Precise IPv4/IPv6 Packet Generator Based on NetCOPE Platform

Jiří Matoušek, Pavol Korček
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, 612 66, Czech Republic
xmatou06@stud.fit.vutbr.cz, ikorcek@fit.vutbr.cz

***Abstract*—This paper presents an architecture of a hardware network packet generator designed for the COMBOv2 cards using the NetCOPE development platform. The packet generator internal structure allows synthetic IPv4 and IPv6 network packet creation as well as the real network packet transmission. Based on COMBOv2 add-on interface card, the generator is able to transmit packets at speed of 2x10 Gbit/s or 4x1 Gbit/s. Synthetic data are created using high quality pseudo-random number generator. If desired, previously captured network packets can be transmitted back to the network with exactly the same time drift as were captured. This is accomplished by extremely precise timestamps, which can be generated by the Timestamp Module included in the NetCOPE platform.**

***Index Terms*—Packet Generator, IPv6, NetCOPE, COMBOv2, Pseudo-Random Numbers, Timestamp.**

I. Introduction

Generating network traffic that reflects different network conditions and topologies is critical for performing valid experiments in network testbeds. Any new network component, protocol or design requires extensive and accurate testing in sufficiently realistic settings. While network simulation tools can be very helpful in understanding the impact of a given change to a network, their predictions might not be accurate due to their simplified and restricted models and settings. Generating synthetic and real network traffic is becoming crucial. Moreover, it becomes important to generate network packets with high precision. The precise emission of network packets can very accurately simulate different topologies and can also reflect the behaviour of the network very realistically.

Towards this issue, our paper presents an architecture of a network packet generator for new generation of IPv6 Internet infrastructure. Similarly, the generator can be used in mixed or in strict IPv4 networks. Our highly precise generator is based on the COMBOv2 cards [1] utilizing the NetCOPE platform [2]. This platform is generally used for rapid development of hardware accelerated network applications. But until now, it has only been used for applications aimed to capturing, filtering or monitoring of the high-speed networks at the wire speed.

II. Related Work

The most common way of generating network traffic is to use open source software tools for capturing packets and their subsequent replay (e.g. `tcpdump` [3] for capturing and `tcpreplay` [4] for replaying). The advantage of this approach is the use of ordinarily available software and hardware components. Unfortunately, generic hardware components can, depending on a vendor, vary in their behaviour and therefore the output traffic might be also very inaccurate as shown for example in [5]. Another disadvantage of this approach is also a very low throughput. For high-speed networks, it is impossible to use this approach for generating network traffic at full wire speed mainly because of the system IP stack.

Another possibility of generating network traffic is represented by commercial hardware network traffic generators. They are powerful tools for generating network traffic at full wire speed. These hardware generators provide broad options for setting the properties of generated traffic, so they are useful for many kinds of network tests and experiments. Their disadvantages include mainly very high price and also their proprietary nature makes them very inflexible for the research on new techniques and protocols. As the representants of this category, we can mention systems from Spirent [6] or stochastically based hardware generators from Ixia [7]. Ixia systems allow the users to create and save synthetic traces to be rerun in the future. These systems can be useful, however, they do not allow replaying of previously captured real traffic. Also, it has been shown that properties of this kind of devices are not accurate enough for some experiments [8].

To the best of our knowledge, there is only one similar network packet generator [9]. This Stanford University Packet Generator (SPG) is based on a so-called the NetFPGA platform. It uses 4x1 Gbit/s hardware card with the FPGA chip and a quite small on-board memory. Packets to be sent by SPG must first be loaded into the platform's memory from a PCAP file stored on a host, and only after that they can be transmitted to the network. This two-stage process means that SPG can only replay short previously captured traces. The largest memory on the board is 64 MB which is about only 0.5 second of traffic at the speed of 1 Gbit/s. Using on-board memory for buffering network traffic rises from the need of sending packets to the card over the PCI bus. There is a 33 MHz 32-bit bus on the NetFPGA platform with a theoretical top transfer rate of 1056 Mbit/s, but there is a significant overhead even in a host where the bus is not shared. Most importantly, the number of DMA transfers between the driver and the platform is limited such that the total throughput is

978-1-4244-9756-0/11/$26.00 ©2011 IEEE

only 260 Mbit/s when individually transferring 1518 B long packets. Some software improvements were made to SPG to increase its ability to generate longer sequences at full wire speed of 1 Gbit/s [5]. However, these improvements caused sending only zero data in packets.

Our work differs from the previous approach in two important aspects. Firstly, for sending packets from a host to a card, we are using the NetCOPE DMA controllers which can ensure, jointly with PCI Express bus, high throughput. It allows using our packet generator without any on-board data buffering in configuration with the 4x1 Gbit/s interface card. Secondly, the NetFPGA based generator expects captured network traffic. Our generator can also be used in a different way, where no real network traffic is available, by virtue of generating network traffic with the high speed synthetic network packet generator used in our architecture.

III. NetCOPE Platform on COMBOv2 Cards

The family of COMBO cards was designed by CESNET [10] and dedicated for building hardware accelerated network applications. The main idea of this family of cards is the maximum flexibility provided both by the mother card extendable by a set of add-on cards and by the FPGA chip on the mother card. Thanks to using add-on cards, it is possible e.g. to change a number of network interfaces or to connect an external precise clock signal generator. On the other hand, the FPGA chip allows changing of COMBO card's functionality while maintaining the benefits of hardware acceleration.

Currently, second generation of the COMBO cards family (COMBOv2) is in use [11]. The mother card of this generation is the PCI Express x8 v1.1 card equipped with the Xilinx Virtex-5 LXT/FXT FPGA chip for a hardware accelerated application, and the Xilinx Spartan-3E FPGA chip for rebooting the mother card on the fly. Equipment of the mother card includes among others the DDR2 SODIMM memory (up to 2 GB) as well as the Low Speed Connectors and the high speed Interface Connectors for connection of add-on cards. From the set of add-on cards we use interface cards (in configuration 2x10 Gbit/s or 4x1 Gbit/s) and COMBOL-GPS card providing precise clock (CLK) signal and pulse per second (PPS) signal based on information gathered from a GPS receiver.

For the rapid development of hardware accelerated network applications on the COMBOv2 cards, the NetCOPE platform has been developed. This platform consists of the firmware for the FPGA chip on the mother card and the software for accessing the card's functionality from the operating system.

A. Hardware

Firmware of the NetCOPE platform comprises three main components: the Generic Interconnection System (GICS) [12], the DMA Module and the Network Module [13]. The user application is tied with all three modules as shown in Fig. 1.

Main components of the NetCOPE platform provide the most important functionality for the user application — connection to the PCI Express bus (implemented by GICS),

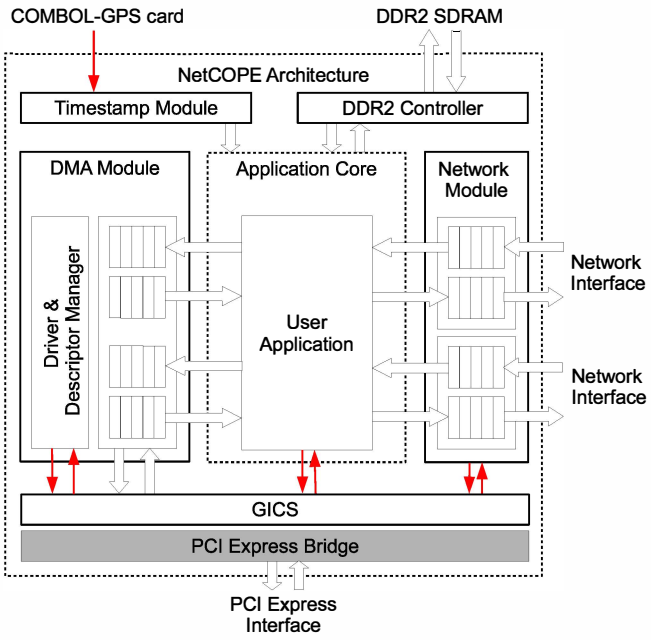

Fig. 1. Firmware of the NetCOPE platform

hardware support for fast DMA transfers (the DMA Module) and simple and uniform access to network interfaces independent of their type (the Network Module). Except these integral parts of the NetCOPE platform, there are also some expansion modules providing extra functionality — the Timestamp Module able to generate precise 64-bit hardware timestamps (thanks to signals from COMBOL-GPS card) and the DDR2 Controller making available memory placed in the DDR2 SODIMM connector on COMBOv2 mother card.

B. Software

The NetCOPE platform software layer handles the control of the COMBOv2 cards and allows the user to use functionality implemented as a part of this platform in the FPGA. Software layer of the NetCOPE platform is built hierarchically and it can be divided horizontally (device drivers, libraries and tools) or vertically (basic input/output operations and support for the fast DMA transfers) [14]. Software provided together with the NetCOPE platform can thus be used to control any user application without any futher changes.

IV. NetCOPE Platform Performance

Since related documents about the COMBOv2 cards family and the NetCOPE platform (e.g. [2] and [1]) describe only high performance operation in receive (RX) direction, we had to examine its performance in transmit (TX) direction, mainly between the host and the card. The reachable throughput from the RAM on host computer to the COMBOv2 internal memory has been tested using DMA transfers over the PCI Express x8 link. Two different configurations of the host computer were chosen, as shown in Tab. I. First one is a standard server platform and second a powerful gaming platform.

TABLE I
CONFIGURATION OF TESTED HOST COMPUTERS

Parameter	1st Configuration	2nd Configuration
CPU	2 x Xeon®CPU E5420 @ 2.50 GHz	Intel®Core™i7 CPU 920 @ 2.67 GHz
Mainboard	Supermicro X7DB8	Supermicro X8STE
Chipset	Intel®5000P	Intel®X58

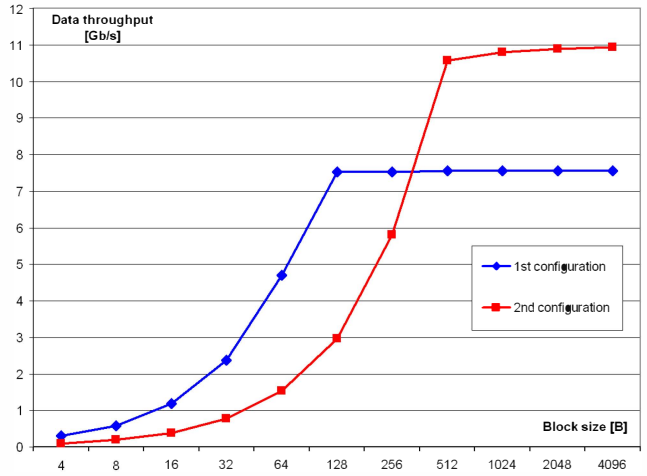

Fig. 2. Host RAM to FPGA throughput test results — FPGA design working frequency: 218.75 MHz, MaxReadRequestSize (PCI Express parameter): 2048 B

All other parameters not shown in Tab. I, were kept unchanged. It is especially the host RAM memory (4 GB) and the operating system version (32-bit Linux-2.6.26.3). DMA engine runs in our test design at 218.75 MHz, and one data transfer bulk consists of $20 * 10^3$ transfers. Each test was performed with a different data block size from 4 Bytes up to PCI Express maximum of 4 kB (4096 Bytes). Measurement was performed by utilizing the FPGA internal counter, which started to count before and stopped immediately after all the transfers were done. The results are shown in Fig. 2. It clearly demonstrates that two different configurations reached highly different results. By this test, it was shown that it is not possible to reliably generate packets from software on an arbitrary server platform. For higher packet rates it is necessary to use the DDR2 memory on COMBOv2 card for storing packets (capture & replay scenario). Another possibility, if there is no sufficient platform, is generating the packets synthetically without fruitless loading of PCI Express bus.

V. PRECISE PACKET GENERATOR ARCHITECTURE

Our Precise IPv4/IPv6 Packet Generator is based on the NetCOPE platform and fully uses functionality provided by platform modules. The generator represents a user application for the NetCOPE platform and it consists of newly implemented modules depicted in Fig. 3. This is the architecture for one network interface. In designs with multiple network interfaces the basic architecture is repeated for each network interface and some other minor changes are made.

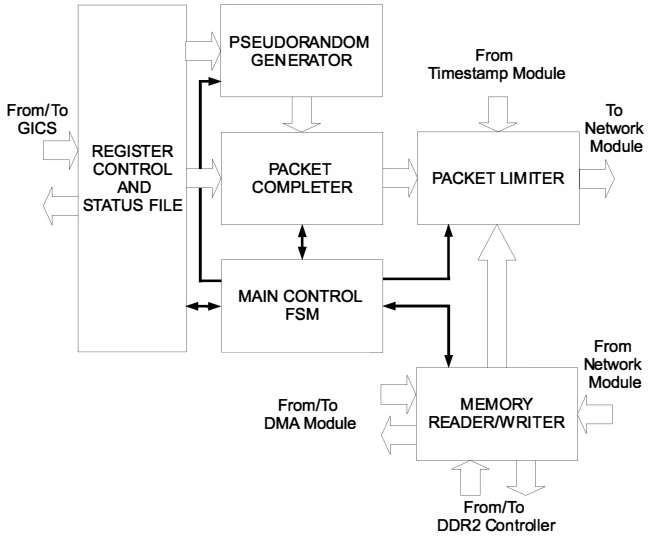

Fig. 3. Precise Packet Generator architecture for one network interface

The main function is to generate synthetic IPv4/IPv6 network traffic at wire speed. For this purpose, the generator architecture includes high quality pseudo-random number generator. In addition to generating synthetic network traffic, our generator is able to replay previously captured real network traffic at speed of 2x10 Gbit/s or 4x1 Gbit/s. In this case, the DDR2 memory present on the COMBOv2 mother card is used for storing network traffic, which is then transmitted via chosen network interface. Content of the DDR2 memory can be loaded either from a PCAP file stored in the host memory or directly from the network interface. Because of the possibility in generating very precise hardware timestamps, it allows to transmit previously captured network traffic with exactly the same time drift as packets were captured.

When the generator is not in operation, the system works as a standard network interface card. In such a case, it is possible to use software tools for capturing and replaying network traffic (`tcpdump`, `tcpreplay`). As we have shown in the previous paragraph, on some platforms it is possible to transfer data from the host memory to the COMBOv2 card with a bit rate higher than 10 Gbit/s. Thanks to this, we could transmit data via one 10 Gbit/s network interface at full wire speed. Unfortunately, this mode of operation cannot guarantee wire speed for both 10 Gbit/s interfaces of an add-on card. To overcome this, DDR2 memory must be used as described above.

A. Pseudo-Random Generator

Our pseudo-random number generator is based on LFSR, which can effectively be implemented in FPGAs [15]. This type of generators are based on a shift register with feedback function. The input of the feedback function is driven by two or more bits from registers. Bit positions are given by the primitive polynomial coefficients, also called taps. The computed output value is fed to the input of whole shift register (Fibonacci scheme) or alternatively to the input of

TABLE II
SCORE FROM DIEHARD TEST (LOWER IS BETTER) [17]

Generator Type	Effective Score
true	22
MLFSR	154
LFSR	756

next register (Galois scheme). For more details, see [16]. As the feedback function, exclusive-OR or its negation is often chosen.

LFSRs are easily implementable in the FPGA, but, on the other hand, generated sequence of numbers has not very good properties, which can be shown e.g. by so-called serial test [17]. To overcome this issue, a new architecture of multiple LFSR used in parallel (MLFSR) was proposed and firstly introduced in [16]. This improved version of LFSR can successfully pass even the Diehard battery test of randomness [18]. Table II shows effective score from Diehard of simple LFSR, MLFSR and also true random generator, where the randomness comes from atmospheric noise [19]. Because of these very good properties, MLFSR was used as a pseudo-random number generator in our packet generator.

Generating synthetic data in IP network packets often requires values from predefined interval (e.g. for IP addresses, TCP/UDP port numbers, etc.). We used normalization for number conversion to desired interval. For fast FPGA implementation, division in normalization was done by shifting and multiplication utilizes hard DSP48 slices. It is also important to note, that MLFSR is capable of generating zero values [16]. This could not be normally reached with a standard LFSR (not allowed state).

B. Packet Completer

In this module of the packet generator, all data forming the Ethernet frame of IPv4 or IPv6 packet are put together into one Ethernet frame. These data include mainly fields of IPv4 or IPv6 header, which can be either generated using pseudorandom generators or specified as constants.

First of all, the completer needs to know the version of IP protocol. When this is clear, it acquires the information about the header fields which are generated and which are constant. Based on this knowledge, it chooses generated value or constant value for each header field.

By default, the payload of the IP packet is filled by pre-defined data pattern. Nevertheless, it is also possible to generate IP packet's payload similarly as IP header fields. In both cases, the length of the payload can be constant or generated as a number from a sequence or as a random value. The Ethernet header for already formed IP packet is added and whole Ethernet frame is sent to the Packet Limiter using the FrameLink protocol.

C. Memory Reader/Writer

This module of the packet generator is dedicated to take care mainly about communication with the memory on the COMBOv2 mother card (through the DDR2 Controller), where

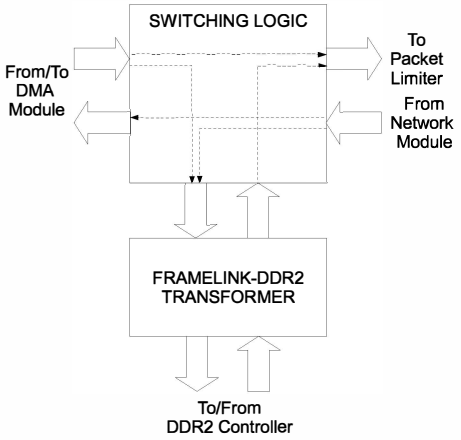

Fig. 4. Memory Reader/Writer internal structure

real network traffic is stored. Another task of this part is to ensure correct interconnection between three different types of data interface (the Network Module interface, the DMA Module interface and the DDR2 Controller interface), all of them are full-duplex. As Fig. 4 shows, there is the FrameLink-DDR2 Transformer for transforming data from the FrameLink protocol (similar to Xilinx LocalLink protocol [20]) used on the Network Module interface and the DMA Module Interface to format used on the DDR2 Controller interface and vice versa. Second submodule implements the interconnection between data interfaces requested in different modes of generator's operation. When the generator is not in operation, the DMA Module interface is connected directly to the Network Module interface. Input of the DDR2 memory interface can be connected to both the DMA Module and the Network module, because storing real network traffic to the memory is possible from the host PC memory as well as directly from the network interface. Output memory interface is connected to the Network Module interface in order to transmit stored real network traffic to the network. Before the transmitted data reach the Network Module, they must pass through the Packet Limiter, which can be either active or inactive.

D. Packet Limiter

This part of the generator plans and controls the data transmission. Structure of this module is depicted in Fig. 5. Processing of incoming data, as well as the module they come from, depends again on selected mode of operation.

During transmission of previously captured real network traffic, the data come from the Memory Reader/Writer module. These data can contain timestamps which can be used for delaying transmission of packet until designated time. For such operation, it is necessary to edit packets' timestamps in software (to shift them forward in time) before packets are loaded back to the COMBOv2 card's memory. In the Packet Limiter, timestamps are extracted from packets and each component is stored in separate FIFO. Packet is then kept in the FIFO until the corresponding timestamp does not match the currently generated one. If all timestamps were shifted the

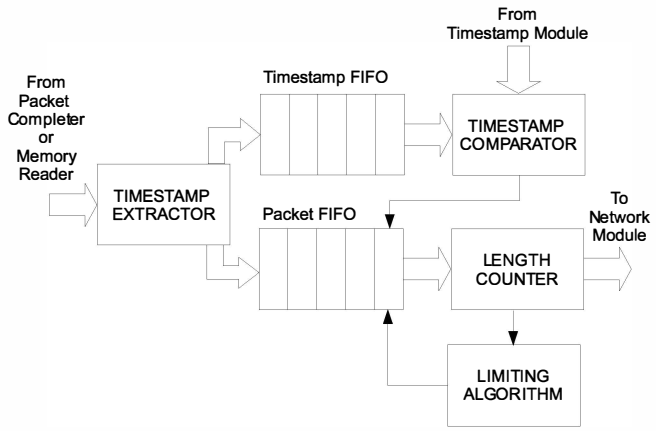

Fig. 5. Packet Limiter internal structure

same time, this mechanism ensures transmission of packets with exactly the same time drift as they were captured.

In the same mode of operation, as well as during generation of synthetic network packets, it is possible to limit transmission of packets to the specified bit rate. This functionality is provided by limiting algorithm. Based on the selected bit rate and a length of the last transmitted packet, this algorithm insert delay before transmission of next packet, so the average link bit rate is at the specified value before next transmission starts. Length of previously transmitted packet is determined during its transmission through the length counter.

Transmission of packets can also be unrestricted, i.e. done at the full wire speed. This is allowed in each mode of operation.

E. Register Control and Status File

Set of registers included in this module serves as a place to store all control and status information of the packet generator. Most of the generator modules use the information stored in registers belonging to this component. A brief description of registers of different modules follows.

A significant number of registers belong to the Pseudo-random Generator. For each field of IPv4/IPv6 header, there is place to store three important values for generating field's content. These values are called *from*, *to* and *increment_size*. Data for the IP headers are then generated starting at value *from* with given *increment_size* and restarting at value *to*. The information whether IPv4 or IPv6 header fields are generated is also present. Another part of the register set is used by the Main Control FSM. This module controls other components of the generator design, so information about the actual mode of operation is crucial for its function. Some registers are dedicated also to the Packet Completer. It has to know which fields of IPv4/IPv6 header are generated by Pseudorandom Generator and which should be considered as constant. The value of such constant field is then expected to be stored in the register dedicated for value *from*. The last part of the control register file belongs to the Packet Limiter. For this module of the generator, there is stored the information about required packet output bit rate. This can be specified either absolutely or relatively.

F. Main Control FSM

To control the operation of all packet generator modules, the Main Control FSM was implemented. Based on selected mode of operation and also on feedback from controlled components, it controls these components and ensures their correct settings according to the actual mode. These modes include generating synthetic network traffic, loading data to the DDR2 memory either from a PCAP file stored in the host memory or from the network interface, transmitting previously captured network traffic from the DDR2 memory to the network interface and operating as a standard network interface card.

Most settings take place in the Memory Reader/Writer, where the interconnection of data interfaces must change in almost each mode. Another important component to control is the Packet Limiter, where two different types of packet transmission limitation can take place. According to an actual mode of operation and also on presence of timestamps, the Main FSM must select an appropriate way of limitation. When the limitation according to timestamps is selected, the Main Control FSM must also ensure a generation of actual timestamps for comparison with those recorded during packet's capturing.

VI. Generator Evaluation

The Precise IPv4/IPv6 Packet Generator is together with the NetCOPE platform targeted primarily for the COMBOv2 mother card with Virtex-5 XC5VLX155T FPGA chips. It also expects attached COMBOL-GPS card and one of the inteface add-on cards (4x1 Gbit/s or 2x10 Gbit/s). It will also be possible to implement the generator on a different network device, if the NetCOPE platform is implementable on it. Nevertheless, in further text we consider its implementation on the COMBOv2 card.

Proposed packet generator was designed in compliance with the main idea of COMBOv2 cards family — maximum flexibility and scalability. It is therefore possible to easily implement it for configurations with different number of network interfaces. The user is also allowed to select preferred implementation of pseudo-random generators and to adjust size of the Packet and the Timestamp FIFOs in the Packet Limiter module. Because all of these nonfixed parameters of the system we present only upper limit of device utilization, which should not be exceeded in any configuration. The values presented in the Tab. III refer to design for one network interface. The expected device utilization of designs with multiple network interfaces can be computed simply by multiplication of these values by number of network interfaces. For comparison and also in order to clarify total device utilization (together with the NetCOPE platform), values for empty NetCOPE design are also shown in Tab. III.

Based on information presented in Tab. III, it is clear that the packet generator together with the NetCOPE platform can be implemented on the COMBOv2 mother card even in configuration with four network interfaces. The most resource consuming parts of the generator's design are the Pseudo-Random Generator and the Register Control and Status File.

TABLE III
DEVICE UTILIZATION OF VIRTEX-5 XC5VLX155T

Resource	Slices	BRAMs
Precise Packet Generator (one interface)	3104	12
NetCOPE platform (2x10 Gbit/s interfaces)	10378	44
Summary	13482	56
Available	24320	212
Utilization (%)	44.56	26.42

TABLE IV
COMPARISON OF PROPOSED PACKET GENERATOR AND PACKET GENERATOR ON THE NETFPGA PLATFORM

Property	Proposed Packet Generator	NetFPGA Packet Generator
10 Gigabit Ethernet support	YES	NO
SW based traffic replaying	YES	NO
DRAM based traffic replaying	YES	YES
Synthetic traffic generation	YES	NO
IPv6 support	YES	YES
Output rate limitation	YES	YES
Timestamp based transmission	YES	YES

This is due to need for generating/storing some information for each IPv4 and IPv6 header field and therefore it would be very difficult to omptimize device utilization caused by these two modules. On the other hand, the least device utilization is caused by the Main Control FSM and the Packet Completer.

Another important parameter is maximal operation frequency. Requirements on generator's frequency raises mainly from connection to the Network Module. Interface of this module contains 64-bit wide data paths with clock signal at frequency of 156.25 MHz. This corresponds to the definition of XGMII protocol [21], where frequency is the same, data path is half of width and DDR transferring is used. This operation frequency is easily achievable for the proposed packet generator.

If we are looking for some similar application, the only one comparable possibility is represented by the packet generator implemented on the NetFPGA platform [9]. To evaluate benefits of our solution, we compare some basic characteristics of these two FPGA-based packet generating applications. Results of this comparison are shown in Tab. IV.

This comparison clearly shows, that proposed packet generator beats that one built on the NetFPGA platform mainly by supporting faster Ethernet standard and by possibility to generate synthetic network traffic. On the other hand, we have to admit, that except the support for generation of synthetic network traffic, all other advantages of our solution are based on differences in selected implementation platforms.

VII. CONCLUSION

In this paper, we have presented an architecture of the packet generator based on the COMBOv2 cards and the NetCOPE platform. Our generator is able to replay previously captured network traffic either directly from software (up to rate 10 Gbit/s) or with usage of an on-board DDR2 memory (at full speed of 2x10 Gbit/s). Another possibility is to generate synthetic network traffic at full speed of 2x10 Gbit/s. Thanks to a possibility in generating extremely precise hardware timestamps, the generator is also able to replay captured network traffic with exactly the same time drift as it was captured.

ACKNOWLEDGMENT

This work was partially supported by the Grant Agency of the CzechRepublic under contract No. GD102/09/H042 Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems and by the Research Plan No. MSM0021630528 Security Oriented Research in Information Technology and by the FIT-S-11-1 project.

REFERENCES

[1] J. Novotný and M. Žádník. (2008) COMBOv2 — hardware accelerators for high-speed networking. [Online]. Available: http://www.liberouter.org/docs/2008-02-10_COMBOv2.Academic_Forum.pdf

[2] T. Martínek and M. Košek, "NetCOPE: Platform for rapid development of network applications," in *Proc. of 2008 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*, Apr. 2008, pp. 1–6.

[3] Tcpdump. (2010) Web site of tcpdump and libpcap. [Online]. Available: http://www.tcpdump.org/

[4] tcpreplay developers. (2010) tcpreplay website. [Online]. Available: http://tcpreplay.synfin.net/trac/wiki/tcpreplay

[5] G. Salmon, M. Ghobadi, Y. Ganjali, M. Labrecque, and J. G. Steffan, "NetFPGA-based precise traffic generation," in *Proc. of NetFPGA Developers Workshop'09*, 2009.

[6] Spirent Communications. (2011) Spirent homepage. [Online]. Available: http://www.spirent.com/

[7] Ixia. (2011) Ixia homepage. [Online]. Available: http://www.ixiacom.com/

[8] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, J. Naous, and G. Salmon, "Performing time-sensitive network experiments," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, Apr. 2008, pp. 127–128.

[9] G. A. Covington, G. Gibb, J. Lockwood, and N. McKeown, "A packet generator on the NetFPGA platform," in *17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009. FCCM '09*, Apr. 2009, pp. 235–238.

[10] CESNET. (2011) CESNET homepage. [Online]. Available: http://www.cesnet.cz/

[11] ——. (2009) Our hardware website. [Online]. Available: http://www.liberouter.org/hardware.php?flag=U

[12] T. Málek, T. Martínek, and J. Kořenek, "GICS: Generic interconnection system," in *2008 International Conference on Field Programmable Logic and Applications*, Sep. 2008, pp. 263–268.

[13] J. Matoušek, "Implementation and verification of network interface blocks," bachelor's thesis, FIT BUT, Brno, 2009.

[14] The Liberouter Project Team. (2010) NetCOPE platform handbook. [Online]. Available: http://www.liberouter.org/netcope/handbook.html

[15] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudo-random sequence generators," Application Note, Xilinx, Inc., Jul. 1996. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf

[16] P. L'Ecuyer and F. Panneton, "A new class of linear feedback shift register generators," in *Winter Simulation Conference Proceedings, 2000*, 2000, pp. 690–696.

[17] P. Korček, "Pseudorandom number generation in FPGA," bachelor's thesis, FIT BUT, Brno, 2007.

[18] G. Marsaglia. (1995) Diehard battery of tests of randomness. [Online]. Available: http://www.stat.fsu.edu/pub/diehard/

[19] M. Haahr. (2011) RANDOM.ORG homepage. [Online]. Available: http://www.random.org/

[20] Xilinx Inc. (2010) LocalLink user interface. [Online]. Available: http://www.xilinx.com/products/ipcenter/LocalLink_UserInterface.htm

[21] *Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3, 2005.