Service Oriented Network Framework Enabling Global QoS and Network Virtualization

Daniel Schlosser and Tobias Hoßfeld
University of Würzburg
Institute of Computer Science, Department of Distributed Systems
Würzburg, Germany
Email: [schlosser|hossfeld]@informatik.uni-wuerzburg.de

***Abstract*—Today's Internet provides access to a large variety of services, which require high data rates and stable data flows for satisfactory service quality. New interactive services have high Quality of Service (QoS) requirements to the network. Packet loss, jitter, delay and effectively available bandwidth are parameters that heavily influence the Quality of Experience (QoE) of a user. The current architecture of the Internet cannot support these QoS requirements on a global scale, as solutions of one Internet Service Provider (ISP) do not necessarily work across borders of ISP domains. In order to provide QoS on a global level, the Internet of the future might offer virtual networks as data delivery services that can guarantee all the requirements of an application or service. We therefore propose a service oriented network framework. Within this framework different virtual network connections with the corresponding pricing schemes that satisfy a given request will be offered to the customer.**

In this paper we (1) introduce the architecture concept, (2) show how the QoS of a network path can be concluded from QoS description of its parts and (3) perform a measurement study to evaluate a measurement approach, which classifies the QoS of a network connection between two routers based on active measurements performed by the routers themselves.

I. Introduction

The Internet of today is used as a general communication medium. It works great for delivering web pages, text messages, files, and all other types of content which have no need for real time delivery or high interactivity. It is shown in [1], [2] that the Quality of Experience (QoE) perceived by the user for the classic services like web site delivery and ftp is mainly dependent on accessible bandwidth. With the increasing availability of broadband access these services can satisfy the users needs with the current architecture of the Internet.

However, in the last years more and more services are provided within the Internet, which need real time traffic delivery as e.g. streaming services for music or video. For these services it is no longer sufficient to guarantee the availability of high bandwidth in order to satisfy the user. For these kinds of applications it is also necessary to provide and guarantee Quality of Service (QoS) in the network. As shown in [3], [4] the QoE of streaming services is effected by jitter and loss in the network. A network, that supports streaming services in a good quality, has to assure that streaming data is neither lost nor delayed in an irregular pattern.

Interactive services like VoIP, video conferences and online gaming demand also for low latency connections, as long delays between the interaction partners worsen the user perceived QoE [5], [6].

The Internet of today can support services like streaming, conferencing and gaming, but is not designed to be adapted to the specific needs of individual applications or services in general. It might be possible to get a QoS assured connection, if the service and the client both connect to the same Internet service provider (ISP). But users of a global service will most probably connect from different ISPs and the data may also traverse some intermediate autonomous systems (AS) in order to be transferred between the application service provider and the user. Hence, it is very hard to practically impossible to get a guaranteed QoS on the data connection of a service on the global level. This fact does not only affect the users, but also the Internet providers. The impossibility of making global QoS guarantees prevents the option of selling high quality network services, which could be charged additionally to basic network connectivity. The willingness of users to pay for those QoS improvements is shown by the fact that many users pay for enabling "fast path" on their ADSL connection. This means that the provider just disables interleaving on the last mile.

Lately, several approaches emerge, for instance [7], which propose to build virtual networks on top of already existing network topologies. The idea is to overcome the problems described above and other problems emerging from the current architecture of the Internet. From our point of view there is a need for an abstraction layer that hides the complete network implementation. Thus, we propose a service-oriented architecture, where the application or the service on application layer requests a data transfer service from the network describing the needs of the data transfer in terms of reliability and QoS. This abstraction separates the application and the network completely. It is therefore possible to modify the complete network stack and also to use virtual networks without the need of deploying new products to the customer. The network just offers different services with the corresponding pricing schemes. After the network service is chosen, the network provides a data stream assuring the quality of the data transfer. The application has neither to deal with transfer protocols nor access technologies nor reliability issues. This solves most of the problems of users, network providers and application

service providers, as the user will get improved quality, the network provider will get additional billing options and the service provider no longer has to care about quality degradations caused by the Internet.

As global QoS is the main target of our virtual network service architecture, network management plays an important role. In order to deliver the promised QoS, the network has to be monitored in detail. Even small changes in some parts of the network might effect the global QoS so that the assured parameters are no longer valid. In this case it would be necessary to reconfigure the underling virtual network in order to keep the service level agreement for this network service. Additionally, a reliable measurement of the QoS perceived by the user will increase the customer's fidelity and his satisfaction in the service. Therefore we will discuss options to measure the QoS of a virtual network service.

In the following we consider related work in section II. Section III describes the service oriented architecture (SOA) based network framework. We discuss the aspects of connecting different network parts together and predicting the QoS for the network in section IV. Furthermore we study the option of using active measurements in order to hedge QoS guarantees in sectionV. Finally, Section VI concludes this work and gives some outlook on future directions.

II. Related work

QoS enabled networks have been studied for quite a while now. Thus, several options for guaranteeing QoS for network connections have been implemented in the traditional IP architecture and for future networks not relying on IP anymore.

A part of the solutions concentrate on the prioritization of traffic in the routers queue. One of the most straight forward approaches is IntServ [8]. Using IntServ each data flow, which needs QoS guarantees has to provide a description of the traffic that will be sent using a bucket model. Each router on the path between source and destination is queried, if it is able to support the QoS requested for the flow. If all routers agree, the connection is established and QoS is guaranteed. Otherwise the flow is rejected. Unfortunately, this algorithm does not scale with increasing network size and can only be used in very small networks.

A more practical approach to QoS assured connection using traffic prioritization is DiffServ [9]. Using DiffServ, a router analyses traversing packets. Each packet is classified into priority classes, which are used to schedule the packet delivery in the router. Routers that see a classified package may adopt their behavior accordingly or change the classification. However, as there is no general rule or specification how these classes have to be handled, QoS can not be guaranteed on network paths traversing more than one administrative domain.

Another strategy to support QoS is to adopt the routing, i.e. to modify the topology of the network. Constraint based routing (CBR) [10] selects different network paths from one source to a destination for different kind of flows. Depending on the concrete framework routes are previously defined or set up in an ad hoc manner. It has to be noted that CBR does neither reserve bandwidth nor prioritize any traffic classes by default. It simply sets up different routes. The topology is optimized, e.g. for low latency, but it is not guaranteed that congestion does not affect the QoS enabled flows. Hence, CBR is often combined with DiffServ or fixed rate MPLS aggregates in order to protect the QoS enabled flows against congestion in the network.

Beside these classic approaches, which try to improve the old IP architecture of the Internet, there are new approaches, which combine different network services, use non layered protocols, or replace the complete protocol stack in order to build the Internet of the future. However, if they want to compete against classic approaches, they need to provide better QoE to the customer. Otherwise the customer will not be interested in the deployment.

In [11] an architecture called SILO is proposed which uses blocks of fine grained functionality, a way to combine them together, and control elements to make them interact smoothly. Each block might be implemented in a different way and there might also be different implementations for the same block. The application works synergistically with the network in order to build up a system that satisfies the needs of the application. The framework is open to integrate features like security and may use techniques to improve the performance even if deployed in hardware.

In [12], a network architecture is proposed, which completely renounce the layered approach of network design and proposes a non-layered paradigm, which is called *role-based* design. Each role is a function description, which is used for processing and forwarding the encapsulated data. Instead of having a fixed order, which the composition of roles has to follow, each transmission can set up its own heap in which the roles can be ordered in any suitable way. As this proposal is not compatible with the current network hardware, it would need a complete change of technology or virtual networks in order to be deployed.

The advantages of using multipath packet dispersion and reordering near the destination are analyzed in [13]. It is shown that if paths with nearly the same latency are used, the throughput and the resilience to link failures is improved. Furthermore the paper discusses the problems of out of order delivery if TCP is used. In addition, it shows that using this technique may lead to large bulk arrivals of TCP packets, which might result into other problems.

Although these three papers differ in the way the authors try to improve the data transfer, they share one common aspect. They all have to face the problem that the traditional IP based Internet is not going to support this approaches without deploying changes to the customer or in the network. Anderson et al. [7] describe a way how to test those new approaches in a virtualized network test bed. It is considered how the existing internet architecture can be tricked in order to get real traffic into this test bed and that it is a problem to achieve absolute QoS in a virtualized system. After successfully testing a new technology, customers might get attracted and the technology can be deployed. But the authors also discuss the problem that

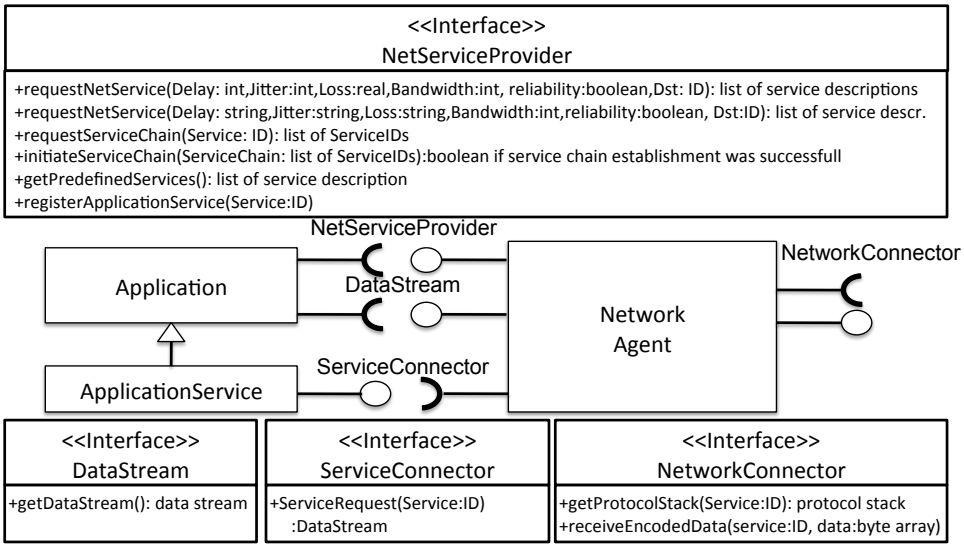

Fig. 1. SOA architecture interfaces

customers might not want to change their equipment any time a new technology advents. As an outcome they postulate the need of a coherent framework in which all the new ideas can be integrated.

In [14], the authors describe an architecture, in which a network node is able to choose and multiplex different network architectures at runtime. They propose a solution in which the network node selects, controls and optimizes all connections based on the requirements of the application and some user provided policies. They state that a abstraction between the network and the application is needed, but do not propose an appropriate interface.

In this paper we consider the problem identified by [7], [15] and propose a simple interface as an abstraction of a network service based on the service oriented architecture approach. In contrast to [15] we do not integrate the selection of a proper network realization in the functionality of the network. We assume that a user wants to decide by himself, which quality of network he wants to use for which price, especially as we consider charging different prices for different QoS. Thus we assigned this task to the user side of the interface. We also discuss how to integrate QoS guarantees for different parts of one consecutive network connection and investigate one possible option to measure the QoS provided by a such a service.

III. Service Oriented Network Framework

Network virtualization can be seen as a way to build networks that hide the underlying network topology. But it can do much more in order to improve the network service and open existing architectures for future technologies. In order to fully separate all kind of network services from the application logic, we propose the following architecture. We use the SOA approach and define the complete network transmission of the data as a service, which can be adapted to the needs of the application. The implementation of this service is completely up to the network. On the one hand this releases the service developer from the need to take care of things that might happen in the network. On the other hand the network side is free to transfer the data in any suitable way, which enables the option of using an already known protocol stack like TCP/IP or to transfer the data over any other protocol, which might come up in the future.

Figure 1 shows the service and interface definitions in UML notation. A simple application exposes no interface to the network but uses the `NetServiceProvider` and `DataStream` interface of the network. The `NetServiceProvider` interface contains all functionality which is needed by the application to request and initiate a data transfer. The `requestNetService` method is accessed by the application to get services offered for a data transfer to the target with the specified QoS level and reliability. It is possible to specify hard limits for the QoS parameters, e.g. maximum delay of 150 ms, maximal jitter of 20 ms, maximal loss of 0.001, and minimal bandwidth of 15 kbps, or to specify it less specific, e.g. a low delay and so on. Specifying a QoS parameter as the maximal value for the specific version or 'none' in the less specific version means that there is no restriction on this parameter. The reliability parameter specifies if it is necessary to transfer all the data. If this parameter is set to false, the network might drop data as long as the drop ratio threshold is kept. If this parameter is set to true, this data has to arrive in the same order it was sent and without loss. For example a file transfer would therefore ask for a high bandwidth and reliability connection in order to achieve a high throughput.

A network receiving such a request would search for paths to the specified target and calculate the supportable QoS levels and the prices for these connections. If the target is not in the same AS, this cannot be done directly. Hence, the network will request a network service from the possible next networks on

the way to the destination. These networks will perform the same procedure until the destination is reached.

Finally the network to which the user is connected offers all services satisfying the QoS requirements to the user. The user or a user agent decides which one to use and requests the complete network service chain (`requestServiceChain`) in order to establish the connection (`initiateServiceChain`) and store it locally for reuse later on. If no connection can be found, the network will offer a basic service without QoS guarantees. The user than has to decide if he wants to start a new request with lower QoS requirements or uses the basic service.

However, the service provider might want to offer some predefined services, e.g. for VoIP or gaming, which will satisfy most users' needs for cheaper prices than connections with specific QoS guarantees. Or maybe a service provider has a cooperation agreement with several network service providers in order to offer the users an easy way to access a sufficient network connection to access the service with high QoE. These services can be requested by the user with the `getPredefinedServices` method and instantiated like the other services.

Another functionality the framework provides is to register an application service. This is only used by service providers running an `ApplicationService`. After the registration the service is made available and advertised in the network. Users can use the service ID as target for their network service requests. A possible option for this ID would be the use of URIs, as they are already well known to the users. However, it is only possible to register an application service, if the `ServiceConnector` interface is supported, which enables the network to establish a data stream and initiate an application service request from a user.

In Figure 1 there is another interface defined for a network service - the `NetworkConnector`. In theory the functionality characterized above would be enough to run a network service. However for practical reasons the `NetworkConnector` is used to peer networks tighter together. This is done by the `getProtocolStack` method which enables network services to exchange specifics of the underlying implementation of a network service. If both networks use the same protocol stack, e.g. TCP/IP, it is possible to exchange data directly on network layer and save the overhead and delay of unboxing the data to application level and than again boxing it for the transfer in the next network segment.

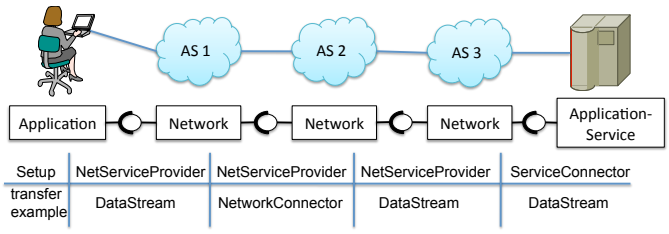

Fig. 2. Example of interface usage during the setup and data transfer of a network service composed of several virtual network connections

An example for this is given in Figure 2. A user on the left has connected to an application service on the right. The data transfer traverses three AS. As it is shown in Figure 2 the data exchange between the user and AS1 and the application service and AS3 is done on application level using data streams provided by the `DataStream` interface. AS1 and AS2 have exchanged information about their services through the `NetworkConnector` interface and found out they use the same technology and network stack. To keep it simple we assume both use TCP and IPv4. Thus, there is no need to unbox the data and then repack it into the same format. Instead the packets can be copied directly. Against this, AS3 uses a different protocol stack. Therefore the data is unboxed at the peering between AS2 and AS3 and handed over on application level. If there is no bridging device between the different autonomous systems the data stream has to be exchanged in any format both ASes have defined in their peering agreement.

At this point we are able to connect two endpoints, e.g. a user and a service, with each other. How the QoS of such a composition can be composed is not a trivial problem and will therefore be discussed in the next section. Against this, charging is much easier to implement. The costs of a connection are defined by the sum of the charging schemes for each part of the composition. In theory any forms of charging, e.g. volume, time, or flat rate based pricing schemes, can be used. Unfortunately a connection which costs x cents per megabyte + y cents per minute + z cents for setup are very hard to understand for the customer. But we assume that a customer will stick to the composition, where he can easily calculate the costs. Hence the choice of the customer on a free market will force providers to agree on some standard pricing schemes or offer different kinds of charging.

IV. Calculating QoS for Patched Network Paths

The estimation of the QoS parameters along an end-to-end path through n network segments, which could be routers or networks themselves, combines the measurements Q_i of the QoS parameter Q through all network segments $1 \leq i \leq n$. Let Q_i^+ denote the combined measurements through the first i network segments. After the next network segment $i+1$, the QoS parameter Q is observed as

$$Q_{i+1}^+ = Q_i^+ \circ Q_{i+1} = Q_1 \circ Q_2 \circ \ldots \circ Q_i \circ Q_{i+1}\,, \quad (1)$$

whereby the operator $\circ$ describes how the measurements are combined. The concrete definition of the operator depends on the actual QoS parameter. Without loss of generality it is sufficient to explain the combination of two QoS parameters due to the recursive description in Eqn. (1). This is in particular the way the measurements are combined within the service oriented network virtualization framework, since the `requestNetService` method for getting information about the QoS of an end-to-end path is also recursively implemented.

In the following, we show how to calculate the QoS for patched network paths. To be more precise, the calculation of

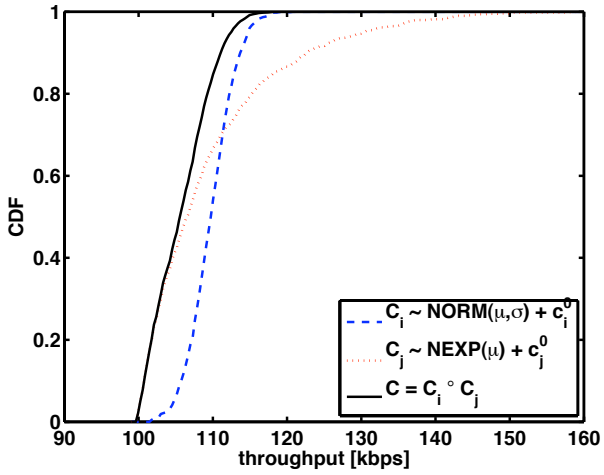

Fig. 3. Example for the estimation of throughput of two individual network segments

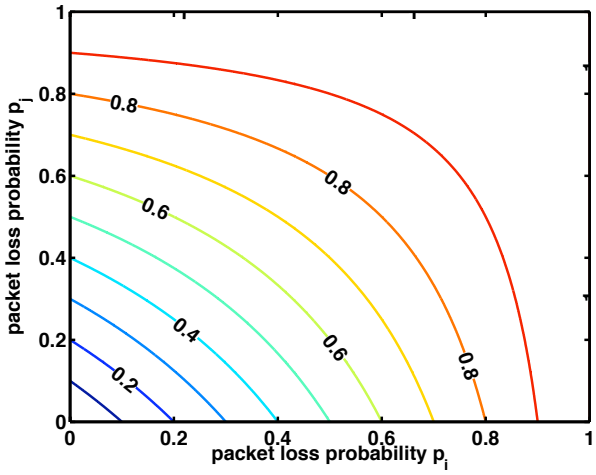

Fig. 4. Contour plot of the resulting packet loss passing along network segment i and then network segment j

bandwidth, packet loss, as well as delay and jitter is presented for two measurements Q_i and Q_j.

A. Bandwidth

We consider two network segments i and j with throughput C_i and C_j, respectively. Then, the available throughput C along i and j is simply the minimum of both, i.e.

$$C = \min\left(C_i, C_j\right). \tag{2}$$

When the available bandwidth at a network segment is measured for a certain time period Δt, the throughput C_i and C_j is described as random variables. Assuming that the throughput on the network links is independent, i.e. C_i and C_j are statistically independent random variables, the cumulative distribution function (CDF) of the throughput C is given as

$$C(x) = P\left[C \leq x\right] = 1 - \left(1 - C_i(x)\right)\left(1 - C_j(x)\right). \tag{3}$$

Figure 3 shows exemplary the CDF of the throughput C for two measured CDFs of C_i and C_j. For C_i, we assume the sum of a minimum available bandwidth $c_i^0 = 100\,\text{kbps}$ and a normal distributed random variable with mean $\mu = 10\,\text{kbps}$ and standard deviation $\sigma = 3\,\text{kbps}$. For C_j, we assume the sum of a minimum bandwidth $c_j^0 = 100\,\text{kbps}$ and a negative exponentially distributed random variable with mean $\mu = 10\,\text{kbps}$. Then, the obtained bandwidth along i and j is computed according to Eqn. (3) and plotted as black solid line in Figure 3.

B. Packet loss

The observed packet loss probabilities in the two network segments i and j are p_i and p_j, respectively. Since packets first traverse network i and then j, the resulting packet loss probability p follows as

$$p = p_i + (1 - p_i)p_j. \tag{4}$$

Figure 4 depicts a contour plot of the resulting packet loss p and prints its value according to Eqn. (4) on the corresponding contour lines, while the packet loss probability of network segment i and j are given on the x-axis and y-axis, respectively.

C. Delay and Jitter

Let T_i and T_j denote the delay through the i-th and j-th network segment. If the delays are independent, the joint distribution of the component delays can be formed using convolution. Thus, the total delay T observed along the path through i and j is

$$T = T_i + T_j, \tag{5}$$

while the probability density function (PDF) $t(x)$ as first derivation of the CDF $T(x)$, i.e. $\frac{d}{dx}T(x) = \frac{d}{dx}P\left[T <= x\right]$, is calculated as

$$t(x) = \int_{\tau=0}^{x} t_i(x)t_j(x-\tau)d\tau \tag{6}$$

with the PDFs $t_i(x)$ and $t_j(x)$ of the corresponding delays.

Figure 5 shows the PDF of the delays T_i and T_j, as well as their sum T. We assume that T_i follows a lognormal distribution with mean $\mu = 12.18\,\text{ms}$ and standard deviation $\sigma = 255.02\,\text{ms}$ plus a minimum delay $t_i^0 = 10\,\text{ms}$. For T_j, we assume the sum of a minimum delay $t_j^0 = 20\,\text{ms}$ and a pareto random variable with parameter $K = 0.3$ and $\sigma = 4$. The PDF of T is calculated according to Eqn. (6) and depicted as black solid line in Figure 5.

Although the PDF of T allows to easily derive the average delay and the jitter, e.g. as standard deviation of the delay T, its calculation faces two problems. First, the computation is too time-consuming and raises therefore challenges in practical implementation of the Networking framework. Second, independence of the delays T_i and T_j is assumed, which might not be valid in practice. The same concerns may also arise for more complex bandwidth or packet loss

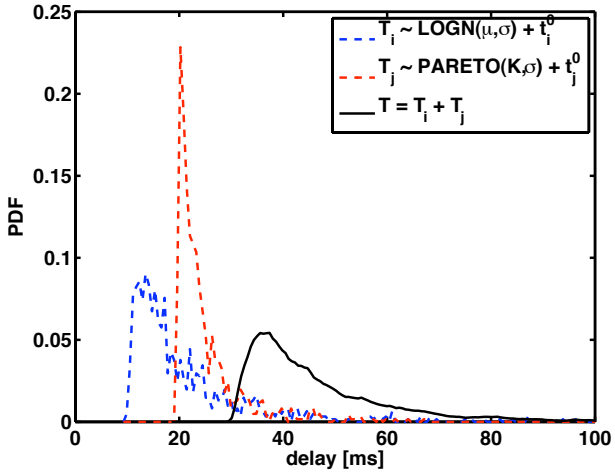

Fig. 5. Example for the estimation of delay of two individual network segments

processes. Nevertheless, in literature several approaches exist which approximate appropriate key performance measures. For example, [16] presents a simple approximation of delay-variation distributions, which is convolution and transform-free method and allows for an accurate and less time-consuming estimation of jitter.

Furthermore this calculation does not consider the delay which is caused by the handover of data between two consecutive networks. In some cases this might be just some additional fixed time span, but might also depend on the data volume and how it is transfered in the different virtual networks. A implementation of the delay distribution therefore also has to consider the handover. However, we will show in the following section that delay is a QoS parameter, which is rather easy to measure. Thus, for practical reasons it might be more efficient to estimate the end-to-end delay of a network service and measure it for compositions, which hardly meet the requirements, instead of calculating it precisely for each connection.

V. Measuring QoS

In practice, the system often does not exactly behave like expected from theory. This is especially the case for systems that use virtualization and where many virtual systems are run on the same hardware and interact with each other. As explained in [7] absolute QoS is therefore hard to maintain, also relative QoS might be achievable. Hence, for a SOA supporting QoS over virtual networks it is necessary to implement features, which can measure the current QoS of a network service.

There are different options, which can be considered for measuring the QoS of a network service. A possible method is a passive measurement, which means that at some point of the network data is collected and analyzed. The big advantage of this methodology is that the system is only observed but not affected by the measurement. However, with a passive measurement many QoS parameters are hard to assess. Although bandwidth can be computed easily, the estimation of round trip times and packet loss is only possible with protocols in which corresponding messages are sent in both directions. E.g. TCP is a protocol that can be used to estimate loss and round trip time with a passive measurement. However parameters like jitter are only accessible with deep knowledge of the timing aspects of a protocol. Against this, active measurements, which introduce traffic to probe the network, are a rather easy method to estimate the QoS of a network service at the moment of the measurement. All QoS parameters, i.e. one-way delay, jitter, packet loss and bandwidth, can be quantified by an appropriate active measurement. The big disadvantage is that the measurement directly influences the system and it is therefore not possible to estimate the QoS of the system without the additional measurement data. In our case this 'disadvantage' can also be used to analyze the effects of a new network flow on the network, if the traffic characteristic of the flow is known and emulated by the measurement. Other measurement strategies combine passive measurements at different points of the network and exchange the data. The advantage of these methods is that, although the exchanged data has only a small influence on the system, the same QoS parameters can be examined as with an active measurement. However, the exchange and correlation of corresponding data is complex and the system is not easy to maintain.

A. Measuring QoS with Cisco IP SLA Tests

In the following we focus on an active measurement, which can be done between two routers. The advantage is that a network provider does not have to deploy special measurement hardware. We examined the measurement quality of the *Cisco IP SLA UPD Jitter Test*. The IP SLA framework, which was formerly known as response time reporter (RTR), supports many test from simple ping up to response time measurements for server requests. However we focus on the UDP jitter test, as it measures all interesting QoS parameters with a single configurable bulk data transmission. The UDP Jitter measurement has the additional advantage that the two routers, which are used for a measurement, can be configured to prioritize the measurement packets. Therefore it is possible to measure the network between the routers as a black box, and the measured QoS parameters are not affected by the load on the routers performing the measurement. Furthermore, if both

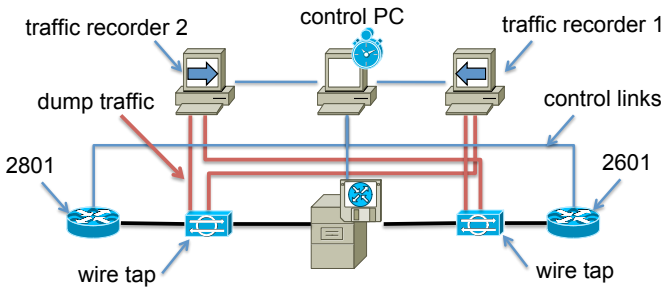

Fig. 6. Testbed setup for IP SLA measurements

customer edge routers are able to perform these kind of tests, they can be used to characterize the global QoS of a network service. The prediction for a specific network service might additionally be improved by emulating the data transfer of the service. However, we used a default option, which sends out 1000 packets of size 214 bytes with a intersent time of 20 ms.

In order to verify the quality of the Cisco IP SLA UDP Jitter test we installed a NetEM network emulator between two Cisco routers, cf. Figure 6. The router performing the measurement test was a Cisco 2801 router with Cisco IOS 12.4(9)T. As remote station we used a Cisco 2601 router with Cisco IOS 12.3(20). The IOS version of the Cisco 2601 is only supporting RTR, which is the former name of the IP SLA Tests, but it can be used on a remote host for the IP SLA measurements. In order to compare the results of the router measurement with the exact values produced by the network emulation, we installed a wire tap on both sides of the network emulator and dumped all packets exchanged between the routers. The measurements are fully automated by a control PC machine, which (1) starts tests on the router with EXPECT scripts, (2) collects test results using SNMP, (3) modifies the network emulator over ssh, (4) controls the PCs dumping data and (5) provided a stratum 2 NTP clock for the complete test bed. This enabled us to measure each combination of a network delay of 0 ms to 250 ms in steps of 25 ms and ten packet loss values between 0% and 2.1%. Each measurement scenario was repeated thirty times. We furthermore measured the influence of jitter between 0 ms and 80 ms for a delay of 200 ms and repeated the test twenty times. Correlated jitter and packet loss are evaluated in cases of 75%, 90%, 99%, and 99,9% correlation and repeated seven and twenty times, respectively.

B. Quality of Delay Measurement Results

As a result of the IP SLA test, round trip times are reported. If both routers are synchronized with a timeserver, it is also possible to measure one-way delays. Therefore we set up some additional test series, in which we adjusted the delays for both directions in the network emulator differently. From these measurements we can confirm, that the one-way delays work independent from each other and expose the same quality of results, which we measured in the other experiments described above.

For each measurement the IP SLA test reports the minimum, the mean and the maximal delay during the measurement. In Figure 7 we show the maximal error of the maximal delays reported by the test in absolute values. It can be seen that the results of the router do not overestimate the maximal delay more than 6 ms in all tested cases. The accuracy for the mean delay and the minimal delay results are even better. The difference between the delay emulated and the reported by IP SLA is 1.42 ms on average. It has to be noted that none of the test results underestimated the delay realized by the network emulator more than 2 ms. Hence, the IP SLA results have a high precision at estimating the network delay in each direction.

C. Quality of Jitter Measurements

In order to study the quality of the jitter results, we first considered jitter values between 0 ms and 80 ms with a step width of 10 ms using a one way delays of 200 ms. Figure 8 depicts the absolute error between the reported jitter values and our accurately observed jitter values. The red line in the middle of each box visualizes median, whereas the upper and lower bound of the box show the inter quartile range. The minimal and maximal reported differences are marked by the end of the vertical line. Differences more than three times the inter quartile range away from the median are considered as outliers and marked by a '+'. The measurement quality for jitter beneath 40 ms is quite good. For higher jitter values, the absolute error range increases and the relative error also increases up to about 10%. However, in practice the jitter values are mostly beneath 30 ms. If the reported jitter is higher, we can use the value as a worst-case approximation, as the

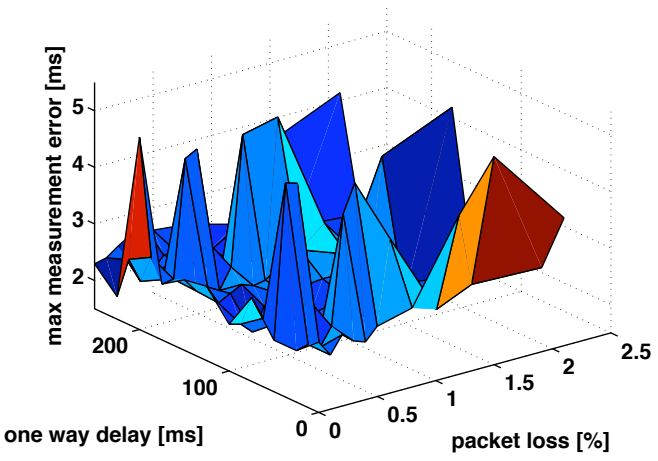

Fig. 7. Maximal error of the delay measurement

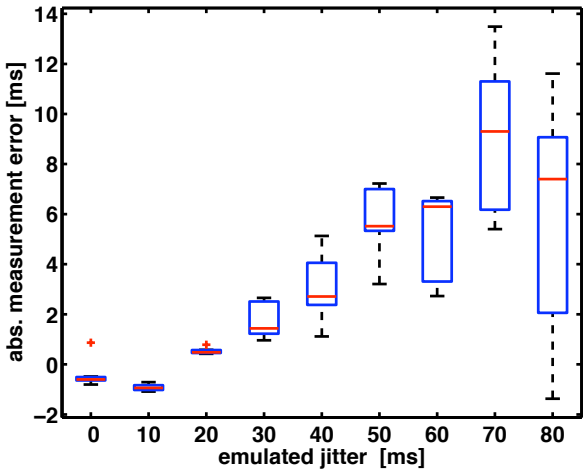

Fig. 8. Error in reported jitter when increasing the emulated jitter

reported values very rarely underestimate the jitter of the system in these cases.

In practice, another phenomenon is quite often observed. Jitter is rarely spread evenly. Sometimes there are periods of higher traffic and congestion in the network, which additionally delays the transmission to some extend. Sometimes there are periods in which the network is less loaded and the transmission delay is only slightly increased. In order to simulate such a scenario, we adjusted the NetEm PC in order to emulate correlated jitter. We consider correlation values of 75%, 90%, 99%, and 99.9% for jitter values of 5 ms, 10 ms and 20 ms. We also consider average transmission delays of 50 ms, 75 ms, 100 ms, and 150 ms.

Figure 9 depicts the influence of jitter correlation to the error of jitter results of the IP SLA test for an average transmission delay of 75 ms. The connected points for each plot depict the mean error between the jitter values, which we calculated from the time stamps recorded in the dump file for the transmitted test packets and the results the router reported. For each measurement point, the vertical line depicts the maximal and minimal difference between these values. Again, we see that the mean error increases for higher jitter values. However, the range is rather small and correlation of jitter does not effect the quality of the measurement. Hence, we can use the results of the UDP Jitter test to estimate this QoS parameter in the network.

D. Quality of Loss Measurements

Packet loss in the network influences the sending behavior of congestion aware protocols, e.g. TCP, which will adopt the transmission rate accordingly. Thus, it is hard to estimate packet loss correctly with active measurements, as TCP based connections might lower their bandwidth and therefore free bandwidth which lowers the measured loss on the link as described in [17]. This general problem of active testing packet loss has to be considered, if measurements are done in a real network.

Figure 10 depicts the packet loss reported by the UDP Jitter test for emulated packet loss values of 0.3%,1.5%, and 2.1%. The plot shows the mean values, the minimum and the maximum of reported loss by IP SLA. The number of packets that did not make their way to the foreign router and back is always correct. However, the deviation of the minimum and maximum reported packet loss value to the actual emulated packet loss value is quite large. This is a general problem, which occurs due to the actual packet loss pattern. One active measurement over a lossy link can be considered as a sample of a random process. It is therefore rather unlikely to estimate the loss probability exactly, but only within a certain range around the real value. The quality of the estimation can only be improved by increasing the number of packets sent within the test.

This basic statistical problem gets harder if we consider bulky loss patterns. As described in [17], packet loss in the Internet often occurs in short loss periods. Within such periods, a high number of packets are dropped by the routers because of congestion. Between two loss periods the router are able to deliver all packets and packets might be delayed but no packets are lost. In order to reproduce this behavior we created correlated loss patterns with our network emulator. Figure 11 visualizes the maximum of packet loss measured during our test with correlated packet loss. For a correlation of 75% and above, nearly all measurements did not detect any packet loss. Therefore the mean values are also nearly zero. It is evident that even tests with thousand packets are not enough to differentiate between 1.5% or 0.3% packet loss. Under these circumstances the measured loss only depends on how much packets are lost in such a period. However the duration and the frequency of those loss periods can not be estimated in this way. Hence, it is not feasible to estimate the packet loss by a single measurement sending one bulk of packets in practice.

For packet loss measurements relying on active bulk probing, there is a general trade-off between costs, i.e. the number

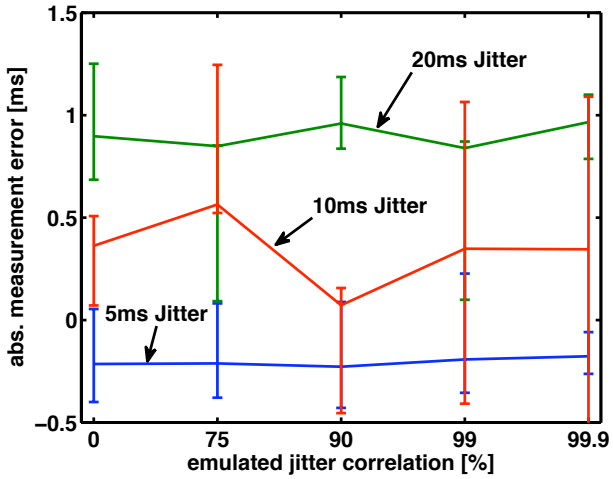

Fig. 9. Error in reported jitter when increasing the emulated jitter correlation

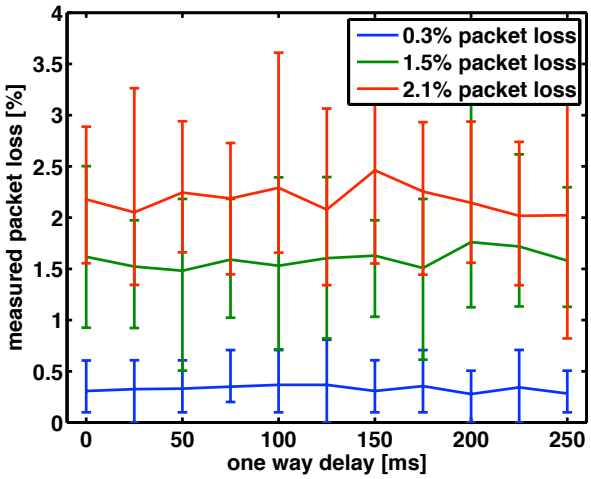

Fig. 10. Measured loss rate for increasing delay

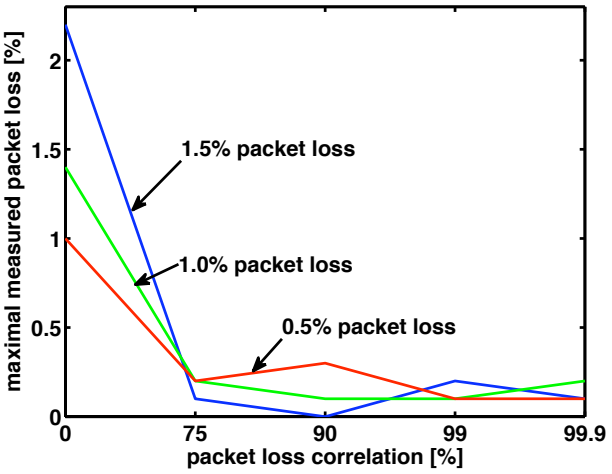

Fig. 11. Maximal measured loss for correlated loss pattern

of packets, and reliability. If a single measurement has to provide a good estimation of packet loss, there has to be a high number of packets transmitted in the test. But there is also another solution. The mean loss values of many consecutive measurements can be used to estimate the mean loss value during the measurement time. This is correct as for each test the outcome is binomially distributed with the same parameters and therefore it is possible to sum them up. Furthermore, it is reasonable to design active tests in such a way, that the quality of the other estimated QoS parameters is sufficient, and to repeat these measurements in order to update these values over time and to better estimate the mean loss during a longer time span.

VI. Conclusion and Outlook

In this paper we proposed a simple interface based on a service oriented architecture approach to provide an abstract view of the network to the application respectively the user. The approach considers QoS as the network functionality the user is mainly interested in and includes charging. We have shown how QoS guarantees for several parts of one connection can be consolidated into a QoS description for the complete service. Furthermore we discussed options to measure the QoS and presented measurements exposing the quality of an available active measurement tool, Cisco IP SLA.

In future work we want to implement a prototype to run in the G-lab (www.german-lab.org) and in PlanetLab. Hereby we will first focus on an implementation of a UDP and TCP based network service and improve the integration between the normal Internet architecture and our service approach over the `NetworkConnector` interface. With this test implementation we want to show that the delay introduced by peering data between networks on the application level is small and justified by the freedom of choosing any network implementation. Furthermore we want to get a realistic performance rating for the time needed to combine several network services to end-to-end connections and calculate the QoS guarantees.

Acknowledgements

The authors would like to thank Jochen Prokopetz for conducting the measurement studies and Prof. Tran-Gia for the fruitful discussions and his valuable comments.

References

[1] I. Rec, "G. 1030-Estimating end-to-end performance in IP networks for data applications," 2005.

[2] R. Beuran, M. Ivanovici, B. Dobinson, and P. Thompson, "Network Quality of Service Measurement System for Application Requirements Evaluation," *SIMULATION SERIES*, vol. 35, no. 4, pp. 380–387, 2003.

[3] D. De Vera, P. Rodriguez-Bocca, and G. Rubino, "QoE Monitoring Platform for Video Delivery Networks," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4786, p. 131, 2007.

[4] M. Claypool and J. Tanner, "The effects of jitter on the peceptual quality of video," in *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*. ACM New York, NY, USA, 1999, pp. 115–118.

[5] I. Rec, "G. 114: One-way transmission time," *International Telecommunications Union, February*, 1996.

[6] T. Jehaes, D. D. Vleeschauwer, T. Coppens, B. V. Doorselaer, E. Deckers, W. Naudts, K. Spruyt, and R. Smets, "Access network delay in networked games," in *NetGames '03: Proceedings of the 2nd workshop on Network and system support for games*. New York, NY, USA: ACM, 2003, pp. 63–71.

[7] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse through Virtualization," *COMPUTER*, pp. 34–41, 2005.

[8] J. Wroclawski, "RFC 2210: The use of RSVP with IETF integrated services," *Status: PROPOSED STANDARD*, 9 1997.

[9] K. Nichols, S. Blake, F. Baker, and D. Black, "RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers RFC 2474," *Internet Engineering Task FORCE (IETF)*, 1998.

[10] O. Younis and S. Fahmy, "Constraint-Based Routing in the Internet: Basic Principles and Recent Research," *IEEE Communications Surveys and Tutorials*, vol. 5, no. 1, pp. 2–13, 2003.

[11] R. Dutta, G. Rouskas, I. Baldine, A. Bragg, and D. Stevenson, "The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet," in *Communications, 2007. ICC'07. IEEE International Conference on*, 2007, pp. 1899–1904.

[12] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap: role-based architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, 2003.

[13] J. Lane and A. Nakao, "Best-Effort Network Layer Packet Reordering in Support of Multipath Overlay Packet Dispersion," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, 2008, pp. 1–6.

[14] L. Völker, D. Martin, I. E. Khayat, C. Werle, and M. Zitterbart, "An Architecture for Concurrent Future Networks," in *2nd GI/ITG KuVS Workshop on The Future Internet*. Karlsruhe, Germany: GI/ITG Kommunikation und Verteilte Systeme, Nov 2008.

[15] L. Völker, D. Martin, C. Werle, M. Zitterbart, and I. E. Khayat, "Selecting Concurrent Network Architectures at Runtime," in *IEEE International Conference on Communications (ICC 2009)*. Dresden, Germany: IEEE Computer Society, 6 2009, (to appear).

[16] R. Armolavicius, "Simple approximations of delay-variation distributions," in *18th ITC Specialist Seminar on Quality of Experience*, Karlskrona, Sweden, may 2008.

[17] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving accuracy in end-to-end packet loss measurement," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 157–168, 2005.