

# Quality of Signalling: A New Concept for Evaluating the Performance of Non-INVITE SIP Transactions

Marco Happenhofer<sup>1</sup>, Christoph Egger<sup>1</sup> and Peter Reichl<sup>2</sup>

<sup>1</sup>Institute of Broadband Communication (IBK) Vienna University of Technology, Vienna 1040 Austria

<sup>2</sup>Telecommunication Research Center Vienna (FTW), Vienna 1220 Austria

**Abstract**—Providing Quality of Service (QoS) and Quality of Experience (QoE) in connection with media data like audio or video is of key importance for the evolution of future packet-based networks. Especially in telecommunications, the subjective service quality as perceived by the end user depends amongst other parameters on the setup success rate as well as on the delay for session setup, which is measured end to end. In this paper, we introduce the new concept of “Quality of Signaling” (QoSg) and present an analysis based on Session Initiation Protocol (SIP) networks, where messages traverse usually several hops (SIP proxies) and several connections. SIP introduces the concept of transactions that are managed hop-by-hop and form the basic building blocks for SIP signaling. We argue that the end-to-end service quality strongly depends on the performance of these transactions and provide a comprehensive analytic, simulative and measurement-based performance evaluation of single SIP transactions.

**Index Terms**—Communication system performance, Discrete event simulation, Measurement, Protocols

## I. INTRODUCTION

Within the current trend from traditional circuit-switched networks towards convergent All-IP architectures, the Session Initiation Protocol (SIP) [1] has gained pivotal importance and has been selected by several standardization bodies (e.g., 3GPP, ITU-T and ETSI) as signaling protocol for their Next Generation Networks (NGN). This paper discusses some key prerequisites for efficient NGN design by proposing a new approach for SIP performance evaluation based on SIP transactions as basic building blocks of this signaling protocol.

SIP performance has been subject to intense research over the last couple of years, most of the related work, however, focuses on end-to-end performance analysis. For example, [3] presents a generic end-to-end performance metric without taking the intermediate network into account. Such analyses give an idea how the end-to-end performance looks like for particular architectures. The authors of [6] measure the call setup delay between certain cities in the US. The authors of [4] model a call setup with queuing theory, but ignore the fact that the messages might get lost. Other research discusses the different transport protocols of a specific SIP topology [5][7]. All these publications give precise information how the performance can be increased by modifying certain parameters, but these guidelines are only valid for their SIP topology.

The aim of our research is to understand the impact of network parameters, which can be described by the IP Perfor-

mance Metrics (IPPM) [13], on the performance of SIP transactions for arbitrary SIP network topologies. We start to analyze the smallest building block in SIP signaling, the so-called SIP transaction. Later on, we will use these insights to model large SIP network topologies and compare our results with measurements.

In order to describe the performance of a single transaction, we introduce the term *Quality of Signaling* (QoSg) as a new concept describing the corresponding metric. Because QoSg corresponds to the performance of a single transaction, by combining these values we can reproduce almost any SIP topology. Furthermore this metric can be used to measure the performance between administrative SIP domains or between SIP components within the same administrative domain.

Service providers that interconnect with each other want to guarantee that service requests from their domain are handled in a justifiable time in the foreign domain. By observing the QoSg values, both providers can verify their Service Level Agreements (SLA). On the other hand a service provider wants to observe his equipment and identify components that are not working properly. QoSg can also be used to achieve this. A significant irregularity in some values can be an indicator for a malfunction in the observed component. However, for such an analysis we need to understand the nature of SIP.

The remainder of this paper is organized as follows: Section II introduces SIP, our metric and the model we used for evaluation with all our assumptions. Sections III, IV and V present three techniques for performance evaluation, i.e. analytic modeling, measurement and simulation. All techniques are presented and their main aspects are described. Section VI shows the results of the analytic model, which are compared to measurement and simulation results in section VII. The last section draws some conclusions and provides an outlook on future work.

## II. BACKGROUND

### A. Session Initiation Protocol (SIP)

The Session Initiation Protocol [1] is a signaling protocol specified by the IETF, used for setting up, modifying and tearing down multimedia sessions. It is a text-based protocol, and uses so called transactions to keep track of the state of the signaling. Each transaction combines request and corresponding response messages. SIP specifies two different transaction types: on the one hand the INVITE transaction, and on the

other hand the Non-INVITE transaction. This differentiation is necessary to respect the delay due to human interaction in the INVITE transaction.

The Non-INVITE transactions are introduced to realize tasks like registration, exchange of capabilities of the SIP user agents, exchange of presence information, and instant message delivery. Additionally, these transactions are used within a call to update the session parameters before a call is established (UPDATE method) or to terminate a call (BYE method).

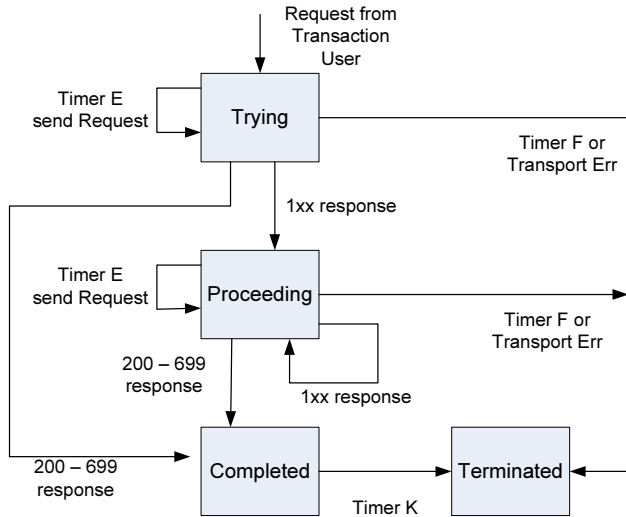


Fig. 1. Non-INVITE Client Transaction

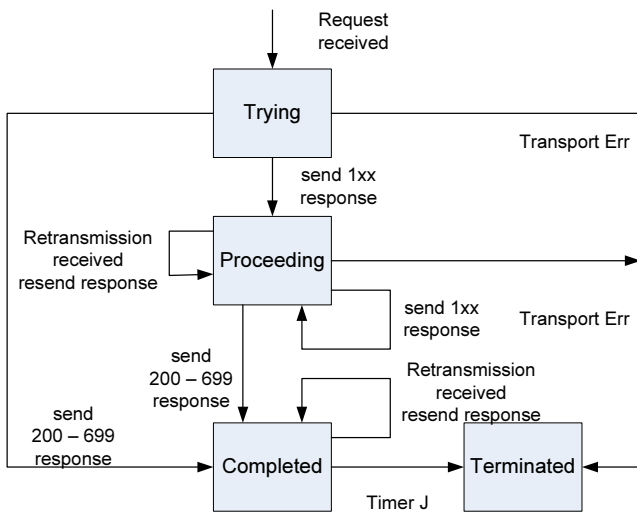


Fig. 2. Non-INVITE Server Transaction

Because SIP allows the usage of different transport protocols, as User Datagram Protocol (UDP) [15], Transmission Control Protocol (TCP) [14] and the Stream Control Transport Protocol (SCTP) [16][17], a common transaction layer is specified, which realizes a reliable and transport protocol independent message delivery. The specification defines a state machine for the client transaction, which is hosted by the sending entity, and for the server transaction, hosted by the receiving entity. Since we concentrate on the Non-INVITE transaction in

this paper, we depict only the state machines of the Non-INVITE transaction in Figures 1 and 2.

UDP is an unreliable transport protocol, so losses are not detected and not recovered. It is the task of the application to ensure reliable transport of the messages. This is done in SIP with the Timer E, which forces retransmissions. Reliable protocols as TCP and SCTP do not need retransmission by the application, so with these transport protocols the Timer E is not applied.

### B. Quality of Signaling

Users expect a constant and good Quality of Experience (QoE). This experience is mostly defined and analyzed for media as voice or video. However, the user also perceives the session setup delay. As mentioned above, the IETF currently defines a metric for SIP [3], but it only takes the end-to-end performance into account. Our approach is to define a set of metrics for a single transaction which we call “Quality of Signaling” (QoS<sub>g</sub>).

Figure 3 illustrates the message flow between UAC and UAS. On the left and right-hand side timestamps are depicted, which represent an event of sending or receiving a message.

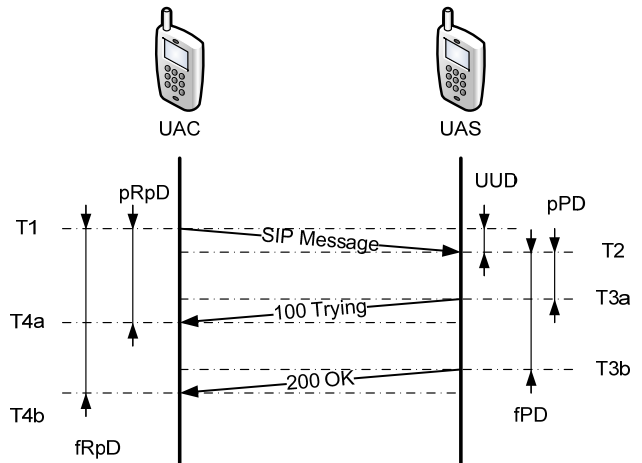


Fig. 3. Message Flow of a Non-INVITE Transaction

In our metric of QoS<sub>g</sub> for the Non-INVITE transaction we define the following terms:

- User to User Delay (UUD) represents the time interval from sending a request until it arrives at the destination. Note that this metric is equal to the one-way delay if and only if no losses occur.
- Processing Delay (PD) represents the time the UAS needs to process the request and to send a response message.
- Response Delay (RpD) is the time a user agent client waits from sending a request until it receives a response message.

Additionally we define the metric of successful transactions as Success Rate (SR). We are further interested in the traffic that this transaction caused, so we introduce the metric RT to count the number of sent requests and RpT for the number of sent responses. We also differentiate the metrics PD, RpD and RpT for final and provisional responses. In this paper we follow the recommendation in [2] and only look at the case where the UAC does not send provisional responses.

### C. Loss Model

The analysis of QoS focuses on single transactions and not on end-to-end performance. Such a single transaction can be influenced by several external parameters. For the evaluation we make some assumptions to reduce the complexity.

We use the term Loss Model, for a model considering only the loss rate between the UAC and UAS and vice versa. Our assumptions are the following:

- There are no proxies between UAC and UAS.
- Processing time on the server (PD) is close to zero (includes also queuing delay).
- Uplink (UAC to UAS) and downlink (UAS to UAC) have no delay.
- The server does not use provisional responses.
- The loss between the user agents is statistically independent. (We do not provide any proof or test that this assumption holds in practice.)

We use the Loss Model to discuss the effects of the loss rate for the QoS metric. In further research we also consider a Delay Model where we assume a loss rate of zero on the links between UAS and UAC but a varying delay and a Delay Loss Model where we vary both. This paper exclusively discusses the Loss Model. In all following discussions, we always assume an unreliable transport like, e.g., UDP.

### III. ANALYTICAL MODEL

In this section we derive the performance values of the transaction by means of discrete Markov chains. To realize this task we use the state machines of the client and server transactions and combine them to a single state machine that maps the state of both transactions. Due to our assumptions made in the previous section we can reduce the state machine to three states. The first state ( $F$ ) represents the case where the client wants to send a request and the server has not received it until now. The second state ( $I$ ) represents the case where the server has received the request, sent a response but the client has not received it yet. And the last state ( $S$ ) assumes that the response has been received by the client and so the transaction has terminated successfully.

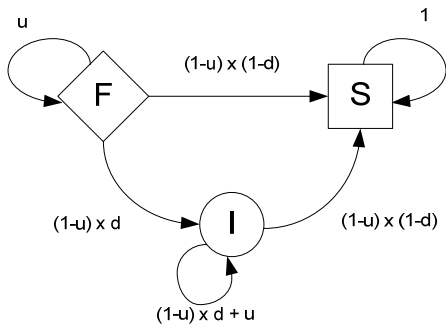


Fig. 4. State Diagram Model 1

Figure 4 illustrates the Markov chain for the Loss Model. The value  $u$  represents the loss rate in the uplink direction (UAC to UAS) and the  $d$  value for the loss rate in the downlink direction (UAS to UAC).

Equation (1) depicts the transition matrix derived from the state diagram above. To derive performance values based on this Markov chain, we use  $\{1,0,0\}$  as initial state vector  $\{F,I,S\}$ . Before any request is sent the probability that a request has not arrived at the server is 1. After sending the first request this vector changes to  $\{u,(1-u)*d,(1-u)*(1-d)\}$ . This reflects the multiplication of the state vector with the transition matrix. Further sent requests will result in further multiplications. We assume default timer configurations, as stated within [1], and thus 11 transmissions of a request until timer F fires. So we can stop the calculations after 11 multiplications and conclude that the success rate SR is equivalent to the value of state S.

$$\begin{Bmatrix} u & (1-u)*d & (1-u)*(1-d) \\ 0 & (1-u)*d+u & (1-u)*(1-d) \\ 0 & 0 & 1 \end{Bmatrix} \quad (1)$$

We can derive further performance parameters of the transaction, e.g., the number of sent requests by observing the  $S$  state over the request transmissions. If for example the  $S$  state is 50% after the 9th transmission and 60% at the 10th transmission, we can conclude that a transaction will need 10 transmissions with a probability of 10%.

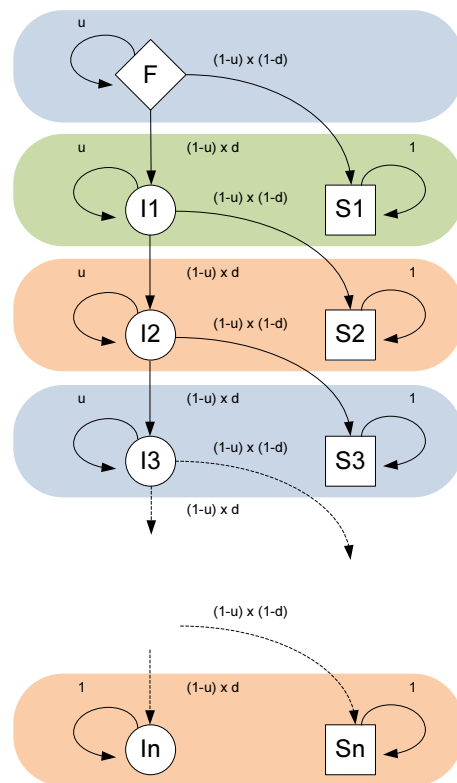


Fig. 5. State Diagram Model 2

To analyze the delay values we describe shortly the retransmission timers of SIP. A retransmission of a request is sent when no response arrived after timer E has fired. This timer is initially set to the same value as timer T1 (Default: 500 ms). Timer E will be doubled each time it fires with a limit of

timer T2, which is by default 4 seconds. After exactly  $64 \cdot T1$  the transaction terminates unsuccessfully. For the default timer configuration we can calculate the following timestamps for request transmissions (2).

$$\{0.0, 0.5, 1.5, 3.5, 7.5, 11.5, 15.5, 19.5, 23.5, 27.5, 31.5\} \quad (2)$$

Our UUD and RpD values must be equal to one of those values, because we made the assumption in the Loss Model that there are no transmission and processing delays. The number of request transmissions denotes which value has to be used. For example if a transaction is successfully finished as described above with exactly 10 transmissions with the probability of 10%, we have to select the 10th value in the vector of timestamps and conclude that the RpD value is exactly 27.5 with the probability of 10%. The estimation of the UUD is similar, whereas the probability of state  $I$  has to be added to the probability of state  $S$ .

Unfortunately we are not able to derive the number of sent responses (RpT) within the above model, so we extended it to the model shown in Figure 5.

The states  $I_x$  and  $S_x$  represent the case where exactly  $x$  responses were sent. With this model we are able to derive all metrics we presented before.

#### IV. MEASUREMENT SETUP

The evaluation of SIP transaction performance by means of measurements consists of two steps, the measurement itself and the corresponding analysis of the measurement results.

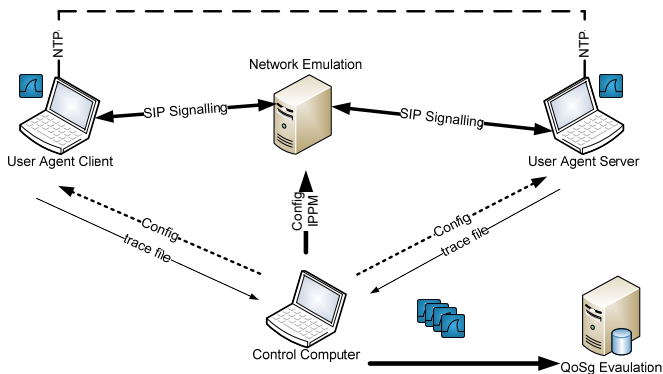


Fig. 6. Measurement Architecture

The first step is to set up an environment for SIP communications. We use SIPp [10] as UAC and UAS. SIPp is a command line tool designed for SIP load generation. An XML file describes which messages have to be sent and which messages to be sent after receiving a request. To realize loss we use an intermediate computer that runs on Linux and is equipped with two interface cards. To allow passing of the packets we configure it as bridge using the Linux tool bridge control (brctl) [11]. We use traffic control (tc) [9] to generate artificial loss. Tc allows configuring the queuing on the interface cards. So we can emulate a connection with certain loss rates, for both directions. We further need mechanisms to measure the performance values. For that purpose we use the logging of SIPp and we also use tcpdump [12]. This has the advantage that we obtain events on the application layer with SIPp, and on the

other hand on the transport layer by means of tcpdump. For the evaluation described in this paper, where we consider only UDP, this makes no difference, which is not the case for other transport protocols. This is because the congestion control mechanism of e.g., TCP will delay some segments when a loss is detected. In this case the application will indicate that a packet was sent, but it is only queued in the outgoing queue of the transport protocol.

Because we want to measure the timing of the signaling, we need to ensure that all participating computers are time synchronized. We use a dedicated NTP network to realize a minimum time offset.

We have measured the performance of different loss rates from 0% up to 98% in 2% steps for each direction leading to 2,500 measurements. We developed a shell script which automatically configures the user agents, sets the loss rate at the bridge, starts tcpdump and additionally the SIPp application on both sides.

Figure 6 depicts our measurement architecture. On the left and right hand side are the user agents and in the middle the bridge that emulates the loss. The control computer executes the script for automatic measurements.

The second step is to extract performance values from the log and trace files. For this purpose we created a Java application that gets as input the SIPp log files and the tcpdump trace files from the client and server. It is possible to get the performance parameter for each single transaction as well as statistic values. The output is formatted as an XML file.

The application parses the log and trace files and extracts the SIP messages. Afterwards it maps the logs and traces of the UAC and UAS to single transactions by comparing Call-ID and CSeq headers. A single Java object stores all messages logged by SIPp and tcpdump for a single transaction. Out of this object we derive all required performance values.

The output is an XML file that stores statistic values for each measurement. To get a format that can be used by gnuplot we use xalan as XSLT tool.

Summarizing our measurement setup, we use SIPp to generate SIP MESSAGE requests (Non-INVITE transactions) as SIP traffic, where the size of a single request is about 388 bytes and the size of the response about 268 bytes. Each measurement includes 5,000 transactions. We analyzed the loss rates from 0% up to 98% for each direction and measured so 2,500 measurement configurations and so in sum 12.5 million transactions.

#### V. SIMULATION

We use the simulation environment IBKSIM, developed at the Institute of Broadband Communications of TU Vienna (www.ibk.tuwien.ac.at), and described in [8]. We have implemented UAS and UAC, including server and client transactions for the Non-INVITE transaction according to [1]. The UAS is connected to the UAC by means of a link implementation with stochastic, variable loss rate from zero to 100 percent of the messages. It is able to vary the loss rate of uplink- and downlink direction independently. The UAC automatically sends

SIP MESSAGE requests (Non-INVITE transaction) over the link to the UAS. The UAS replies immediately with a response message. Due to the losses on the link, a message (request or response) can get lost. A logging module traces the events of receiving and sending of messages and calculates the performance values. We run a single simulation for 100 seconds and send 25 messages per second what leads to 2,500 requests. We vary the loss rate of the link from 0% to 99% in 1% steps in uplink as well as downlink direction. This leads to 10,000 simulation runs.

#### A. User Agent Client

Our implementation has several configuration parameters, including SIP timers, the amount of messages to be sent over the link per second, as well as the start and stop time. For our analysis, we use default SIP timer values, as described in [1]. We start the sending at the beginning of the simulation with a value of 25 messages per second. To be as realistic as possible, and to avoid synchronization effects, the sending rate is a stochastic value, using a negative exponential distribution. The UAC also hosts the client transaction, which is implemented as specified in [1].

#### B. User Agent Server

In our environment, the UAS accepts all received messages and commits the reception with 200 OK. Configurable are here the values for the SIP timers. To avoid any side-effects we omit a processing time (PD=0). The User Agent Server hosts the server transaction, and also follows the recommendations of [2] to not send any provisional responses.

#### C. Link

Our link implementation has configurable delay and loss values. According to the Loss Model there is no delay on the link. We vary the loss value at uplink and downlink direction from 0 to 99% respectively. 100% would make no sense, because the metrics we use are partly not applicable for this value, e.g., it would make no sense to calculate a final response delay, when the request always gets lost.

### VI. RESULTS OF THE ANALYTICAL MODEL

In this section we present the results of the analytic model which will be compared to our simulation and measurement results afterwards.

Figure 7 depicts the success rate of the transactions, which decreases with an increasing loss rate in each direction symmetrically. It follows that loss in the uplink direction has the same impact to the success rate as loss in the downlink direction.

For the values of RT, RpT, RpD and UUD we use the 90% percentile, i.e. 90% of all transactions are below the values of the curves. We decided to illustrate the 90% percentile, because it visualize much better the retransmission by means of steps in the diagram as the mean value, which would be smoother.

Figure 8 depicts the number of sent requests, which correlates with the loss rate. The larger the loss rate is in each direction, the larger is the number of sent requests. The number of sent response messages is illustrated in Figure 9. This number correlates with the loss rate in the downlink direction and correlates negatively with the loss rate in uplink direction. This

behavior comes from the fact that the server transaction resends a response of course only, if a retransmission of a request was received.

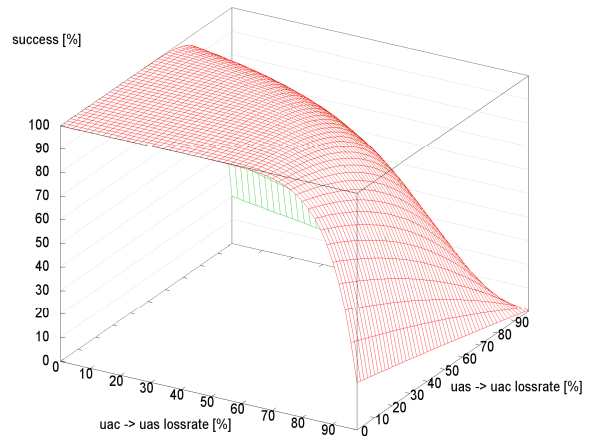


Fig. 7. Success Rate (SR)

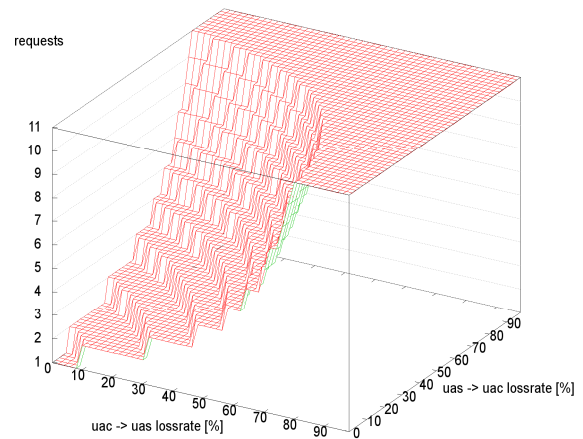


Fig. 8. Requests Transmits (RT) (90% percentile)

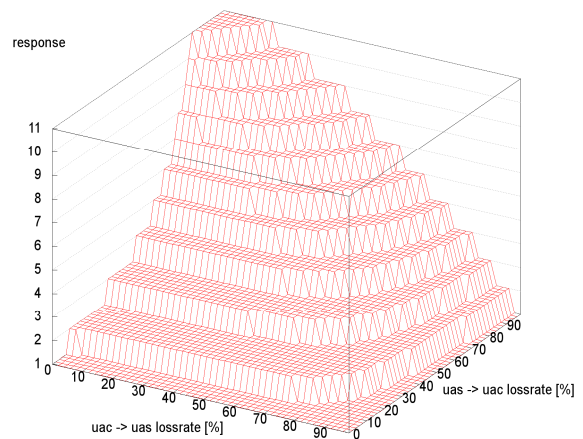


Fig. 9. Response Transmits (RpT) (90% percentile)

The RpD, shown in Figure 10, depends on the number of request transmits what can be seen in the similar shape to Figure 8. The difference is that the RpD is of course not applicable for failed transactions.

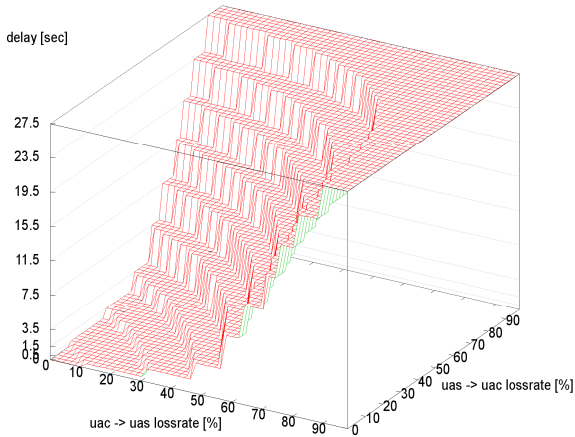


Fig. 10. Response Delay (RpD) (90% percentile)

Figure 11 shows that the delay between UAC and UAS depends only on the loss rate in uplink direction.

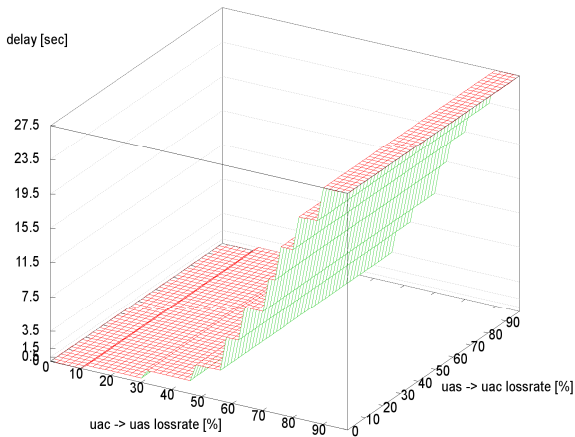


Fig. 11. User to User Delay (UUD) (90% percentile)

## VII. COMPARISON WITH MEASUREMENT AND SIMULATION

In this section we show that all three models generate the same performance values. We use the Kolmogorov-Smirnov (KS) test [1] to verify the values.

In the last section we used for the visualization of the analytic model the 90% percentile of the performance values (RpD, UUD, RT and RpT). Now we want to compare their distributions with those of the measurements. We explain the KS test for the RpD of the analytic and measurement model. Afterwards we present the results of the KS test for the other performance parameters for these two methodologies. At the

end of this section we further show exemplarily for RpD the result of the comparison between simulation and analytic model.

We use as null hypothesis that the probability distribution of the analytic model is equal to the probability distribution of the measurement. The alternative hypothesis is that these probability distributions are not equal.

$$\begin{aligned} H_0 &: F_{analytic}(x) = F_{measurement}(x) \\ H_1 &: F_{analytic}(x) \neq F_{measurement}(x) \end{aligned} \quad (3)$$

In order to calculate the  $d_{max}$  value, which presents the maximum difference between both cumulative distribution functions, we have to prepare the distribution function of the measurement. Because of the assumptions we made for our loss model (no delivery and processing delays), the distribution function of the analytic model has only 11 different delay values which are exactly the timestamps for the request transmissions (2). In the measurement environment, this assumption is not perfectly realized; hence we rounded all measured values down to the next lower transmission time value. By modifying these delay values we also modify the cumulative distribution function, figure 12 shows the extent of the rounding and that 90% are within 10 ms.

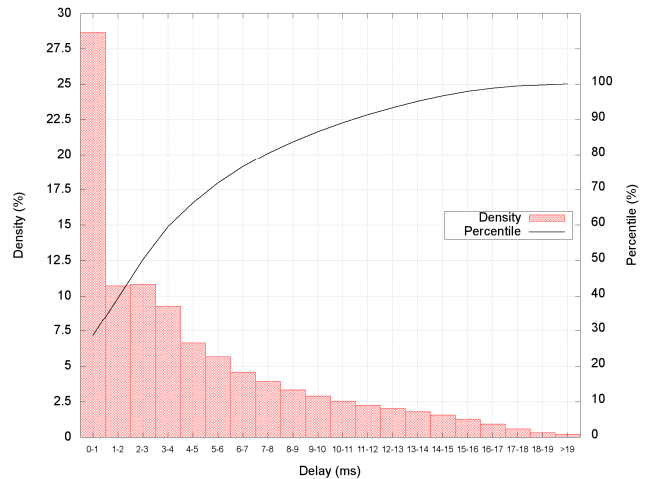


Fig. 12. Response Delay (RpD) difference between measured values and transmission times

An explanation for those differences is probably the queuing delay on the interfaces and at the application layer. Figure 13 depicts the minimum, mean and maximum value for the difference between measured value and transmission time depending on the loss rate. We assume that queuing and processing cause the delay, because its value increases by increasing loss rate and further the number of messages in the system (see figure 8 and 9). Thus we argue that the modifications we have done on our cumulative distribution function are negligible, because it corrects queuing and processing side effects.

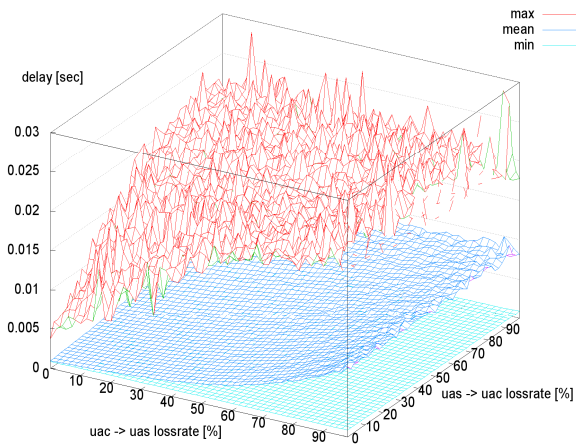


Fig. 13. Response Delay (RpD) difference between measured values and transmission times vs. loss rates

Due to rounding of the measured delay values we get a distribution function with exactly 11 delay values as the distribution function for the analytic model. The largest difference between these two distributions ( $d_{\max}$ ) is used by the KS test. If this value is larger than a critical value, the null hypothesis is rejected. Selecting the significance level and the number of samples derives this critical value.

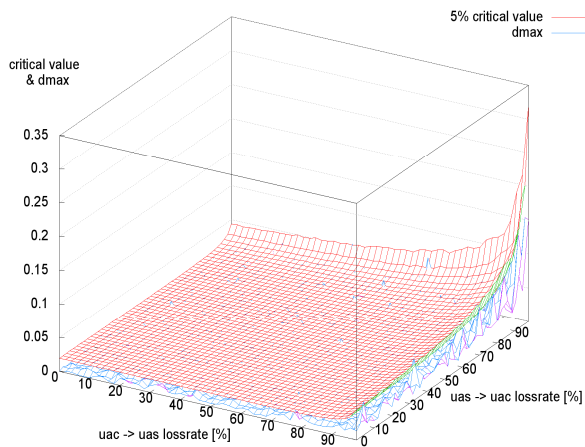


Fig. 14. KS test for Response Delay (RpD) of analytic model and measurement

We decided to use a 5% significance level as proposed in [1]. This level describes the probability, that a single test rejects the null hypothesis, even though the null hypothesis is valid. That means for our test that up to 5% of our entire KS tests may fail, however, the null hypothesis stays valid.

The results of the KS test of the measurement and analytic model for the performance values UUD, RT and RpT are illustrated in figure 14 to 17.

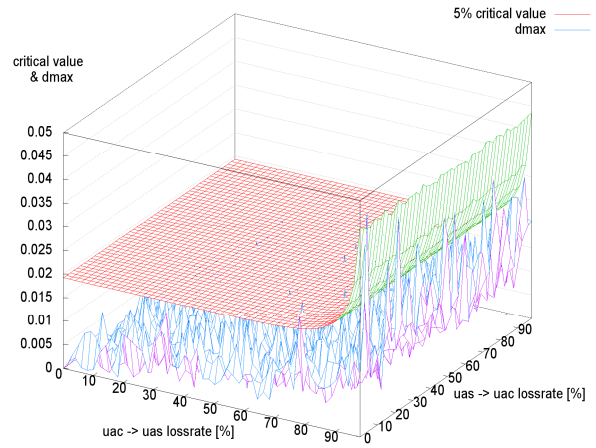


Fig. 15. KS test for User to User Delay (UUD) of analytic model and measurement

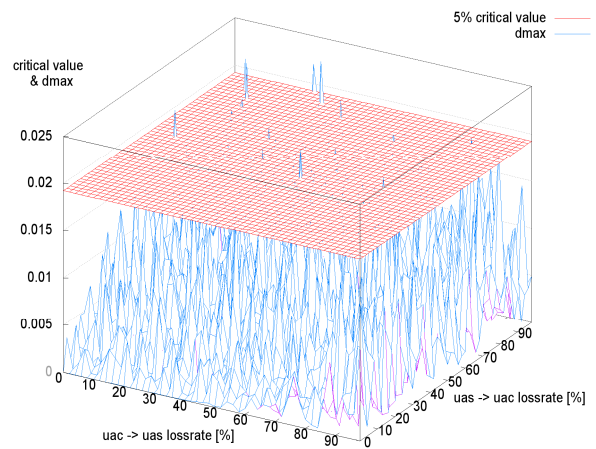


Fig. 16. KS test for Response Transmits (RpT) of analytic model and measurement

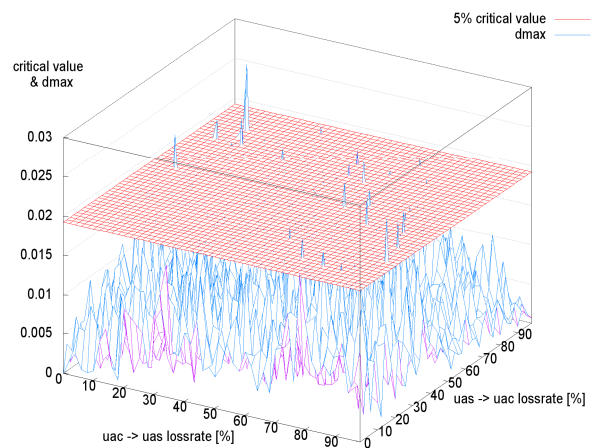


Fig. 17. KS test for Request Transmits (RT) of analytic model and measurement

In each of the four diagrams, the majority of the  $d_{\max}$  values are below the 5% critical value. Table II depicts the percentage of these rejected KS tests.

TABLE II  
PERCENT OF THE REJECTED KS TESTS

value	percent of rejects
response delay (RpD)	1.00%
user to user delay (UUD)	0.68%
request transmits (RT)	1.20%
response transmits (RpT)	0.64%

It shows that the KS test for all parameters does not reach the value of 5%. Thus, we conclude that the null hypothesis for all parameters cannot be rejected.

Additionally we present the results of the KS test for the response delay (RpD) done with the analytic model and the simulation, illustrated in Figure 18. 1.56% of all KS tests failed and so we can also not reject the null hypotheses for this test.

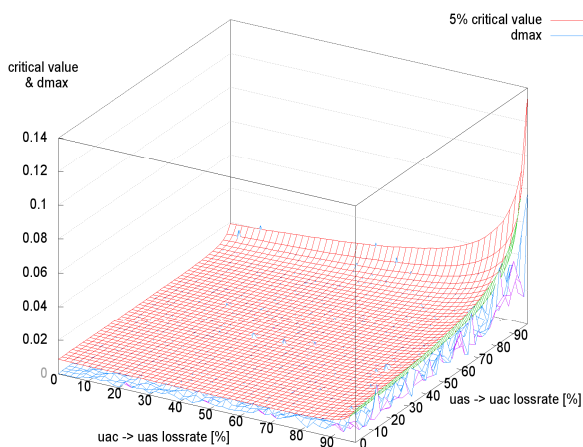


Fig. 18 KS test for response delay (RpD) between analytic model and simulation

Finally we compare the results of the simulation and the results of measurements, where only 0.3% of all KS tests were rejected. So we conclude that all three approaches lead to equal results.

## VIII. CONCLUSIONS

In contrast to most other research in SIP performance this paper focuses on the analysis of a single transaction instead of end-to-end performance. We present three methods to get from IPPM values to QoS performance values. We intend to use these values to calculate the end-to-end performance values for a certain SIP network topology. The QoS metric can be used between two SIP proxies, to observe the performance of SIP signaling. By observing this metric, the operator of the SIP network can identify components that do not work properly. This helps to satisfy SLAs and to avoid service outage due to faulty components.

We present an analytic model and a simulation to evaluate the performance of the Non-INVITE transaction (using a non-reliable transport protocol). Both models produce almost the same results and are also comparable to the measurements. We use the Kolmogorov-Smirnov test to verify the equality of all three models.

This paper covers only a small part of the possible research that can be done to be able to use QoS parameters for SLA or monitoring purposes. Our current research analyses other transport protocols, more realistic models as e.g., a delay loss model as well as introduce processing delays at proxy or UAS side. Additionally the INVITE transaction has to be analyzed in a similar manner. Finally we want to evaluate transient behavior of SIP through several proxies, using simulation.

## REFERENCES

- [1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol," IETF, RFC 3261, June 2002.
- [2] R. Spark, "Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction," IETF, RFC 4320, January 2006.
- [3] D. Malas, A. Morton, "SIP End-to-End Performance Metrics," IETF, draft-ietf-pmol-sip-perf-metric-04, September 2009, unpublished.
- [4] Vijay K. Gurbani, Lalita J. Jagadeesan and Veena B. Mendiratta, "Characterizing Session Initiation Protocol (SIP) Network Performance and Reliability," Lecture Notes in Computer Science, vol 3694/2005, pp. 196-211, October 2005.
- [5] M. Otha, "Performance comparisons of transport protocols for session initiation protocol signaling," 4<sup>th</sup> International Telecommunication Networking workshop on QoS in Multiservice IP Networks. IT NEWS 2008, pp. 148-153, 2008.
- [6] Tony Evers, Henning Schulzrinne, "Predicting Internet Telephony Call Setup Delay," in IPTTEL 2000 (First IP Telephony Workshop), 2000.
- [7] G. Camarillo, R. Kantola, H. Schulzrinne, "Evaluation of transport protocols for the session initiation protocol," IEEE network ISSN 0890-8044, 2003, vol. 17, no5, pp. 40-46.
- [8] L. Wallentin, M. Happenhofer, C. Egger, J. Fabini "XML meets Simulation: Concepts and Architecture of the IBKSim Network Simulator," Simulation News Europe SNE, unpublished.
- [9] Linux Advanced Routing & Traffic Control, URL: <http://lartc.org/> (November 2009).
- [10] SIPP, URL: <http://sipp.sourceforge.net/> (November 2009).
- [11] The Linux Foundation, <http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge>, (November 2009).
- [12] TCPDUMP/LIBCAP public repository, URL: <http://www.tcpdump.org/>, (November 2009).
- [13] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics," IETF, RFC 2330, May 1998.
- [14] J. Postel, "Transmission Control Protocol," IETF, RFC 793, September 1981.
- [15] J. Postel, "User Datagram Protocol," IETF, RFC 768, August 1980.
- [16] R. Stewart, "Stream Control Transmission Protocol," IETF, RFC 4960, September 2007.
- [17] J. Rosenberg, H. Schulzrinne, G. Camarillo, "The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)," IETF, RFC 4168, October 2005.
- [18] J. Hartung, B. Elpelt, K. Klösner, "Statistik", Oldenburg Wissenschaftsverlag, 2002, ISBN 3-486-25905-9.