is not tight to this particular QoS values. QoS_C values are measured every 2 seconds at the VLC client and later processed by the Magneto event processing module along with their corresponding timestamp, service ID and user ID. QoS_C values correspond to the number of choppiness events which were reported by the VLC client. N-KPI values are measured over 20 seconds period and they are received in N-KPI reports which consist of several N-KPI measurements (packet loss, packet mean jitter, packet max jitter and packet max delta) with corresponding timestamp. The pre-processing task outputs a single table composed of numerous records having as attributes QoS_C values and N-KPI measurements as shown in Figure 5. Time attribute corresponds to the timestamp of every N-KPI report. QoS_C represents the number of choppiness events reported by VLC between two consecutive N-KPI reports (i.e. during the time when the corresponding packet loss, packet mean jitter, packet max jitter and packet max delta were estimated).

TIME	PKT_LOSS	JITTER	MAX_JITTER	MAX_DELTA	QoS_C

Figure 5 Output of Pre-processing task

To build the relationship between QoS_C label and N-KPI attributes, we need to select the N-KPI attributes which have impact on the value of QoS_C. We do that on the basis of the correlation matrix build with DME and presented in Figure 6. We can see that QoS_C label is highly correlated with Packet Loss and Mean Jitter N-KPIs. Therefore we only use these N-KPIs for the DM predictive task.

Attributes	Packet Loss	Mean Jitter	Max Jitter	Max Delta	QoS_C
Packet Loss	1	0.771	0.155	0.004	0.862
Mean Jitter	0.771	1	0.461	0.078	0.752
Max Jitter	0.155	0.461	1	0.039	0.115
Max Delta	0.004	0.078	0.039	1	0.003
QoS_C	0.862	0.752	0.115	0.003	1

Figure 6 Correlation Matrix of N-KPI and QoS_C attributs

To perform learning phase of DM process we collect N-KPI and QoS_C measurements of an IPTV session lasting for 45 minutes. We emulate variation of the N-KPI values using Netem tool installed in the emulated network which allows us to introduce packet loss into the streaming session. We vary packet loss across different values between 0% and 4% and measure the actual packet loss at the laptop with VLC client. The presence of packet loss causes also variation of other N-KPIs. Above 4% packet loss, the video stream is impossible to follow by a viewer.

First we tune the DM algorithm for the learning phase using cross-validation meta-learning process. We run several rounds of cross-validation method changing kernel functions of the SVR algorithm and their parameters at every round. We compared performance of the DM algorithm expressed as the Root Mean Square Error (RMSE) of the QoS_C value estimated at each round with specific kernel function and its parameters. As a result we selected Dot Product kernel function with its parameter C=17.267 because it had the lowest RMSE equal to 34.788.

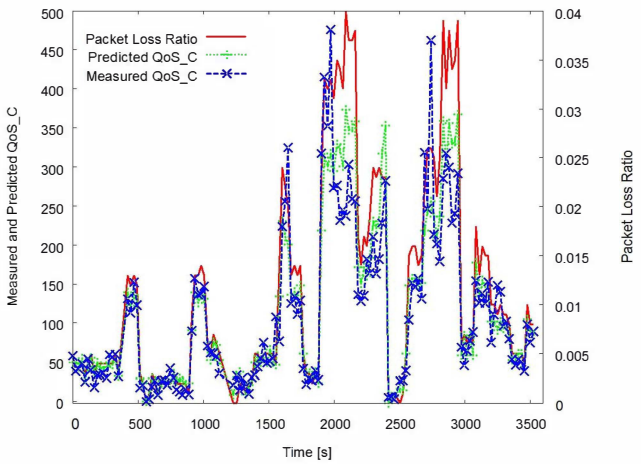

Figure 7 Measured Packet Loss Ratio and measured QoS_C values and predicted QoS_C values during the 1 hour video stream.

To finalize the learning phase, we assigned the optimal kernel function and its parameters to the SVR algorithm to learn the model of the relationship between QoS_C and N-KPI values (Packet Loss and Mean Jitter). The obtained data model was applied to a new IPTV stream during the testing phase. Due to space limitation we do not show details and assessment of the learning phase (e.g., amount of data, time, processing requirement)

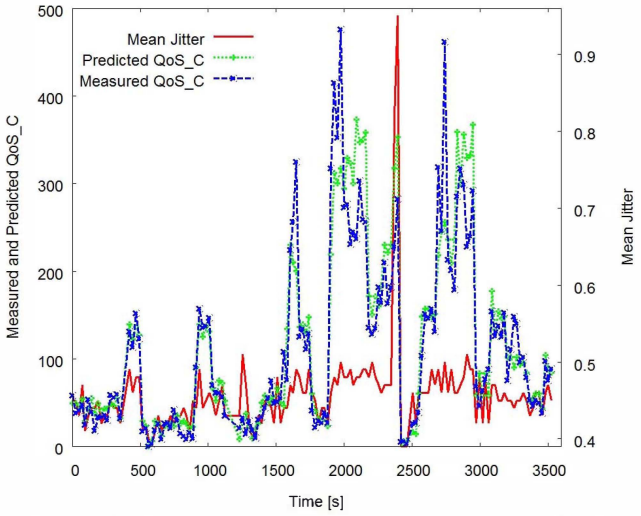

Figure 8 Measured Mean Jitter and measured QoS_C values and predicted QoS_C values during the 1h video stream.

We monitored N-KPI values during a 1 hour of IPTV streaming and we sent them to the DM Engine for the processing during testing phase. We predicted QoS_C labels on the basis of previously built data model and collected N-KPI values, using SVR with predefined parameters. The evolution of measured Packet Loss Ratio and QoS_C values during the played IPTV stream is depicted in Figure 7. The figure also shows measured QoS_C values that we monitored in order to compare them with the forecast. We can see that predicted and measured QoS_C values are close. Figure 8 depicts the variation of measured QoS_C and Mean Jitter values during the same IPTV stream.

The correlation between Mean Jitter and measured QoS_C is visible but not as much pronounced as for Packet Loss Ratio and measured QoS_C quantities. In this test scenario we did not control the variation of the Mean Jitter value and its change was due to the variation of Packet Loss Ratio enabled by Netem tool. As a future work we plan to emulate Mean Jitter in a more controllable manner (by setting it with Netem tool) and we will evaluate its impact on the QoS C.

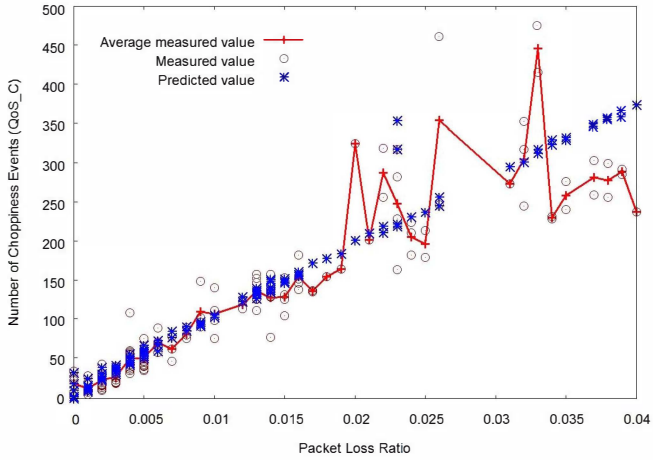

Figure 9 Measured and predicted values of QoS_C against Packet Loss Ratio.

The correlation between Packet Loss Ratio and QoS_C values measured during the 1 hour IPTV stream is shown in Figure 9. It shows also the forecasted QoS_C values. The prediction is very accurate, especially for the values of packet loss ratio below 0,02. For the larger values of packet loss, the variance of measured QoS_C values measured for the same value of packet loss ratio increases significantly and therefore it is much more difficult to forecast QoS_C. The RMSE of the forecast during the testing phase is 42.42. We can also see that the DM Engine predicts several values of QoS_C for the same Packet Loss Ratio. This is because data model depends on two parameters, Packet Loss Ratio and mean jitter, and the latter parameter has the impact, although less significant than Packet Loss Ratio, on the forecast of QoS_C.

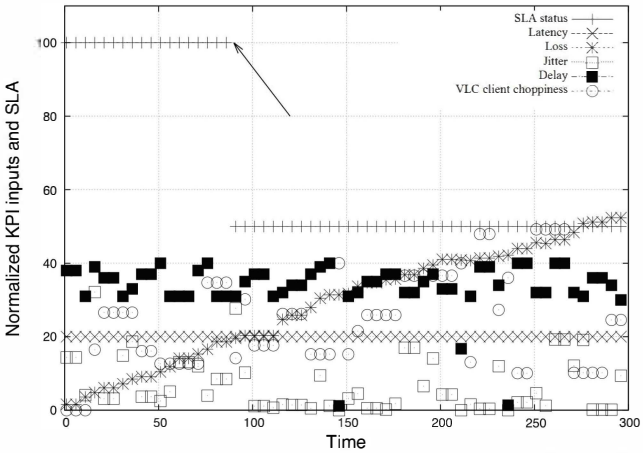

Figure 10 SLA and inputs. The top line in the above figure denotes the SLA status which goes from OK to jeopardized at the arrow.

E. SLA Calculation using Modeling Engine

The SLA engine is able to calculate overall SLA status for individual users as well as aggregated services. This gives a view on the Quality of Experience based on the service model which takes various KPIs into account in combination with the service and network topology. An example is shown in Figure 10. The other lines illustrate the myriad of inputs (KPIs) such as loss, jitter and delay. We have normalized the KPI values in order to illustrate different values in same plot. In this specific scenario the major impact comes from increasing loss which makes the SLA status decrease.

VI. Related Work

Network performance measurement has a long tradition and is commonly used for network management and monitoring. In the era of numerous services being deployed in a network (VoIP, IPTV, MMTel) ISPs and operators are putting more effort to monitor service performance. Several publications describe possibility of monitoring service performance on the basis of measurements obtained from network elements.

Furuya at al. [10] investigated relationship between IP network performances and voice quality for VoIP service. The authors discovered high correlation between each of the IP network performances and the corresponding voice quality. They suggest the possibility managing the quality of service of VoIP by monitoring the corresponding network performances.

Racz et al. [9] describes architecture for network and service quality measurement. The architecture can determine general network properties (e.g., packet delay, packet loss) as well as the characteristics of a service (single VoIP session performance). The authors do not describe how these values are linked and how service parameters could be deduced from the network measurements.

Terminal reports are proposed and sometimes used in a number of contexts such as in Set-top-Boxes [19] and with RTSP client reports (i.e., RTCP [20]). However the use of terminal reports in those contexts is different from our overall solution. Network performance measurements have been researched a lot [13]. In our overall solution we used the information from terminal reports and probes.

VII. Conclusion

In this paper we presented a new method of quality monitoring of end-user services that combines a number of techniques in a novel way. The proposed solution aims to overcome problems which relate to the complexity of the environment where end-user services are provided.

Described system uses a service modeling approach which represents relationships between entities important for monitoring of service quality. It includes information about network resources delivering the service, their N-KPI values, QoS values and status of SLAs. QoS values are fed into a service modeling engine and to a data-mining engine from a small sample set of "smarter" end-user terminals. Similarly, the N-KPIs are collected from the network and fed into the same modeling and data-mining engine. The data-mining engine automatically builds a data model representing the relationship between the QoS values and N-KPIs. Subsequently, the data-mining engine estimates the QoS values for terminals where QoS terminal reports are not available and these estimates are fed into the modeling engine to calculate SLAs.

The proposed system was implemented and tested using an IPTV service. Our experimental results show that the described approach can be used to monitor quality of service in a real environment by effectively estimating the QoS values and calculating the SLA status. Our solution is not bound to the IPTV service and in the future we plan to use the system for monitoring of other type of services in even more heterogeneous environments.

Acknowledgment

This paper describes work undertaken in the context of the CELTIC MAGNETO project, which is partially funded by "Enterprise Ireland" as part of the International Collaboration Support Programme and by the Austrian FFG. The authors are very grateful for Liam Fallon, Anne-Marie Bosneag, Magneto consortium members and anonymous reviewers for their comments and support for this work.

References

[1] Magneto Project: http://projects.celtic-initiative.org/magneto/index.html

[2] Real Time Streaming Protocol (RTSP) http://tools.ietf.org/html/rfc2326

[3] J. Kielthy, K. Quinn, R. Toribio, P. Arozarena, S. Handurukande, M. Garcia Mateos, M. Zach; Design of a HAN Autonomic Control Loop IEEE MACE 2010 Workshop

[4] VLC http://www.videolan.org/vlc/

[5] Raisanen, V., Service Modeling, Wileyy, ISBN 9780470018071, 2006

[6] Tan, P.-N.; Steinbach, M.; Kumar, V. *Introduction to Data Mining;* Addison Wesley: 1 ed.; 2005.

[7] V. Vapnik, S. Golowich "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing", *Neural Information Processing Systems*, Vol. 9. MIT Press, Cambridge, MA.

[8] Rapid – I. Rapid Miner Community Edition. http://rapid-i.com

[9] P. Racz, D. Donni, B. Stiller; "An architecture and implementation for IP Network and Service Quality Measurements," Network Operations and Management Symposium (NOMS), 2010, pp.24-31, April 2010

[10] Furuya, H.; Nomoto, S.; Yamada, H.; "Experimental investigation of the relationship between IP network performances and speech quality of VoIP," Telecommunications, 10th Intl. Conference on, 2003

[11] S. Wallin, V. Leijon, and J. Ehnmark, "SALmon - A Service Mod- eling Language and Monitoring Engine," in Proceedings of the IEEE Intl. Symposium on Service-Oriented System Engineering December 2008

[12] Handurukande, S. and Wallin, S. and Jonsson, A, "IPTV service modeling in Magneto networks", Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP}, pp.51-54

[13] Bohoris, C. Pavlou, G. Cruickshank, H. Using mobile agents for network performance management. IEEE/IFIP NOMS 2000.

[14] IPTables www.netfilter.org/

[15] Netem www.linuxfoundation.org/collaborate/workgroups/networking/netem

[16] Epser http://esper.codehaus.org

[17] TShark http://www.wireshark.org/docs/man-pages/tshark.html

[18] RTP http://www.ietf.org/rfc/rfc3550.txt

[19] Data Model for a TR-069 Enabled STB http://www.broadband-forum.org/technical/download/TR-135.pdf

[20] RTP Control Protocol Extended Reports (RTCP XR) http://www.rfc-editor.org/rfc/rfc3611.txt