# NetPerfMeter – A Versatile Tool for Multi-Protocol Network Performance Evaluations

Thomas Dreibholz, Hakim Adhari, Martin Becke, Erwin P. Rathgeb
University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstraße 29, 45326 Essen, Germany
{dreibh, hakim.adhari, martin.becke, rathgeb}@iem.uni-due.de

***Abstract*—Transport Layer protocols supporting multipath transfer, i.e. to simultaneously utilise multiple Network Layer paths, are actively discussed in the IETF – particularly in the context of Multi-Path TCP (MPTCP) and Concurrent Multipath Transfer for SCTP (CMT-SCTP). Congestion control for such protocols is an important research topic.**

**In this code contribution paper, we introduce our application model NetPerfMeter. NetPerfMeter has been developed for performance evaluations of different transport protocols, like for the Linux/FreeBSD performance metering application NetPerfMeter.[1][2]**

**Keywords: Application Model, NetPerfMeter, Performance Evaluation, Multipath Transfer**

## I. Introduction

Transport Layer protocols – like UDP [1], TCP [2] and SCTP [3]–[5] – are a highly crucial part of the protocol stack for all Internet applications. In order to achieve a good application performance, it is therefore useful to optimise and tune their performance, e.g. by using CMT-SCTP [6]–[11] or MPTCP [12]. Clearly, simulations – particularly based on OMNeT++ and the INET Framework – are very useful for this purpose.

The current INET Framework already contains application models for TCP (*TCPBasicClientApp*, *TCPSinkApp*, ...), UDP (*UDPBasicApp*, ...) and SCTP (*SCTPClient*, *SCTPServer*, ...). However, each of these application models uses its own parameter sets and has its own behaviour, making a performance comparison between different Transport Protocols (e.g. TCP vs. SCTP) difficult. In order to overcome this challenge, the NetPerfMeter application model has been developed. It provides an application model which is independent of the underlying Transport Layer protocol. This model, which is intended as a code contribution, will be shortly presented in this paper.

## II. The NetPerfMeter Module

### A. The Module

The NetPerfMeter application model has been realised as a simple module named NetPerfMeter, which has to be connected to the UDP, TCP and SCTP modules. In order to simplify the usage, it has been integrated into the *StandardHost* module as depicted in Figure 1. *StandardHost* is the basis for nearly all IP-based models, i.e. routers as well as endpoints. The integration of NetPerfMeter has been realised as array "netPerfMeter", i.e. there may be a variable number of NetPerfMeter instances.

[1]Parts of this work have been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG).

[2]The authors would like to thank Irene Rüngeler, Michael Tüxen and Feng Wang for their friendly support.

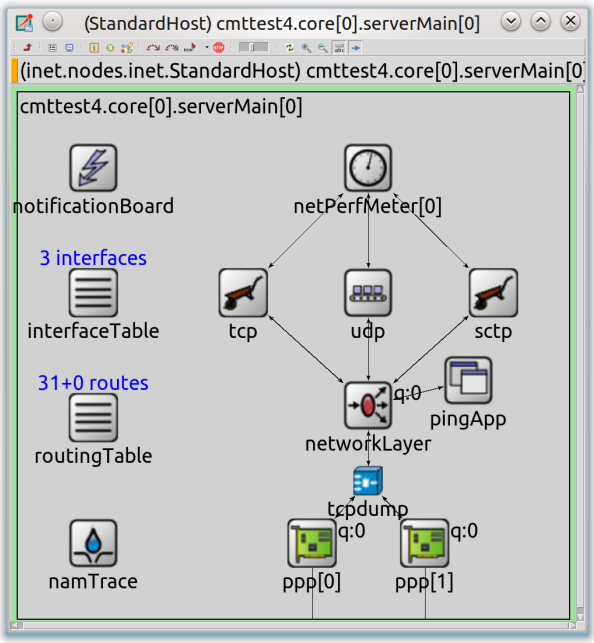


Figure 1. The Extended StandardHost Module

### B. Features and Parameters

Similar to the real Linux/FreeBSD-based performance metering program NetPerfMeter [13], the application module provides the unidirectional and bidirectional transfer of saturated and non-saturated flows as well as statistics recording. Its parameters are listed in Table I.

For a performance test, a connection is established between two NetPerfMeter instances. This connection establishment process, by using the protocol specified by the parameter protocol (i.e. "SCTP", "TCP" or "UDP"), is triggered by the instance in the so-called *Active Mode* (i.e. the client side). The mode of an instance is configured by using the activeMode parameter; a setting of "true" turns the instance into active mode. Address and port of the remote instance in the so-called *Passive Mode* (i.e. the server side; configured by setting activeMode to "false") are provided by the parameters remoteAddress and remotePort. The local address and port of an instance – in both modes – may be set by the parameters localAddress and localPort. Since SCTP supports multi-homing, the local address may actually be a list of addresses, or – in the usual case – just left empty. In this case, *all* available Network Layer addresses are used. The primary path may be set by the primaryPath parameter. Also, for SCTP, the number of outbound streams (parameter: outboundStreams) and the maximum number of

| Parameter | Functionality | Default Setting |
|---|---|---|
| activeMode | Active Mode (true) or Passive Mode (false) | true |
| protocol | Transport Protocol (“SCTP”, “TCP” or “UDP”) | “TCP” |
| localAddress | Local IP Address (“” for “*any*” Address) | “” |
| localPort | Local Port | 9000 |
| remoteAddress | Remote IP Address (Active Mode only) | “” |
| remotePort | Remote Port (Active Mode only) | 9000 |
| primaryPath | Primary Path (SCTP only) | “” |
| outboundStreams | Number of Outbound Streams (SCTP only) | 1 |
| maxInboundStreams | Maximum Number of Inbound Streams (SCTP only) | 16 |
| connectTime | Absolute Time of Connection Establishment | 0s |
| startTime | Relative Time of Payload Data Transfer Start | 1s |
| resetTime | Relative Time of Measurement Start | 5s |
| stopTime | Relative Time of Measurement Stop | 30s |
| frameRate | Frame Rate (0Hz = saturated sender) | 10Hz |
| frameSize | Frame Size (0B = flow turned off) | 1452B |
| frameRateString | Outgoing Frame Rate per Stream, separated by “;” | “” |
| frameSizeString | Outgoing Frame Size per Stream, separated by “;” | “” |
| maxMsgSize | Maximum Message Size (SCTP and UDP only) | 1452B |
| queueSize | Queue Size (SCTP and TCP only) | 1000B |
| unordered | Fraction of Unordered Messages (SCTP only) | 0.0 |
| unreliable | Fraction of Unreliable Messages (SCTP only) | 0.0 |

Table I
THE PARAMETERS OF THE NETPERFMETER MODULE

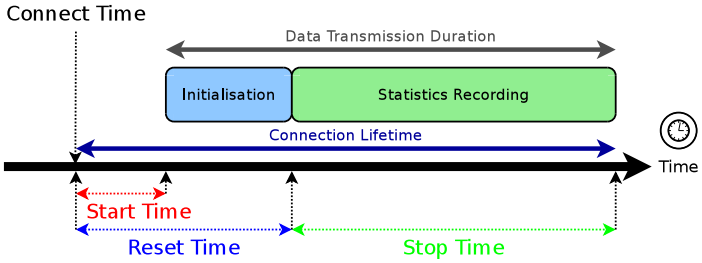


Figure 2. The NETPERFMETER Timing Configuration

inbound streams (parameter: maxInboundStreams) can be configured. These settings are used during the SCTP association setup to negotiate [3] the number of streams in each transfer direction.

Figure 2 provides an illustration of the NETPERFMETER connection timing configuration. Connection setup is started at the given *Connect Time* (parameter: connectTime). After a given *Start Time* (parameter: startTime; relative to the begin of the connection establishment), the transfer of payload data starts. At that time, the connection must have been established; otherwise, the simulation stops with an appropriate error message. Usually, the beginning of a communication leads to some kind of irregular initialisation behaviour. For example, the congestion window may have to grow using slow start, etc.. In order to avoid these effects distorting the results, the *Reset Time* (parameter: resetTime; relative to the connect time) defines the length of a settling time span. After that time, all previously generated statistics are reset. The actual duration of the statistics recording phase is given by the *Stop Time* (parameter: stopTime; relative to the time of the statistics reset). At the end of this phase, the data transfer is stopped, scalar statistics are written and the connection is finally shut down.

Outgoing payload data is transmitted as frames in a given interval with a given size. Frames are split up into datagrams, with a maximum size given by maxMsgSize. The respective parameters to configure frame rate and frame size are frameRate and frameSize. The setting of frameSize=0 B turns the flow off; frameRate=0 Hz configures a saturated sender. A saturated sender tries to send as much data as possible. The message queue is therefore filled with up to queueSize messages. Since UDP has no flow control, a saturated sender is only possible for SCTP and TCP. For using SCTP, if there are multiple outbound streams, the frame rate and frame size of each stream may be configured separately by providing them as colon-separated strings by the parameters frameRateString and frameSizeString, respectively. The application of strings in this case is necessary, since OMNET++ does not support parameter arrays. Furthermore, it is possible to send a given fraction of the datagrams with unordered delivery (parameter: unordered). or using partially reliable transfer (parameter: unreliable). For each message, its kind of delivery (i.e. ordered or unordered) and reliability (reliable or partially reliable) is selected randomly, with a uniform distribution, according to the configured fractions (from 0.0 – i.e. 0% – to 1.0 – i.e. 100%).

## III. THE CODE CONTRIBUTION

The NETPERFMETER source package can be downloaded from the project website [14]. It is also provided as a patch against the current GITHUB INET FRAMEWORK tree. The package consists of:

- the NETPERFMETER application module source in `src/applications/netperfmeter/`,
- a TCP module improvement to indicate available send buffer space and a bugfix for the REDQueue module,
- the modified StandardHost module and
- an example in `examples/netperfmeter/`. The included `README` file provides some further details.

Some more details on the implementation can be found in [15].

## IV. CONCLUSIONS

In this code contribution paper, we have introduced the NetPerfMeter application model for performance evaluations with different underlying Transport Layer protocols. Its application purpose is the performance comparison of different Transport Layer protocols, particularly with respect to multipath transport [16]. Some important research results based on this model can be found e.g. in [7], [9], [17]–[19].

## REFERENCES


[1] J. B. Postel, "User Datagram Protocol," IETF, Standards Track RFC 768, Aug. 1980, ISSN 2070-1721.

[2] ——, "Transmission Control Protocol," IETF, Standards Track RFC 793, Sept. 1981, ISSN 2070-1721.

[3] R. R. Stewart, "Stream Control Transmission Protocol," IETF, Standards Track RFC 4960, Sept. 2007, ISSN 2070-1721.

[4] A. Jungmaier, "Das Transportprotokoll SCTP," Ph.D. dissertation, Universität Duisburg-Essen, Institut für Experimentelle Mathematik, Aug. 2005.

[5] T. Dreibholz, M. Becke, H. Adhari, E. P. Rathgeb, I. Rüngeler, R. Seggelmann, and M. Tüxen, "Improvements to the SCTP Environment in the INET Framework," University of Duisburg-Essen, Institute for Experimental Mathematics," OMNeT++ Code Contribution, Feb. 2012.

[6] M. Becke, T. Dreibholz, J. R. Iyengar, P. Natarajan, and M. Tüxen, "Load Sharing for the Stream Control Transmission Protocol (SCTP)," IETF, Network Working Group, Internet Draft Version 03, Jan. 2012, draft-tuexen-tsvwg-sctp-multipath-03.txt, work in progress.

[7] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "On the Use of Concurrent Multipath Transfer over Asymmetric Paths," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida/U.S.A., Dec. 2010, ISBN 978-1-4244-5637-6.

[8] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "Evaluation of Concurrent Multipath Transfer over Dissimilar Paths," in *Proceedings of the 1st International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Singapore, Mar. 2011, pp. 708–714, ISBN 978-0-7695-4338-3.

[9] T. Dreibholz, H. Adhari, M. Becke, and E. P. Rathgeb, "Simulation and Experimental Evaluation of Multipath Congestion Control Strategies," in *Proceedings of the 2nd International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Fukuoka/Japan, Mar. 2012.

[10] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. Rathgeb, "Implementation and Evaluation of Concurrent Multipath Transfer for SCTP in the INET Framework," in *Proceedings of the 3rd ACM/ICST International Workshop on OMNeT++*, Torremolinos, Málaga/Spain, Mar. 2010, ISBN 978-963-9799-87-5.

[11] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006, ISSN 1063-6692.

[12] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. R. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF, Informational RFC 6182, Mar. 2011, ISSN 2070-1721.

[13] T. Dreibholz, M. Becke, H. Adhari, and E. P. Rathgeb, "Evaluation of A New Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain," in *Proceedings of the 19th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Hvar/Croatia, Sept. 2011, ISBN 978-953-290-027-9.

[14] T. Dreibholz, "Thomas Dreibholz's SCTP Page," 2012. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/sctp/

[15] ——, "Evaluation and Optimisation of Multi-Path Transport using the Stream Control Transmission Protocol," Habilitation Treatise, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, 2012.

[16] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb, "A Future Internet Architecture supporting Multipath Communication Networks," in *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Maui, Hawaii/U.S.A., Apr. 2012, ISBN 978-1-4673-0269-2.

[17] ——, "On the Fairness of Transport Protocols in a Multi-Path Environment," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Ottawa/Canada, June 2012.

[18] T. Dreibholz, M. Becke, H. Adhari, and E. P. Rathgeb, "On the Impact of Congestion Control for Concurrent Multipath Transfer on the Transport Layer," in *Proceedings of the 11th IEEE International Conference on Telecommunications (ConTEL)*, Graz/Austria, June 2011, pp. 397–404, ISBN 978-953-184-152-8.

[19] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. Rathgeb, "Applying TCP-Friendly Congestion Control to Concurrent Multipath Transfer," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth/Australia, Apr. 2010, pp. 312–319, ISSN 1550-445X.