Monitoring of SIP-Based Communication

Using Signalling Information for Performance Measurements

Thomas Lindh[1], Emma Roos[2]
[1] *Laboratory for Communication Networks, School of Electrical Engineering*
[2] *School of Technology and Health*
Royal Institute of Technology, KTH, Stockholm, Sweden
Thomas.Lindh@syd.kth.se, hdm04ero@syd.kth.se

Abstract

This paper presents a prototype implementation of end-to-end monitoring of performance parameters in SIP-based communication. The approach is to integrate signalling information and measurements of user data traffic. Test measurements illustrate some results that can be obtained per session; packet loss, round-trip delays and their variation, inter-arrival jitter and throughput.

1. Introduction

The topic of this paper is end-to-end monitoring of SIP-based communication. The approach is to integrate the control plane (signalling) and the user data plane, i.e. to take advantage of the signalling information in SIP messages to measure performance parameters in data traffic between the endpoint users.

Monitoring agents at the user clients measure packet loss, round-trip time, inter-arrival jitter and throughput. The prerequisite is that no dedicated measurement equipment is deployment at the end users. This means that high precision synchronized clocks are not required. Two different solutions are discussed, one where the measurements are managed from the SIP server side, and another where the user clients manage the monitoring activities themselves.

There are several approaches to monitoring of performance parameters in the user traffic set up by signalling system (SIP, H.323 etc.). Active testing between dedicated probes is one common solution among the current products on the market. Another is to collect the reports made by implementations of RTCP (Real-time control protocol) [1]. RTCP-XR is an extension to RTCP which makes it possible to send the reports to a common server for processing [2]. This solution is however limited to RTP traffic.

The approach in this paper is instead to use a more general-purpose traffic meter that is triggered by the signalling information when the calls are set up. A similar idea has previously been proposed for measurements in ATM networks [7].

The paper is organised as follows. In Section II the relevant SIP signalling information is explained. The NeTraMet flow meter agents used at the endpoints are described in Section III. The system solutions and implementations are presented in Section IV. Examples of measurement results in Section V illustrate the prototype implementation. Discussion of results and the conclusions are found in the two final sections.

2. Monitoring of SIP-Based Communication

Session initiation protocol (SIP) establishes IP telephony calls and multimedia sessions [3]. The approach in this paper is to use the signalling information between user clients and SIP servers to configure and trigger performance measurements of user traffic between the endpoints.

A typical sequence of SIP messages between a user agent and a SIP proxy server is shown in Figure 1. The calling party sends an INVITE message to the SIP proxy server (or directly to the called party if a proxy server is not used). The IP address and user name of the calling user agent are found in the Contact field (sip:username@ip-address:sip-port) and a call identification number in the Call-ID field. The SDP part (session description protocol) of the INVITE message contains the UDP port number that will be used by the calling user agent for voice and various media information. If the called party is detected and accessible, a 180-RINGING message is sent. It conveys the IP address of the called party. The 200 OK message, sent when the call is answered, contains the IP address and port number used by the called party for data transport. A BYE message indicates that the call is terminated. The BYE message from a user passes the SIP proxy server if configured as “record route”; otherwise the terminating messages are sent directly between the endpoints and are never seen by the proxy servers.

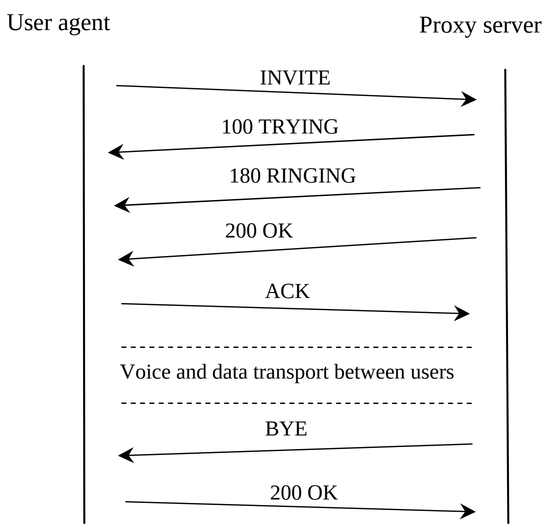

Figure 1. SIP messages for a typical call session.

3. Flow Meter Agents – NeTraMet OAM Version

The traffic flow meter NeTraMet has been used to measure the traffic at the endpoint users [4]. Traditionally a flow meter counts bytes and packets belonging to the defined flow at a single point. In previous work a multipoint traffic flow performance meter has been implemented as an extension to NeTraMet (OAM version) [5].

3.1. Combining Active and Passive Methods

The OAM version of NeTraMet was developed as a realisation of a method for a traffic flow performance meter that combines active and passive methods ([6] and [7]). It is designed as a combination of traffic flow meters and dedicated monitoring packets to obtain these goals. The size of a monitoring block determines the precision and resolution of the delay estimates, and the resolution of the loss and throughput metrics. The block size can be expressed in terms of time periods or number of packets or bytes. The packet filter specification controls the granularity of the traffic flows that are being monitored.

The method consists of three main components: packet filters to specify the traffic flows, traffic meters to count the number of packets and bytes, and monitoring packets that activate storage of intermediate values of the meters and provide samples of packet delays. These monitoring packets are inserted periodically or randomly between blocks of data packets as indicated in Figure 2.

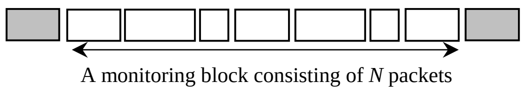

Figure 2. Two monitoring packets enclose a monitoring block that consists of N user packets on average.

The precision, accuracy and resolution of the metrics are determined by the size of the monitoring block, which is the distance between two consecutive monitoring packets (expressed in time intervals or in number of data packets). This is the main parameter of the monitoring method used to adjust the accuracy of the results. The method gives the following results per traffic flow (defined as a unique combination of destination and source addresses and port numbers, transport protocol and possibly other parameters).

- Packet delays.
  Estimates of packets delays and delay variation based on samples are obtained. One-way delays can be measured if the meters have synchronized clocks, otherwise round-trip times are measured.
- Packet loss.
  Beside the long-term averages for the entire measurement periods, the distribution of losses per monitoring block is obtained. The length of loss periods and loss-free periods can be computed.
- Throughput.
  Utilized capacity for the entire measurement period and per monitoring block are obtained.

Packet loss is measured exactly (due to the passive technique) while the delay estimates are based on sampling using monitoring packets. The statistical issues related to the distribution of the length of the loss periods and loss-free periods are analysed further in [7].

4. NeTraMet

The meter is controlled by a set of rules written in a high-level language, the simple ruleset language (srl), which defines the flows to be measured. NeMaC manages the meters remotely and collects the flow data from the meters [8]. As distributed, NeTraMet meters traffic flows, where a flow is defined as a set of packets having specified values of their address attributes. To configure NeTraMet one writes a ruleset specifying which flows to meter, and which attribute values are required for each flow. The meter is modified by Nevil Brownlee [5] to maintain a table of measurement groups. Each group may have a list of flows belonging to it. Two new flow attributes are also implemented:

- GroupIdent, which is the ID of the measurement group the traffic flow belongs to;
- MeasurementData, which indicates that dedicated monitoring packets (belonging to a specific measurement group) are associated with a traffic flow. These packets cause the meter to create measurement data records for each traffic flow in the measurement group.

When flow data is read from the NeTraMet meter, it includes all the measurement data records produced for a flow since the previous meter reading. This system is simple to understand and use, and it allows one to collect measurement data for any required flow. For example, using a single ruleset, one could measure the behaviour of flows to several different destinations (IP addresses), carrying different protocols (TCP, UDP), and different application traffic (port numbers). The measurement modifications to NeTraMet are included in the NeTraMet distribution, from version 4.5b8 and later on. In our implementation and tests version 5.1b11 was used, in which round-trip time as well as one-way delays can be measured.

5. System Solutions for End-to-End Monitoring

The basic idea is to use a general-purpose traffic meter (not limited to specific protocols or traffic) that is triggered by the signalling information in the SIP messages that precede the established call. Two different solutions are presented below. In the first case a Perl program at the SIP server side manages the monitoring of traffic between the SIP user agents. In the second case the monitoring activities are managed by the clients themselves without participation of the SIP server side.

5.1. A Server-Based Solution

In this case the monitoring activities are handled by a manager program written in Perl that resides on the same machine as the SIP proxy server. The system consists of the following components and functions.

- Capture and analysis of the SIP messages to and from the SIP proxy server on UDP port 5060.
- A srl file (simple ruleset language) is created and compiled to a rules file when a call is being set up. The information needed is the callers IP address and port number for voice and data transport plus the port number used by the monitoring packet generator.
- NeMaC uses the configured rules file to control the meters at the SIP user agents and collect the measurement data.
- Monitoring packets are generated from the calling party (A) with the desired packet frequency and pattern. When a monitoring packet arrives at the called party (B) it triggers a monitoring packet to be sent in the opposite direction from B to A (Figure 4).
- The meters at both SIP user agents count packets and bytes in the data flows between the end users. The cumulative values of these counters are stored when a monitoring packet is detected along with a timestamp plus an ID of the monitoring packet.
- When the manager detects a BYE message the monitoring of the session and the monitoring packet generator are terminated. The proxy server has to be configured to "record route"; otherwise the BYE messages are sent solely between the user clients.

Monitoring data gathered by the meters are collected via SNMP by NeMaC and saved as flow files at the proxy server side. The measurement results are then computed in the following way.

5.2. Results

Packet losses can be computed as the difference between the number of packet and bytes sent at meter A and the number of packets and bytes received at meter B and vice versa. A monitoring packet sent from user agent A gets timestamp T_1 when it leaves meter A, and timestamp T_2 when it arrives at user agent B (Figure 4). A monitoring packet sent from user agent B gets timestamp T_3 when it leaves meter B and timestamp T_4 when it arrives at user agent A. One-way delays from the calling party to the called party is T_2-T_1, and T_4-T_3 in the opposite direction. Accurate results require stable and synchronized computer clocks. The round-trip time is computed as T_4-T_1-(T_3-T_2), where T_3-T_2 is the extra delay at user agent B. The inter-arrival jitter can be computed as the variation of the inter-arrival times compared to the inter-transmission times. This is e.g. the case in the RTCP receiver reports based on RTP timestamps [1]. Let *T_diff_sent* be the time distance between when packet *n*+1 and packet *n* and were sent, and *T_diff_received* be the time distance between when packet *n*+1 and packet *n* were received. The inter-arrival jitter for the packets *n* and *n*+1 is then *T_diff_received* minus *T_diff_sent*. The advantage is that synchronized clocks are not required, which is the case if jitter is computed as variation in delays.

The cumulative counter values of bytes and packets and a timestamp are stored each time a monitoring packet is recognized by the meter. This makes it possible to compute throughput (sent and received) in bits per second and packet loss with a resolution that corresponds to the size of the monitoring block, i.e. the distance between consecutive monitoring packets. The loss process can be expressed in terms of the length of loss periods and loss-free periods, and throughput can be given per monitoring block [7].

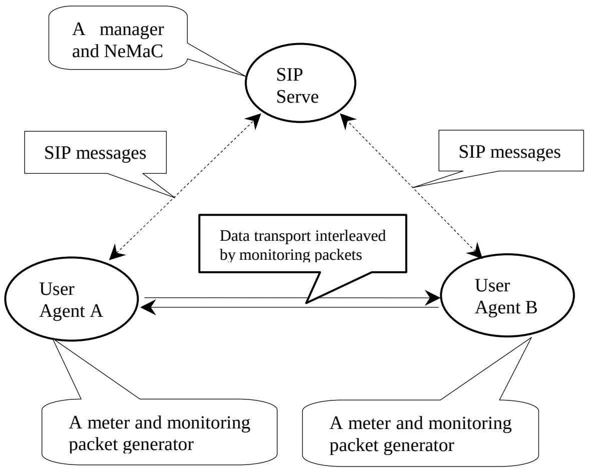
A manager and NeMaC
SIP Serve
SIP messages
SIP messages
Data transport interleaved by monitoring packets
User Agent A
User Agent B
A meter and monitoring packet generator
A meter and monitoring packet generator

Figure 3. A server-based solution for end-to-end monitoring of performance parameters in data traffic between the user agents.

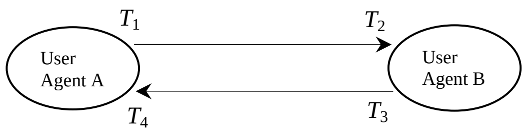

Figure 4. Timestamps for monitoring packets in both directions.

5.3. A Client-Based Solution

For a client-based solution (Figure 5) the monitoring activities are handled by a program written in Perl at the user agents. In principle, the only difference is that the manager function including NeMaC is moved from the SIP proxy server side to the clients. The measurement results are the same as presented in the previous case.

6.0. Measurement Results

The server-based solution has been tested with a user agent in Karlskrona in the southern part of Sweden and the other user agent and the proxy server placed at KTH in Stockholm. The calls between the user agents (Linphone [9] and Kphone [10] were used) are set up using the SER SIP server from iptel.org [11]. The VoIP traffic uses the UDP/RTP protocols.

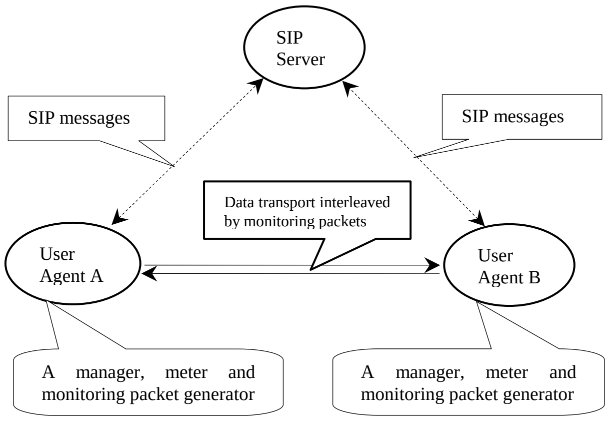
SIP Server
SIP messages
SIP messages
Data transport interleaved by monitoring packets
User Agent A
User Agent B
A manager, meter and monitoring packet generator
A manager, meter and monitoring packet generator

Figure 5. A client-based solution for end-to-end monitoring of performance parameters in data traffic between the user agents.

Figure 6 to Figure 12 illustrate possible results running this prototype implementation. Approximately one monitoring packet was sent per second and around 50 UDP/RTP packets per second. The monitoring packets have the same size (200 bytes) and the same transport protocol (UDP) as the user packets. The measurement periods lasted 5 minutes. The monitoring packets and timestamps follow the procedure outlined in Figure 4. The following statistics and graphs (produced in MATLAB) were obtained for delays, delay variations, inter-arrival jitter and throughput. The loss rate was very low and is therefore not shown in the graphs.

The inter-arrival jitter (defined in Section IV.B) for packets sent from Karlskrona to Stockholm is shown in the histogram in Figure 6, and in the other direction in Figure 7. The round-trip time distribution is depicted in Figure 8 with a mean value of 12.7 ms. The distribution of round-trip time variation computed according to the ITU-T type of definition [12] is shown in Figure 9, and according to an IPPM type of definition [13] in Figure 10. The ITU-T type of jitter has been interpreted as the packet delays minus the minimum packet delay (an approximation of the propagation time). The IPPM type of jitter has been interpreted as the difference between consecutive packet delays.

The time between the arrival of the monitoring packet in the receiving node and the departure time of the monitoring packet in the opposite direction, T_3-T_2, is subtracted when the round-trip time is computed. The histogram in Figure 11 shows this extra delay time. The throughput for incoming traffic (measured at party B) per monitoring block can be seen in Figure 12.

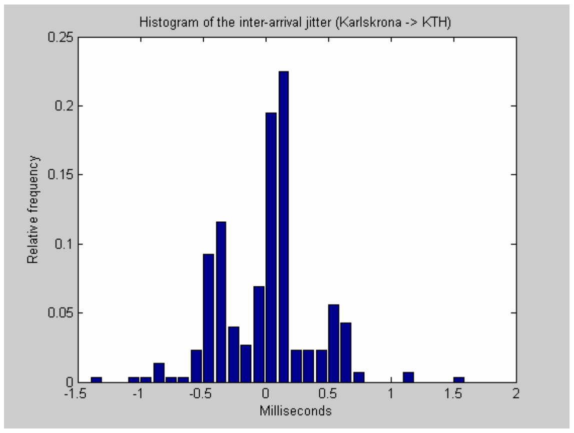

Figure 6. A histogram showing inter-arrival jitter (Karlskrona to Stockholm).

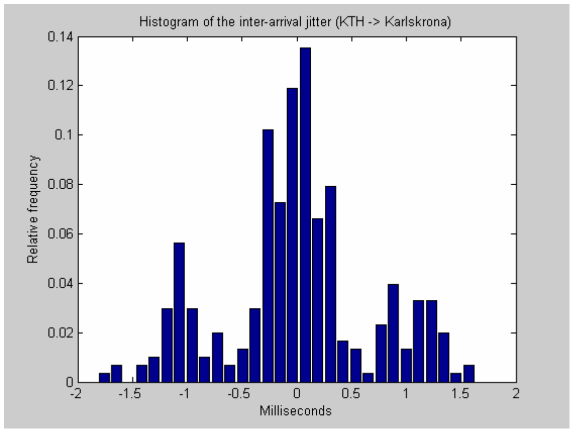

Figure 7. A histogram showing inter-arrival jitter (Stockholm to Karlskrona).

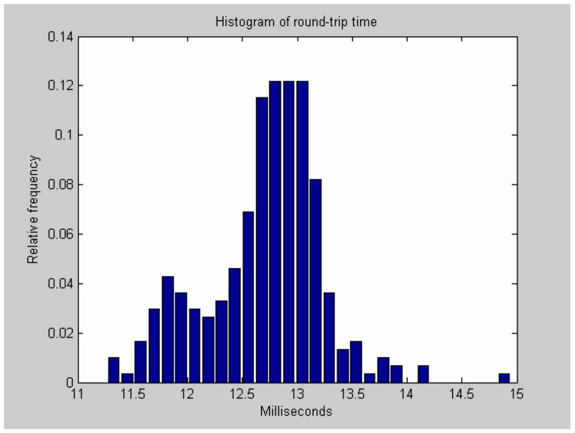

Figure 8. A histogram showing round-trip time distribution during the measurement period.

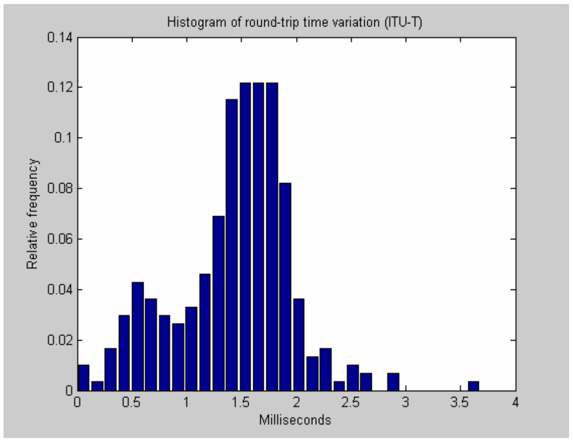

Figure 9. A histogram showing round-trip time variation during the measurement period with the ITU-T type of definition.

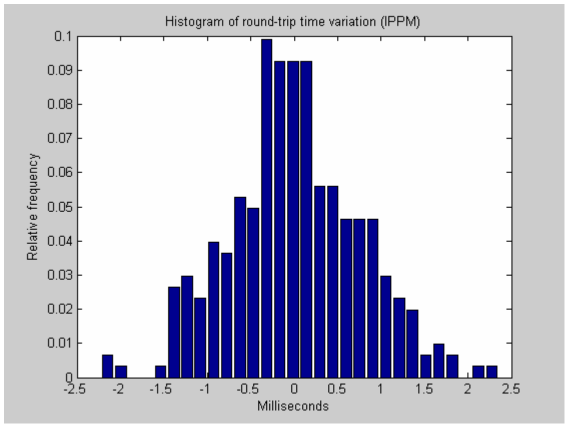

Figure 10. A histogram showing round-trip time variation during the measurement period with the IPPM type of definition.

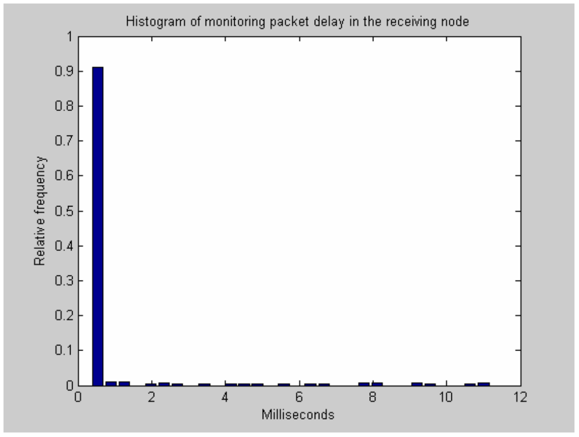

Figure 11. A histogram showing the monitoring packet delay time in the called party node before a monitoring packet is sent in the opposite direction to the calling party.

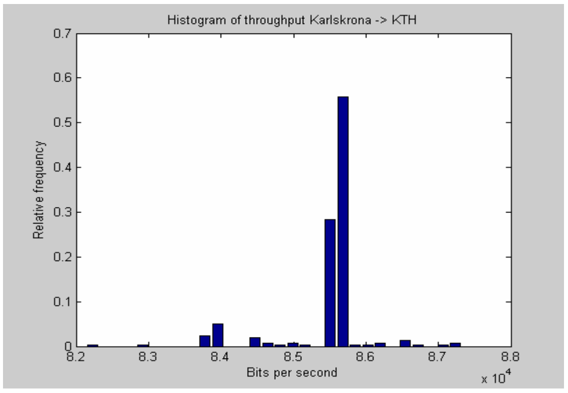

Figure 12. A histogram showing throughput per monitoring block from Karlskrona to Stockholm during the measurement period.

7. Discussion of Results

The server-based solution can be used by an operator in different ways. One alternative is to monitor randomly chosen sample sessions to get an overall picture of the call quality for all sessions handled by the specific proxy server. Other possibilities are to measure certain customers or samples of calls for every customer. The requirement for the server-based implementation is that the SIP proxy server is configured to "record route".

In end-to-end measurements between ordinary end users require stable and synchronized computer clocks. Our experience is that personal computers attached to NTP servers are inaccurate and unreliable. This is also shown in several studies, e.g. [14]. Our conclusion is therefore that in absence of high precision synchronized time stamping functions it is better to refrain from one-way delay estimates, and instead focus on measuring round-trip time and inter-arrival jitter between ordinary end users. One difficulty in measuring end-to end delays is how to distinguish the contribution to the delay budget from the different parts of the network and the client itself. From a network operator's point of view it seems likely that end-to-end delay measurements would benefit a lot from complementary edge-to-edge measurements within the network as well. This would facilitate to determine how different parts of an end-to-end connection contribute to the entire delay budget. An alternative approach is the protocol RTCP-XR that relies on the reports from the real-time control protocol, a companion protocol to RTP. However, our approach has been to apply a more generic measurement method that can be reused in other scenarios and is not dependent entirely on RTP. The round-trip time variation was presented in two different metrics, one inspired from ITU-T and the other from IPPM's definition. In the first case the constant part of the delays was eliminated in order to focus on the variable part. This is probably an advantage if you want to use these results to e.g. dimensioning jitter buffers.

The future work for this prototype implementation is to perform more extensive measurements and especially study packet loss. This method using monitoring packets and NeTraMet makes it possible to determine the loss distribution in time, such as the length of loss-free periods and loss periods.

8. Conclusions

We have presented an architecture and prototype implementation of end-to-end monitoring of performance parameters between end users in SIP-based communication. The approach is to combine signalling information and measurements of user data traffic. Two different prototype implementations have been presented. The SIP proxy server-based solution consists of a Perl manager program that detects when calls are being set up and activates the meter agents at both user agents. The actual measurements are performed by the NeTraMet meters and controlled by NeMaC. In the client-based solution the manager has been moved from the server side to the clients. The test measurements illustrate some of the results that can be obtained per session; packet loss, round-trip delays and their variation, inter-arrival jitter and throughput. The granularity of the metrics is determined by the size on the monitoring block, i.e. the distance between monitoring packets.

9. References

[1] [RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003.

[2] RTP Control Protocol Extended Reports (RTCP XR), RFC 3611, November 2003.

[3] SIP: Session Initiation Protocol, RFC 3261, June 2002.

[4] NeTraMet - a Network Traffic Flow Measurement Tool , http://www.caida.org/tools/measurement/netramet/

[5] NeTraMet beta versions: ftp://ftp.auckland.ac.nz/pub/iawg/NeTraMet/beta-versions/

[6] T. Lindh and N. Brownlee: "Integrating Active Methods and Flow Meters - an implementation using NeTraMet", Passive and Active Measurement workshop (PAM2003), San Diego, April 2003.

[7] T. Lindh, "Performance Monitoring in Communication Networks", Doctoral thesis, KTH, April 2004, http://media.lib.kth.se/dissengrefhit.asp?dissnr=3724

[8] NeTraMet & NeMaC Reference Manual v4.3, http://www2.auckland.ac.nz/net/Accounting/ntmref.pdf

[9] Linphone: http://www.linphone.org/?lang=us&rubrique=1

[10] Kphone: http://www.wirlab.net/kphone/

[11] SIP Express Router, http://www.iptel.org/ser/

[12] Network Performance Objectives for IP-Based Service, ITU-T Recommendation Y.1541, February 2002.

[13] IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), RFC 3393, November 2002.

[14] P. Arlos,"On the quality of computer network measurements", Doctoral thesis, Blekinge Institute of Technoloy, 2005.