AvBandTest – a Testing Tool for Implementations of Available Bandwidth Estimation Algorithms

Dmitry Kachan, Eduard Siemens
Anhalt University of Applied Sciences - Department of Electrical, Mechanical and Industrial Engineering
Bernburger Str. 57, 06366 Koethen, Germany
E-mail: d.kachan@emw.hs-anhalt.de, e.siemens@emw.hs-anhalt.de

***Abstract* — This work describes a test tool that allows to make performance tests of different end-to-end available bandwidth estimation algorithms along with their different implementations. The goal of such tests is to find the best-performing algorithm and its implementation and use it in congestion control mechanism for high-performance reliable transport protocols. The main idea of this paper is to describe the options which provide available bandwidth estimation mechanism for high-speed data transport protocols and to develop basic functionality of such test tool with which it will be possible to manage entities of test application on all involved testing hosts, aided by some middleware.**

Keywords: **High-speed transport; available bandwidth, congestion avoidance, testing of algorihtms.**

I. INTRODUCTION

A transport protocol is a complex system with various number of different logic parts – modules, which together perform transmission of data between involved peers. In [1] we have already shown that even modern commercial transport protocols reveal throughput performance being far from the optimum, and there is a large room for improvements. However, optimization of such a system is a rather complex task, and usually it is reduced to improvement of some particular modules. This paper addresses improvement of congestion avoidance algorithms (congestion control module) and performance of high-speed reliable transport protocols by means of finding a best-suitable available bandwidth estimation mechanism.

II. RELATED WORK

Phenomena like congestion in IP networks occur due to one of the fundamental principle of Internet – best-effort delivery. For reliable transport protocols the only chance to perform well is to use some mechanisms to control instant network utilization and properly react on congestions occurrence. Congestion control mechanisms used in contemporary transport protocols are mostly either window-based or rate-based. Window-based congestion control algorithm are well known and widely used in TCP [2] [3]. Rate-based congestion control has been widely used in ATM systems [4] [3]. However, in [5] Y. Gu et al. use rate based congestion control even in IP networks for a UDP-based transport over high-bandwidth and high-delay links. Their experiments show that the use of rate based congestion control for transport protocols is quite efficient. In [6] L. J. Latecki et al. use slightly modified SLoPS (Self-Loading Periodic Streams) algorithm [7] to develop congestion control for media applications based on estimation of the instantly available bandwidth estimation. A. K. Aggarwal et al. in [8] are using available bandwidth estimation to detect congestions in the data networks. This technique of available bandwidth estimation is also known from TCP [9].

There are many different approaches for measurement of available bandwidth. Most of them are based on PGM (Probe Gab Model) [10] [11] [12] or on PRM (Probe Rate Model) [7]. Within PGM methods, one peer sends a train of packet pairs to a corresponding peer, and based on the dispersion of pairs of packets, receiver peer can make an estimation of the available end-to-end bandwidth. The advantages of these methods are that they are quite fast and generate not much additional traffic in the network. However PGM methods are not able to provide adequate estimation of available bandwidth in presence of cross traffic in the multi-hop path. Considering that Internet has almost always a multi-hop configuration, there are some critical views on the accuracy of results of PGM models for available bandwidth estimation [13].

The methods based on PRM provide more adequate results of estimation. In [14], C. D. Guerrero et al. are comparing common solutions for available bandwidth estimation, and according to this research the minimal value of estimation error has been achieved by *pathload* [7] – a tool based on PRM (SLoPS algorithm). The disadvantages of this method are high estimation time due to multiple iterations of the algorithm, and high load of a link with estimation traffic.

The idea of this work is to use both PGM- and PRM-based algorithms in the ways, where they provide their best advantages: less estimation time or more accurate estimation result to make high speed data transmission more intelligent.

III. AVAILABLE BANDWIDTH ESTIMATION FOR HIGH SPEED TRANSPORT PROTOCOLS

There are two basic ways how to use available bandwidth estimation in transport protocols – initial estimation and estimation during the transmission. First one should be a very

fast method, which, probably, gives not very accurate, but at least approximated values of available bandwidth before the transmission starts. It is necessary to define the initial data rate at the very beginning. For this phase of transmission, an algorithm based on PGM could be used.

Estimation of available bandwidth during transmission is a method, which could take more time, however it will be expected to achieve more precise results. The result of this estimation will be used for soft reaction of transmission on the changes in a network, such as an appearance or disappearance of cross traffic in a path, e.g. by increasing or decreasing of sending rate in order to avoid congestion and for maximal link utilisation. For this kind of estimation PRM-based methods can be used. High speed transport protocols such as UDT [15] or RWTP [16] inject a time stamp into each data packet. It could eliminate the main disadvantage of PRM (high load of a link by probe traffic), since probing traffic could be carried piggy-back in data traffic of the transport protocol and the results could be transmitted by means of ACK messages. It means that no extra traffic will be generated. In that case SLoPS algorithm should be slightly modified to not make active measurements that include sending of probing traffic, but to make a passive, periodically analysing the time stamps in received packets.

To develop and test the modules for available bandwidth estimation it is possible either to implement the respective algorithms within an open source protocol, or to write a light weight application – vehicle, that gives a chance not only to implement algorithms, but also make performance tests to evaluate them. Implementation of such algorithms directly in the source code of a whole protocol stack could unintentionally break the protocol, or modification of one certain software module could negatively affect another module. Besides that testing of such, implemented within the protocol stack, algorithm is difficult because its behaviour would strongly depend also on implementation of the protocol.

IV. Structure and Implementation of AvBandTest

The implementation of the algorithms and accordingly the test tool has been implemented using C++ under Linux operation system. The reason for this it is that the source code of high speed protocols such as UDTv4 [15] or RBUDP [17] are implemented in C/C++. According to it, congestion control mechanism implemented in C++ can easily be linked to these protocol implementations in further tests.

In Figure 1 an interconnection model between two computing nodes is shown. For running tests, two types of traffic are used: first one is probing traffic – that are packets, generated by PGM and PRM algorithms to estimate

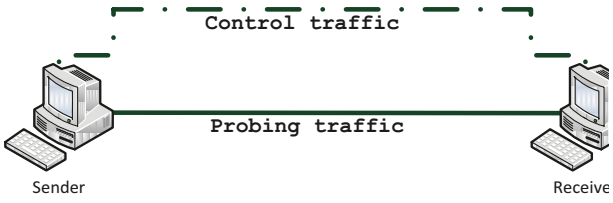

Figure 1. Interconnection scheme

bandwidth; the second one is a control traffic that contains messages which set parameters for available bandwidth estimation, start an estimation session and share result with their peer. Most of the control traffic messages will be used only during experiments to find the best algorithm implementation and the best estimation policy e.g. regarding the amount of probing traffic or the time of estimation, for using it in high speed transport protocols. After implementation of such a mechanism in a real congestion control mechanism, negotiation will be performed by means of service messages of the protocol. It is important that implementation of available bandwidth estimation mechanism will not use additional socket connections.

Since different kinds of impairments take place in telecommunication networks, some packets of both, control and probing traffic, could be received corrupted. The implementation of estimation mechanism should be able to handle corruption of probing traffic. In contrary, integrity of control traffic is out of scope of this test tool and according to that control interconnection will be implemented by means of an existing middleware. Such implementation of interconnection should be robust and allow focusing on the probing traffic only without paying much attention on generating of control information.

The structure of AvBandTest is presented in Figure 2.

A. Middleware

The Object Management Group (OMG) has defined a Common Object Request Broker Architecture (CORBA) [18] decades ago. Different vendors made different implementations of CORBA such as: MICO [19], OmniORB [20] etc. The main problem of CORBA is that the standard was not fully defined, and so different implementations of CORBA are mutually not fully compatible. It made the idea of a really interoperable middleware based on CORBA utopic. Furthermore, the development of many versions of CORBA is already discontinued, e.g. the recent version of MICO has been released September, 14th 2008. Moreover, with definition of a new C++ standard – C++11, applications, that worked under MICO are not compiling anymore. The better

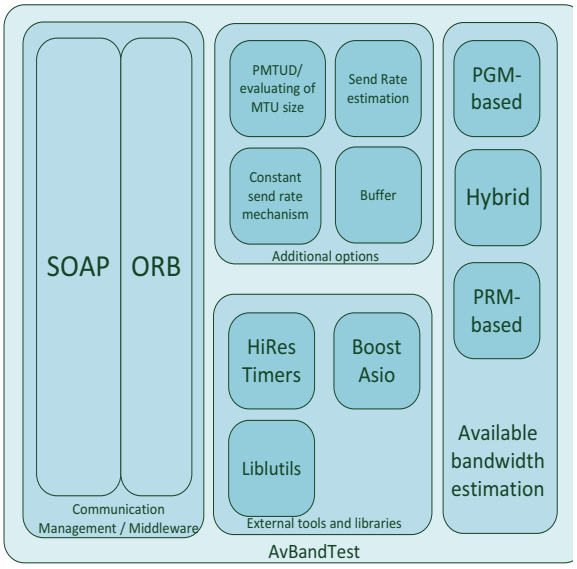

Figure 2. Structure view of AvBand

situation is with OmniORB – latest release was in July of 2011. In contrary to MICO CORBA it allows establishing of connections directly from application, OmniORB uses NameService for communication between hosts. It makes using of such approach not comfortable, because it is needed to starts NameService on each involved host. Moreover CORBA by itself is a sophisticated system with huge amount of its own abstractions, what makes development using this middleware relatively difficult task.

Another widely used protocol specification for remote object invocation is SOAP [21] (Simple Object Access Protocol) which provides a simple and robust technique for message negotiation and data structure exchange. Many different implementations of SOAP use various protocols e.g. HTTP and SMTP for message transmission [22]. There is an open source software development toolkit for C++ – gSOAP, which provides means to automatically generate XML and WSDL code from C++ data and vice versa [23].

On the one hand CORBA has a complete ORB architecture, but difficult enough implementation and not finally defined standard. On the other hand SOAP provides stable communication means, however it has no built-in ORB architecture. The problem of communication management for *AvBandTest* has been solved by development of a simple ORB architecture like one provided by SOAP.

B. External tools and libraries

In C++, all operations with sockets are used from the standard C library. A simple IP communication application, using native C operations, seems quite heavy and hard to read, especially due to the lack of strong type-safety. Moreover, the error handling is also inherited from C, what is pretty inconvenient for C++ programming. Object oriented approach in C++ for handling of IP communications is already implemented e.g. by *Asio* library that is included in a set of libraries called Boost. There is a big community that discusses, makes changes and tests the source code of the Boost library. Moreover, many of Boost libraries in the past have been assigned as C++ standard. The library *Boost.Asio* [24] provides mostly all what is needed to make a comfortable interconnection between sockets including convenient error handling mechanism.

The library *Liblutils* is written by E. Siemens in 2002 and is used within *LTest* [25]. It contains a lot of different function including C++ socket interconnection, but this part is here exchanged to *Boost.Asio* due to the wide supporters community of Boost. *AvBandTest* uses *Liblutils* now only due to a number of convenient strings manipulation functions in C++.

In such tasks as congestion control, time measurement, accurate time fetching and efficient time calculation becomes critical part. In [26] the authors share some novel ideas how to measure the time with the maximal possible resolution on common PC systems by performing of assembly code, wrapped around by C++ interfaces to get an access directly to timer hardware. All these ideas and a number of comfortable functions are implemented in the work of I. Fedotova in the library *HiResTimers* which is also described in [26]. This library is used by *AvBandTest* for time interval measurements.

C. Additional options

1) Buffer

Estimation of available bandwidth will be performed by the transmission of probe traffic on one side and reception of it at another side. For both, PGM and PRM, it is necessary to transmit time stamps in each packet of probe traffic. Beside the time stamp, each sample must carry a sequence number to pervert packets reordering in the network. Both, time stamp and sequence number allocate together not more than 12 bytes of memory: 4 bytes for nanoseconds, 4 bytes for seconds of Unix epoch and 4 bytes for the sequence number. In [11] is shown that the best size for sample packets for PGM should be "not very small and not very large" - between 600 bytes and 1 500 bytes. However it is important to note, that this research has been done with a presumption, that maximal transfer unit (MTU) in internet is 1 500 Bytes. In high speed networks, to achieve maximum capacity, sometimes extended MTU size (Jumbo frames till 9 000 bytes) is used. So it makes sense to check whether these packet sizes are also optimal for extended MTU sizes in the path, or the size of probing packets should be also extended. Nevertheless, there are only 12 bytes of useful data in each sample and to satisfy size conditions of probe packets, the rest space will be allocated with dummy data, filled up randomly.

In a transport protocol the very important part of implementation is the data buffer at the sender and receiver side. The rate of data reception and speed of access to data strongly depends on the buffer implementation. For test purposes *AvBandTool* should also have buffer mechanism that does not slow down the data transmission performance. The buffer has been designed as one separate class, "*ReceivedData*" that contains a vector with the measurement data from received packets. There is no necessity to store whole samples because most of data in the packet do not carry measurement information. For fast storing measured data at the receiver, receiver side will be notified by the sender via the control channel about the amount of expected packets to allocate memory for all expected packets before the reception of IP packets starts. Such operation will be repeated for each measurement's iteration and after calculating of result the buffer will be released. Further improvement of the receiver's buffer could be done by adapting measurements to periodical on-the-fly available bandwidth estimation. This measurement will be continuous with some intermediate results and it will be hard to release the buffer after some certain time period in that case. This problem could be solved by implementation of "*Ring buffer*", which idea is described by E. Siemens et al. in [27].

Beside this vector, the class contains a number of functions to access to particular data components of a sample such as sequence number or time at which the sample has been sent or received. The class has interfaces to calculate and to return such statistical parameters as mean of inter-packet time at the reception or inter-packet time at transmission.

2) Constant send rate mechanism.

Such protocols as RBUDP have an option to send data with some certain data rate that should be chosen once before data transmission starts. However it would be more efficiently to use a mechanism that analyses the actual situation in the

network and notifies the transport engine about changes of available bandwidth. In that case transport will use available resources by maximum.

In *AvBandTest* a rate control mechanism is introduced to emulate data transmission on the data rates that are adopted according the available bandwidth during the data transmission. The main hitch of rate control on the data rates close to 10 Gbit/s is the accurate time measurement. The simple example of transmission of 1 500 bytes packet through 10 Gbit/s network can show it. The mean inter-packet time at the sending side in this case is:

$$t[s] = \frac{S[Byte]}{R\left[\frac{Byte}{s}\right]} = \frac{1\,500}{1\,250 \cdot 10^6} = 1{,}2 \cdot 10^{-6} s, \quad (1)$$

where t – is mean inter-packet time, S – IP packet size and R – used data rate.

Packets smaller than 1 500 has even less inter-packet time, so the measurement tool should be able to measure such short time intervals accurately. In [28] authors show that usual timers in Linux are not stable in choosing of timer source. Furthermore, time request's cost is significantly lower if to request timer system directly, instead of using standard Linux system call "*clock_gettime*". Time measurement by means of direct requests to timers are implemented in [26], where is shown that cost of time requests using this library for different machines were about 1 µs in the worst case and the values of tens of nanoseconds in regular cases. The accuracy of time measurement will always depend on type and hardware realisation of timer. However within *HiResTimer* library it is possible to achieve the most accurate time measurements.

V. First results

Using Apposite 10G in the 10G-lab of Anhalt University of Applied Sciences (Koethen, Germany), it is possible to emulate IP connections with different bottlenecks on the both sides: sender and receiver. In this way, the accuracy of bandwidth estimation of a prototype version of AvBandTest tool can be tested. The simplified diagram of the testbed is presented in Figure 3. Within presented setup the following bottlenecks were emulated for both sides: sender and receiver: 10 Mbps, 100 Mbps, 1 000 Mbps, 10 000 Mbps. At the time of writing of this paper, the only easiest PGM-based algorithm has been implemented. It works as follow: sender sends to the receiver a predefined amount of packets, which can be defined at the time of application start. These packets have been sent back-to-back – with minimal possible delay between them. Each packet contains: sequence number of a packet, time stamp, as described in section IV. Sender evaluates sending data rate, while it sends data, as sum of bytes of first thousand packets, or even less, if the whole transmission contains less than thousand packets, and divides it by time, that sender spent on sending of this amount of data. After transmission the

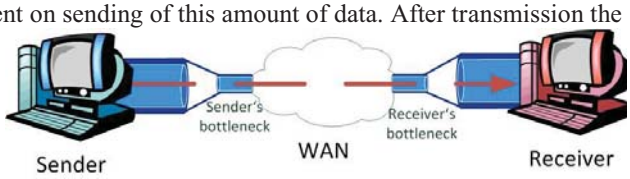

Figure 3 Simplified chart of testbed topology

receiver calculates the differences between timestamps that were included in each two consecutive data packets and calculates the mean value of these differences. For the same pairs of packets receiver calculates the mean value of time differences between the moments of receiving of each packet. The available bandwidth is evaluated as shown in (2)

$$AvBand[bit/s] = \frac{R_s[bit/s]}{\frac{1}{n}\sum_{i=1}^{n}\frac{\Delta t_i^R[s]}{\Delta t_i[s]}}, \quad (2)$$

where $AvBand$ is available bandwidth; R_s is sending rate; n – is a whole number of received pairs; Δt_i^R – is an inter-packet time at reception for *i* pair of packets; Δt_i – is an inter-packet time at transmission for *i* pair of packets.

Practically this method means that available bandwidth is back-proportional to a relation of sending inter-packet time interval and reception inter-packet time interval.

The plot in Figure 4 shows available bandwidth estimation error for each emulated bottleneck. The error here is a difference between emulated bottleneck and result of estimation in percent from bottleneck. Figure 5 shows the results of evaluation on a link with 50 ms of RTT, uniformly distributed in the forward and backward directions, and 0.5% of packet losses in the network in between the sender and receiver. "s" and "r" on the plots correspond to the positioning of the bottleneck on sender side and on the receiver side accordingly. In the case of 10 000 Mbps there are no bottlenecks on the sender and receiver sides. The evaluation has been performed 5 times for each combination of

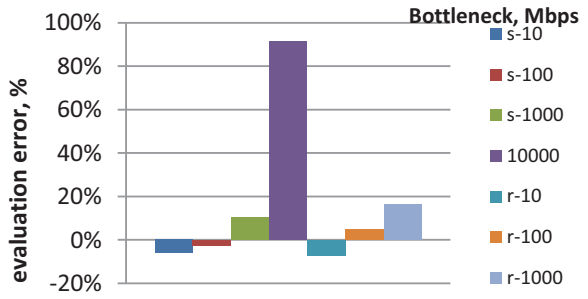

Figure 4. Error of available bandwidth evaluation in the network without impairments

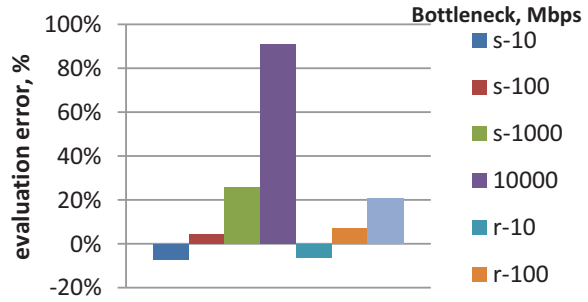

Figure 5. Error of available bandwidth evaluation in presence of 50 ms. of RTT and 0,5% of packet loss in the network

bottlenecks and the average values of mistakes are shown. It is worth noting that within bottleneck on sender side the result of evaluation is better in the network without impairments; however, in presence of them this behavior is not saved. We assume that for initial available bandwidth measurement the error of about 15% is acceptable. The obtained results showed that algorithm needs improvement. The current prototype implementation of AvBandTest is not able to measure available bandwidth up to 10 000 Mbps: as the plots show, in both cases evaluation of 10 000 was completely wrong. Improvement of this point is a significant task for further work.

VI. Conclusion

This work describes problems of available bandwidth estimation tests, which are used for development of fast data transport protocols. There are two significant parameters which can be tested with such techniques: estimation of the very beginning sending rate and estimation of the target data rate of rate based congestion control. Discussion about basic functionality and brief overview of components are presented in this work. A prototype of the tool that allows evaluation of different approaches for available bandwidth measurements has been implemented and tested as a result of this work.

VII. References

[1] D. Kachan, E. Siemens, V. Shuvalov. Comparison of contemporary solutions for high speed data transport on WAN connections. Accepted to: ICNS, Lisbon, Portugal. 2013.

[2] M. Allman, V. Paxson, W. Stevens. M. Allman, V. Paxson, and W. Stevens. IETF RFC 2581. 1999.

[3] A. S. Tanenbaum. Computer Networks. Third edition. New Jersey : Prenice Hall PRT, 1996. p. 813. ISBN 0-13-349945-6.

[4] H. Ohsaki et al. Rate-based congestion control for ATM networks. ACM SIGCOMM Computer Communication Review. 1995, Vol. 25, pp. 60-72.

[5] Y. Gu, X. Hong, M. Muzzucco, R. Grossman. Rate Based Congestion Control over High Bandwidth/Delay Links. IEEE/ACM Transaction on Networking. 2003, Vol. 11.

[6] L. J. Latecki, T Jin, J. Mulik. A Two-stream Approach for Adaptive Rate Control in Multimedia Applications. IEEE Int. Conf. on Multimedia & Expo. 2004.

[7] M. Jain, C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Transactions on Networking. 2003, Vol. 11 (4), pp. 537-549.

[8] A. K. Aggarwal, A. N. Bharadwaj, R.D. Kent. Active congestion control using available bandwidth-based congestion detection. Proc. ICAI'05/MCBC'05/AMTA'05/MCBE'05. 2005, pp. 390-397.

[9] R. Wang et al. Efficiency/Friendliness Tradeoffs in TCP Westwood. Proc. Computers and Communications, ISCC. 2002, pp. 304-311.

[10] C. Dovrolis, P. Ramanathan, D. Moor. What do packet dispersion techniques measure? proceedings of INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. 2001, Vol. 2, pp. 905-914.

[11] M. Jain C. Dovrolis. Packet-Dispersion Techniques and a Capacity-Estimation Methodology. IEEE/ACM Transaction on Networking. 2004, Vol. 12, pp. 963-977.

[12] V. J. Ribeiro et al. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. Passive and Active Measurement Workshop. 2003.

[13] L. Lao, C. Dovrolis, M. Y. Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. ACM SIGCOMM Computer Communication Review. 2006, Vol. 36 issue5.

[14] C. D. Guerrero, M. A. Labrador. On the applicability of available bandwidth estimation techniques and tools. Computer Communications. Vol. 33 Issue 1, 2010, pp. 11-22.

[15] Y. Gu, R. L. Grossman. UDT: UDP-based Data Transfer for High-Speed Wide Area Networks. Computer Networks (Elsevier). May 2007.

[16] E. Siemens, R. Einhorn, A. Aust. Multi-Gigabit Challenges: Similarities between Scientific Environments and Media Production. ACIT - Information and Communication Technology : s.n., 2010.

[17] E. He, J. Leigh, O. Yu, T. DeFanti. Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. In Proceeding of 5th Int. Conf. on Cluster Computing. 2002.

[18] S. Vinoski. CORBA: integrating diverse applications within distributed heterogeneous environments. Communications Magazine, IEEE. 1997, Vol. 35, pp. 46-55.

[19] A. Puder, K. Römer. MICO: An Open Source CORBA Implementation. - 3rd ed. s.l. : Morgan Kaufmann Publishers, 2000.

[20] S.-I. Lo, S. Pope. The Implementation of a High Performance ORB over Multiple Network Transports. In Middleware 98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing. 1998.

[21] M. Gudgin et al. W3C Recomendations. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). [Online] 04 27, 2007. [Cited: 01 2013, 14.] http://www.w3.org/TR/soap12-part1/.

[22] F. Curbera. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI . Internet Computing, IEEE. 2002, Vols. 6 , Issue: 2 , pp. 86-93.

[23] R. A. van Engelen. gSOAP Toolkit. The gSOAP Toolkit for SOAP Web Services and XML-Based Applications . [Online] [Cited: 01 15, 2013.] http://www.cs.fsu.edu/~engelen/soap.html.

[24] C. Kohlhoff . Boost C++ Libraries. Boost.Asio. [Online] [Cited: 01 15, 2013.] http://www.boost.org/doc/libs/1_52_0/doc/html/boost_asio.html.

[25] E. Siemens, S. Piger, C. Grimm, M. Fromme. LTest – A Tool for Distributed Network Performance Measurement. Proc. Consumer Communications and Networking Conference, 2004. First IEEE. 2004, pp. 239-244.

[26] I. Fedotova, E. Siemens, H. Hu. A High-precision Time Handling Library for Tracking Internet Packet Dynamics. Accepted to: ICNS, Lisbon, Portugal. 2013.

[27] E. Siemens, Xiaopeng Qiu. Datentransport in multimedialen Systemen: Effiziente Pufferspeicher für schnellen Datentransport. 2009. 978-3836479837.

[28] D. Kachan, E. Siemens, H. Hu. Tools for the high-accuracy time measurement in computer systems. T-Comm – Telecommunication and Transport. 2012, 8, pp 23-27.