# A Configurable FPGA-Based Traffic Generator for High-Performance Tests of Packet Processing Systems

Andreas Tockhorn, Peter Danielis, Dirk Timmermann
*University of Rostock*
*Institute of Applied Microelectronics and Computer Engineering*
*18051 Rostock, Germany*
*Tel./Fax: +49 (381) 498-7269 / -1187251*
*Email: {andreas.tockhorn;peter.danielis;dirk.timmermann}@uni-rostock.de*

***Abstract*—For the evaluation of high-speed packet processing systems, high performance traffic generators are needed. They have to be configurable to produce various traffic patterns and achieve preferably full Gigabit link utilization. The evaluation of packet processing systems has become a critical task as current open-source—especially software—tools have a lack of throughput and suitable hardware generators are very expensive. Thus, an FPGA-based testing framework containing a traffic generator and a performance monitor, each of them based on an FPGA, was developed. In order to evaluate different packet processing systems, this framework maintains a high level of configurability. Summarized, this paper presents an FPGA-based traffic generator, dedicated to high throughput testing of packet processing systems.**

***Keywords*-Traffic Generator, Performance Measurement, Network Testing**

## I. Introduction

Today, an increasing number of users subscribe for an Internet connection. Due to their increasing bandwidth demands, Packet Processing Systems (PPS), which execute tasks like packet classification, manipulation, and forwarding have to process packets at very high rates. To guarantee their correct functionality, they have to be tested under worst-case conditions i.e., with maximum traffic load. Consequently, traffic generators are needed for generating traffic to fully utilize links in order to evaluate a PPS's performance under these conditions. At the same time, traffic generators have to to be configurable to generate various patterns of traffic occurring in practice. Software-based traffic generators provide high configurability but have limited performance. Traffic generators implemented in hardware e.g., on a Field Programmable Gate Array (FPGA) are able to generate traffic at very high rates. However, they often lack a high degree of configurability as providing high configurability in hardware results in enourmous hardware costs or are too expensive for the proposed use.

Thus, this work has been inspired by the need of an affordable and configurable traffic generator offering full Gigabit Ethernet link utilization. The developed framework consists of an FPGA-based traffic generator, an FPGA-based traffic monitor to measure Gigabit link utilization, and a software tool to configure the traffic generator. An unique selling proposition of the proposed architecture is its integration in the hardware design process of PPS by reusing the therefore developed testbenches.

The traffic generator's performance is compared to that of the open-source software based traffic generator packETH [1]. PackETH provides convenient handling to carry out extensive tests and, to the best of our knowledge, its performance has not been evaluated before. We exemplarily derive the necessary level of configurability from tests to evaluate the performance of a recently published PPS called IPclip (IP Calling Line Identification Presentation) [2].

Briefly summarized, the main contributions of this paper are the following:

- Investigations on requirements for a configurable traffic generator are carried out.
- A combined hardware-software framework consisting of a configuration tool, traffic generator, and traffic monitor is proposed.
- The traffic generator's performance results for the IPclip use case are presented. They are compared to the performance results achieved for the traffic generator packETH.
- Integration of the framework into the hardware design process of PPS is illustrated.

The remainder of this paper is organized as follows: Section II contains an overview of related work. Section III investigates requirements for a configurable traffic generator. Section IV introduces a combined hardware-software prototype for the developed traffic generator. Section V presents performance results for both the proposed packet generator and packETH. The paper concludes in Section VI.

## II. Related Work

There is a wide variety of tools to generate traffic—both commercial and open-source solutions.

One example for a commercial software solution is a highly configurable traffic generator offered by ZTI for Windows XP and Vista [3]. It achieves a link utilization of up to 97.4 % for 1 Gbit/s Ethernet links. However, for

 ISBN: 978-1-61208-125-0

testing PPS as in the authors' targeted use case, ZTI's traffic generator is both too complex, does not offer full Gigabit Ethernet link utilization, and is not complimentary.

Most of available open-source software traffic generators are designed for Linux, either as Linux kernel modules or user space tools. Compared to kernel modules, user space tools have a limited performance as kernel modules directly operate on the network device driver bypassing the kernel networking subsystem. The Kernel-based Traffic Engine (KUTE) is an example for a Linux kernel module [4]. KUTE is able to send UDP packets and approximately achieves up to 41.4 % link utilization for Gigabit Ethernet links in the authors' investigated test cases. Examples for user space tools are the Brawny and RobUst Traffic Engine (BRUTE), the Real-time UDP Data Emitter (RUDE), the Multi-Generator (MGEN), and the Internet Traffic Generator (ITG) [5][6][7][8]. None of them achieves higher link utilization rates than KUTE does.

Hardware traffic generators mostly base on Field Programmable Gate Arrays due to their flexibility and high performance [9][10]. In [9], the authors state their architecture is scalable from 50 Mbit/s up to 2.5 Gbit/s but do not carry out performance evaluations. The work proposed in [10] aims at evaluating high performance QoS traffic servers rather than achieving full link utilization. There are some approaches combining general-purpose PCs with network processors such as BRUte on Network prOcessor (BRUNO) [11]. However, the authors do not carry out performance evaluations either.

In contrast to the authors' approach, none of the open-source packet generators' performance is sufficient to test PPS with full link utilization. Furthermore, most FPGA-based traffic generators are designed for dedicated purposes or, in the authors' opinion, lack a detailed evaluation of performance for extensive test cases.

Another group of traffic generators are commercial Hardware based generators e.g., from IXIA [12]. Depending on the traffic patterns to be sent, these generators are suitable for testing PPSs with high loads. Most of them can easily be extended for higher generation loads, by adding extension cards with further generation engines. However, these systems are very expensive. Depending on the desired configuration, thousands of dollars up to more than 100,000 dollars have to be spent. Therefore, these well-performing traffic generators are not appropriate for the proposed scenario.

## III. Requirements for a configurable Traffic Generator

On the one hand, traffic generators for testing PPS have to provide a high level of configurability to be able to configure extensive test cases. To exactly reproduce practical traffic conditions, it is necessary to be able to adjust all necessary parameters. Preferably, every single frame should be definable. Thereby, a frame's adjustable parameters depend on the PPS' use case. On the other hand, traffic generators should achieve high traffic rates anyway. Especially if the performance of a PPS shall be evaluated, achievable throughput of the traffic generator is of high importance. Since an optimal combination of these two requirements can only be satisfied by expensive hardware based traffic generators, an FPGA-based traffic generator was developed.

Before the specific requirements for the presented traffic generator will be worked out, its use case shall be briefly explained. The PPS to be tested, is the recently invented IPclip mechanism [2][13]. IPclip is implemented on the access nodes of an internet service provider. It is a mechanism providing Trust-by-Wire in IP-based networks by adding trustworthy location information to IPv4 and IPv6 packets. Thereby, IPclip adds some additional header options to the IP header. These options include location information e.g., a GPS position and some information about the access node. Since the IPclip prototype has already been functionally verified [14], further investigations concerning its performance and the stability of its implementation under very high traffic loads shall be made. Since it inserts additional information to the IP headers and therefore increases packet lengths, it can be necessary to drop a certain percentage of frames in case of full link load. To reproduce these real scenarios, it is mandatory to use test equipment capable of generating traffic fully utilize a 1 Gbit/s Ethernet link. As exposed in the preceding section, only hardware-based traffic generators are able to provide appropriate traffic patterns in conjunction with high configurability.

Link utilization denotes the quotient of achieved throughput (TP) divided by the theoretically possible TP for a link as stated in Formula 1:

$$Link\ Utilization = \frac{Achieved\ TP}{Theoretically\ Max.\ TP} \quad (1)$$

Thereby, TP is the number of bytes traversing a link per second (see Formula 2).

$$TP = \frac{Number\ of\ Bytes}{1s} \quad (2)$$

When generating frames with a traffic generator, the frame size in bytes is an important parameter, since it directly determines the maximum number of frames i.e., headers, which have to be generated. The total number of bytes on a 1 Gbit/s link is calculated as the sum of following values:

- 12 bytes Inter Frame Gap (IFG)
- 7 bytes preamble and 1 byte Start-of-Frame-Delimiter (SFD)
- frame size in bytes
- 4 bytes Frame Check Sequence (FCS)

 ISBN: 978-1-61208-125-0

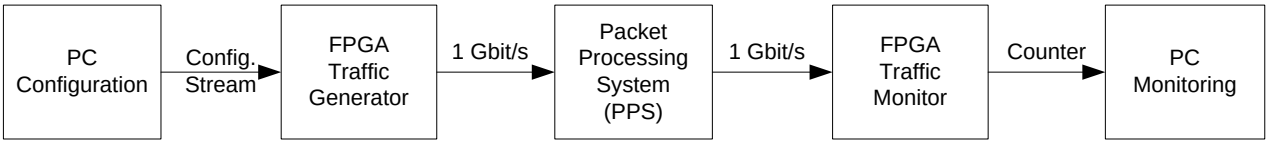


Figure 1. Test setup consisting of PC running the configuration PC, hardware traffic generator, PPS under test, and monitoring HW and PC.

The IFG, preamble, SFD, and FCS are added by transceiver chips and cannot be influenced by the traffic generator. Thus, to calculate the total number of bytes per link, 24 bytes have to be added to the size of each sent frame.

The smallest size an Ethernet frame can have is 60 bytes (without IFG, preamble, SFD, and FCS). That is, 84 bytes are occupied on a 1 Gbit/s Ethernet link for each frame. As we focus on PPS, which take their decisions solely based on header information of frames, processing minimum length frames represents the worst-case. On a 1 Gbit/s Ethernet link, at most approximately 1.5 million smallest size frames can be transmitted per second. That is, the corresponding maximum number of headers to be processed by the PPS. Contrary, only the headers of 81,000 frames have to be processed when sending maximum length frames of 1,514 bytes. Consequently, achieving full link utilization for smallest size frames is a critical task for traffic generators.

To enable the proposed traffic generator to fully utilize a Gigabit link for minimum sized frames, some limitations concerning the configurability of header options had to be made. Considering the IPclip use case, header options have been defined to be configurable based on their relevance for the IPclip mechanism. These fields include IP source address, two VLAN tags, and the length of the to be generated frame.

Summarized, the proposed architecture for a configurable traffic generator shall be able to fully utilize a 1 Gbit/s Ethernet link and be configurable in regard to the IPclip use case. A further major requirement was to be able to reuse the hardware testbench of the developement process of the IPclip system.

## IV. Prototype Realization

To fullfill the above mentioned requirements, an architecture divided into a HW and a SW part was developed.

The software part enables convenient and flexible configuration and only generates values for header options, which shall be variable. The hardware, implemented on an FPGA ensuring throughput of 1 Gbit/s, is configured by this software running on a PC. The main task of the hardware is to generate standard conform frames i.e., IP packets based on these variable header options send from the configuration software. To clarify, the hardware implements a kind of traffic amplification. From a small configuration set, it generates a larger complete frame.

Figure 1 depicts the general setup for the usage of the proposed system architecture and a monitor to measure the output of the PPS to be tested. In the first phase, a software on a PC generates sets of values for the configurable header options. Following, these sets are transmitted via an Ethernet link to the FPGA, which implements the HW part of the traffic generator. The therefore sent frames (see Figure 2) consist of one byte, which determines the operation for the HW. This byte is followed by as many configuration sets as a maximum length frame can contain. The HW implemented on this FPGA is the main functional part of the generator. It receives these configuration sets. From each set it generates one Ethernet frame. These generated frames are mostly fixed in their composition. Only the options, which have to be configurable in order to test the PPS, are generated and sent from the PC. The resulting frames are transmitted on the link to the PPS. After the previously mentioned steps, the task of traffic generation is finished. The PPS under test receives the generated frames. It processes them and transmits the traffic to another FPGA. This one implements a traffic monitor able to count the number of received frames and bytes and collect other information about the incoming data stream. These values are transmitted via a second Ethernet link to a PC. On this PC, a software determines performance metrics of the PPS e.g., regarding frame rate, bandwidth, and packet drops. Only a HW-based monitor was able to correctly measure the traffic under high loads. This especially applies for traffic streams consisting of many minimum length frames.

During the evaluation of the traffic generator itself, nearly the same setup has been used. It only differs in the HW being directly connected with the monitor. Thereby, the parameters of the generated traffic could be measured conveniently. The implemented traffic monitor is not capable of evaluating fine grained statistical information about the traffic. It does not bother with timing data e.g., jitter etc., since this is out of

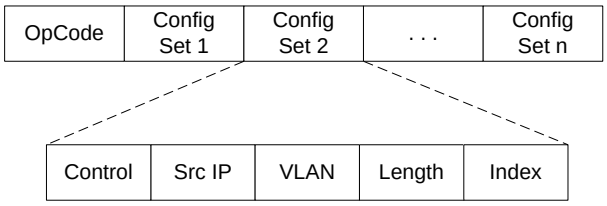


Figure 2. Structure of a configuration frame containing an operation byte followed by as many configuration sets as possible.

 ISBN: 978-1-61208-125-0 

the scope for this work and the envisaged use case of IPclip.

The fundamental idea of this architecture is an amplification of the configuration stream to a data stream of 1 Gbit/s. In order to reduce the amount of data to be generated by and sent from the PC, the configurable header options are as small as possible. An example traffic stream shall be generated, in which frames are 500 bytes long on average. Each frame has a specific frame length (2 byte), source IP address (4 bytes), and include one VLAN tag (2 byte). One control byte is needed to define, which options of the header to be configured with this set. Therefore, a configuration set for one single frame consists of 9 bytes. Resulting from this scenario, amplification of link utilization by a factor of 55 is realized. This example only clarifies the fundamental mechanism. It does not take into account values like inter frame gap, preamble, and start of frame and FCS.

### A. Hardware

This section illustrates the hardware architecture of the traffic generator (see Figure 3). Fundamentally it consists of two finite state machines (FSM). At the input of the traffic generator, an FSM receives frames from the configuration software running on a PC. These frames contain commands from the software part and configuration sets, respectively. All configuration sets are stored into a first in first out memory (FIFO). Since the link to the PPS shall only be fully utilized and no deterministic traffic patterns shall be sent, it does not matter if this memory is full resulting in dropped configuration sets. However, it is more important that the FIFO never runs empty to maintain highest load on the PPS under test. Consequently, the generation of frames does not start until the FIFO is filled for the first time.

The second FSM depicted in Figure 3 generates frames. Therefore, it reads a set from the FIFO. First of all, it processes the control byte determining, which header options to be configured. Based on this information, a frame of the desired length and the configured header options is generated. Consequently, all header options as well as the payload of the frame, which are not configured by the configuration set, are determined statically or randomly. Whether these fields shall be filled with random or static data is determined by the configuration software. Since, the proposed traffic generator shall be able to build frames of many different protocols, there exists corresponding state machines to implement the needed header structures.

Although the proposed architecture is able to realize an enourmous amplification in link utilization, the FIFO containing the configuration sets can run empty. Therefore, two solutions were implemented. The first one is to repeat the last generated frame. To configure this solution, the configuration PC sends a frame before the actual test run starts, which determines how often generated frames shall be repeated. For the envisaged IPclip use case the same frame can be repeated immediately. However, when testing PPSs,

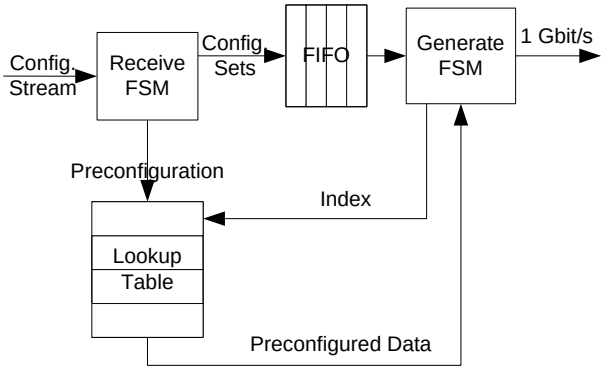


Figure 3. Architecture of the configurable traffic generator's HW.

which implement some caching functionalities, repetition of frames is no appropriate action. This would probably falsify the performance evaluation of those systems. Therefore, a second solution was implemented. Before the generation of frames starts, a configuration stage is introduced. In this phase the configuration software sends frames containing predefined data for long header options or a set of several header options. These predefined data is stored in a buffer within the traffic generator's hardware. During the process of generating frames, these sets of preconfigured data can be selected by an even smaller index, which is part of the variable configuration sets generated by the PC. A possible application of this solution might be the preconfiguration of IPv6 addresses. IPv6 addresses would take 16 byte to be transmitted if they were entirely configurable. It is much more efficient to transmit just an index for a preconfigured lookup table. With a two byte long index, a 65,536 deep table of IPv6 addresses can already be addressed.

### B. Configuration Software

As mentioned at the beginning of this section, a software running on a host PC is used to transmit variable sets of configurable header options to the hardware part of the traffic generator. Therefore, a C++ program using the WinPcap library [15] to transmit the configuration sets over an Ethernet link to the described hardware was implemented. First of all, a protocol for these configuration frames had to be defined. Since the hardware of the traffic generator is directly connected to the configuration PC and therefore receives any frame sent from it, no addressing scheme is needed. Hence, no Ethernet header is required. Thus, it is possible to use as many bytes as possible for configuration sets. However, it is possible to implement an Ethernet and an IP header in order to be able to realize a spatial separation of the configuration PC and the hardware part of the generator. Figure 2 depicts the structure of the configuration frames. The first byte of a frame defines how the receiving state machine handles a frame. For example, a frame sets the replay counter, or writes a preconfigured table, or contains

 ISBN: 978-1-61208-125-0 

Table I
THEORETICAL CONFIGURATION THROUGHPUT (TP) FOR VARIOUS SETS OF CONFIGURABLE PARAMETERS. THE NUMBERS IN THE FRAME LENGTH FIELD DEFINE THE AVERAGE LENGTH OF THE RESULTING FRAMES.

| | Set 1 | Set 2 | Set 3 |
|---|---|---|---|
| Frame length (2 Bytes) | X | X | X |
| IPv4 Address (4 Bytes) | X | | X |
| IPv6 Address (16 Bytes) | | X | |
| VLAN Tag (2 Bytes) | | 2X | |
| LUT Index (2 Bytes) | | X | X |
| Set Size in Bytes | 7 | 25 | 9 |
| Config. TP [Mbit/s] (avg. length 60 Bytes) | 85 | 303 | 109 |
| Config. TP [Mbit/s] (avg. length 500 Bytes) | 14 | 49 | 18 |

a set of configurable options to generate frames. In the usual case of sending configurable options, this operation byte is followed by configuration sets to be written in the afore mentioned operation FIFO. The length of each configuration set depends on the header options it determines. Hence, each set starts with the control byte that indicates the structure of the configuration set i.e., which options it configures. A variable number of configuration sets can be transmitted. Configuration sets are always transmitted using maximum length frames as each Ethernet frame implies an overhead. This overhead results from the inter frame gap, preamble, start of frame delimiter, and the frame check sequence.

Since the mechanism of amplifying configuration bandwidth to full 1 Gbit/s bandwidth was briefly summarized in the beginning of this section, the following description shall investigate it more detailed. To simplify the examinations, a configuration frame is assumed to include parameter sets of the same length. Table I depicts different scenarios of configurable header options.

Each column of the table illustrates a possible configuration set for the IPclip use case. Header options marked with a cross are determined by the configuration PC and therefore transmitted to the FPGA. The table also depicts the length of the resulting configuration set in bytes. The control byte of each configuration set has to be considered for the length of the set as well. Based on the length of a configuration set, the bandwidth of the configuration stream is evaluated, which is necessary to maintain a traffic stream of 1 Gbit/s at the output of the hardware part. Further overhead that has to be considered consists of inter frame gap, preamble, start of frame, and the FCS. Since the length of frames within the generated Ethernet stream has the most important influence on the required configuration bandwidth Table I depicts the required bandwidth once for minimum length frame and as a second example for a resulting Ethernet stream with an average frame length of 500 bytes. As apparent in a traffic stream consisting of longer frames on average, the demands to the configuration bandwidth and therefore to the configuration PC dramatically decreases.

Following, we describe the traffic generator's integration into the hardware development process of PPS. During the functional test of the PPS, before it is synthesized for an FPGA, it has to be functionally verified. Therefore, a testbench, which generates several different frames has already been implemented. This testbench is standard C respectively C++ code enriched with some SystemC code to connect it to the VHDL design process. The testbench's algorithmic part, determining the test patterns, can be reused for the configuration software of the proposed traffic generator. This part generates variable header options to test the PPS. Instead of further building frames from these information transmitted to the PPS's VHDL description within a simulator, variable header options are stored into the previously described configuration sets. These configuration sets are transmitted to the described hardware, where corresponding frames are transmitted to the physical prototype of the PPS.

## V. EVALUATION

In this section, the achieved results for different scenarios are discussed. Furthermore, these results shall be compared with results achieved with packETH. In advance, the main parts of the test equipment have to be mentioned. The described hardware is synthesized for a Xilinx Virtex 4 FPGA (XC4VFX20). As a host PC for running the configuration software a Pentium D at 2.8 GHz with 2 GB RAM running Windows XP SP3 was used. To evaluate the performance of packETH, version 1.6 running on the same PC executing Ubuntu 8.04 is used.

First of all, throughput the software generator packETH achieves was measured for different frame length. As depicted in Table II, the longer the generated frames the higher the throughput of packETH on the Linux PC. As apparent from Table II, packETH does not achieve absolutely full link utilization even for maximum frames. However, for all these cases the proposed traffic generator reaches full link utilization independent from the length of generated frames. As pointed out in Section III, the most important requirement for our use case is to ensure full link utilization. This was not possible with packETH even with an increased process priority to minimize scheduling interrupts.

Further tests for several different configuration scenarios implied by the chosen IPclip use case achieved the same results. Since all the frames transmitted with packETH are configured before it starts transmitting frames, its throughput solely depends on the length of created frames. Therefore, the authors do not illustrate any other packETH results.

The output of the traffic generator only depends on a sufficient configuration bandwidth. Though, performance of the configuration software was evaluated for the worst case of 25 bytes long configuration sets (control byte, IPv6 address,

 ISBN: 978-1-61208-125-0 

Table II
LINK UTILIZATION OF A 1 GBIT/S ETHERNET LINK.

| Frame Size | Link Utilization in % | |
|---|---|---|
| | packETH | Traffic Generator |
| 60 | 18.8 | 100 |
| 120 | 28.8 | 100 |
| 240 | 48.1 | 100 |
| 500 | 81.7 | 100 |
| 1000 | 99.3 | 100 |
| 1514 | 99.0 | 100 |

2 VLAN tags, frame length, index). As the host PC was able to achieve this theoretically evaluated throughput, the traffic generator is able to reach full utilization of a Gbit link for all other scenarios of the IPclip use case. However, the performance of the proposed architecture heavily depends on performance of the configuration PC. First attempts to test the configuration software show that the PC was capable of providing the required configuration bandwidth. However, the hardware part of the traffic generator was not able to constantly generate the desired traffic. Further investigations show that scheduling delays on the configuration PC interrupt the configuration stream for short periods so that the FIFO, which buffers the configuration sets on the FPGA, runs empty. This issue could be solved by increasing the process priority of the configuration software on the PC.

## VI. CONCLUSION

This paper has introduced an architecture for a traffic generator for high throughput performance tests of packet processing systems. To ensure flexibility and a guaranteed full link utilization, the proposed architecture consists of a combination of hardware and software. Furthermore, this architecture integrates well into the hardware design process of packet processing systems by reusing an the therefore implemented testbench. Compared to already existing high performance test equipment, the proposed solution is significantly cheaper as it only requires a medium efficient PC and an FPGA development board including two Ethernet interfaces. As exemplarily shown for packETH, flexible and free software solutions cannot fulfill the authors' requirements to link utilization. This especially applies to the generation of minimum length frames. It could be shown that the developed packet generator achieves full 1 Gbit/s link utilization even for all scenarios implied by the IPclip use case.

## REFERENCES

[1] M. Jemec, "packETH – Ethernet Packet Generator," 2011. [Online]. Available: http://packeth.sourceforge.net/

[2] S. Kubisch, H. Widiger, P. Danielis, J. Schulz, D. Timmermann, T. Bahls, and D. Duchow, "Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP," in *Proc. of 1st ITU-T Kaleidoscope Conference*, May 2008, pp. 375–382.

[3] ZTI, "Traffic Generator and Measurement Tool for IP Networks (IPv4 & IPv6)," 2008.

[4] S. Zander, D. Kennedy, and G. Armitag, "KUTE – A High Performance Kernel-based UDP Traffic Engine," Swinburne University of Technology, Tech. Rep. 050118A, January 2005.

[5] N. Bonell, S. Giordano, and G. Procissi, "BRUTE: A High Performance and Extensible Traffic Generator," in *Proc. of SPECTS*, ser. 37 (3), 2005, p. 839845.

[6] J. Laine, S. Saaristo, and R. Prior, "RUDE & CRUDE: Real-time UDP Data Emitter and Collector," 2008. [Online]. Available: http://rude.sourceforge.net/

[7] Naval Research Laboratory, "The Multi-Generator (MGEN) Toolset," 2008. [Online]. Available: http://cs.itd.nrl.navy.mil/work/mgen/

[8] S. Avallone, A. Pescape, and G. Ventre, "Analysis and Experimentation of Internet Traffic Generator," in *Proc. of New2an'04*, 2005, pp. 70–75.

[9] A. Abdo and T. J. Hall, "Programmable Traffic Generator with Configurable Stochastic Distributions," in *Proc. of Canadian Conference on Electrical and Computer Engineering*, 2005, pp. 747–750.

[10] J. M. Claver, P. Agust, G. Len, and M. Canseco, "A Reprogrammable and Scalable Multimedia Traffic Generator/Monitor on FPGA," in *Proc. of International Conference on Field Programmable Logic and Applications*, 2007, pp. 567–570.

[11] G. Antichi, A. D. Pietro, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "Design of a High Performance Traffic Generator on Network Processor," in *Proc. of 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, 2008, pp. 438–441.

[12] "Ixia - Leader in Converged IP Testing," 2011. [Online]. Available: http://www.ixiacom.com/

[13] H. Widiger, S. Kubisch, P. Danielis, J. Schulz, D. Duchow, T. Bahl, and D. Timmermann, "IPclip: An Architecture to restore Trust-by-Wire in Packet-switched Networks." Montreal, Quebec, Canada: The 33rd IEEE Conference on Local Computer Networks (LCN), October 2008.

[14] P. Danielis, S. Kubisch, H. Widiger, J. Schulz, D. Duchow, T. Bahls, D. Timmermann, and C. Lange, "Trust-by-wire in packet-switched ip networks: Calling line identification presentation for ip (hardware prototype demonstration)." University Booth, Munich, Germany: DATE 2008, March 2008.

[15] G. Varenni, L. Degioanni, F. Risso, and J. Bruno, "WinPcap, The Packet Capture Library and Network Monitoring Library for Windows," 2010. [Online]. Available: http://www.winpcap.org/

 ISBN: 978-1-61208-125-0 