



**Repositorio Institucional de la Universidad Autónoma de Madrid**

<https://repositorio.uam.es>

Esta es la **versión de autor** del artículo publicado en:

This is an **author produced version** of a paper published in:

IEEE Communications Magazine 54.3 (2016): 80 –87

**DOI:** <http://dx.doi.org/10.1109/MCOM.2016.7432152>

**Copyright:** © 2016 IEEE

El acceso a la versión del editor puede requerir la suscripción del recurso

Access to the published version may require subscription

# Accurate and affordable packet-train testing systems for multi-Gb/s networks

Mario Ruiz, Javier Ramos, Gustavo Sutter,  
Jorge E. López de Vergara, Sergio López-Buedo, Javier Aracil



**Abstract**—Communication networks face nowadays a relentless increase in traffic load. Multi-Gb/s links are becoming widespread, and network devices are under a continuous stress, so testing whether they guarantee the specified throughput or delay is a must. Software-based solutions, such as the packet-train traffic injection, were adequate for lower speeds, but they have turned inaccurate in the current scenario. Hardware-based solutions have proved to be very accurate, but usually at the expense of much higher development and acquisition costs. Fortunately, the new affordable FPGA SoC devices, as well as high-level synthesis tools, can very efficiently reduce these costs. In this paper we show the advantages of hardware-based solutions in terms of accuracy, comparing the results obtained in an FPGA SoC development platform and in NetFPGA-10G to those of software. Results show that a hardware-based solution is significantly better, especially at 10 Gb/s. By leveraging high-level synthesis and open source platforms, prototypes were quickly developed. Noticeable advantages of our proposal are the high accuracy, the competitive cost with respect to the software counterpart, which runs in high-end off-the-shelf workstations, and the capability to easily evolve to upcoming 40 Gb/s and 100 Gb/s networks.

## 1 INTRODUCTION

OUR day-to-day activities are becoming more and more dependent on communication networks: Social networks, mobile apps, e-commerce, etc. Such widespread use of communications has implications at both the access and server sides of the network. At the access side, it drives the development of faster access technologies, such as Ten Gigabit Passive Optical Network (10G-PON) or fourth generation (4G) and beyond mobile networks. At the server side, it causes exponential increases in network traffic being faced by data centers. As a result, network testing is nowadays more necessary than ever, because networking equipment is over-stressed due to the huge amount of traffic. However,

network testing becomes a complex and expensive task at this traffic load operating point.

We note that the main measurement parameters for assessing the quality of Internet access services are, according to [1], upload and download speed, Packet Loss Rate (PLR), delay and jitter. As link speed grows, measuring these parameters is not only more difficult, but also calls for testing devices that must provide unprecedented accuracy. For example, the transmission of a minimum-size Ethernet frame takes only 67 ns at 10 Gb/s. Apart from the difficulties of measuring at such small timescales, switching time is no longer negligible, making it imperative to include the switching equipment within the test scope as well.

In order to measure these network parameters on a Device Under Test (DUT) —such as a network switch or router— or a set of chained DUTs —such as a network path— we propose using the packet-train technique. This technique has proven to be effective and highly immune to interferences such as cross-traffic load at the end-user equipment [2]. Unfortunately, current software tools are severely constrained when it comes to performing this type of measurements at high speeds (e.g. 10 Gb/s), even if they run at kernel level in the operating system.

In this paper we show the shortcomings of software-based solutions and how to overcome such limitations using a hardware-based solution. The new Field Programmable Gate Array (FPGA) System on a Chip (SoC), which combines a powerful microcontroller and a programmable logic fabric, can be used to develop accurate and affordable testing devices for high-speed networks. Moreover, we also show how open-source platforms and High-Level Synthesis (HLS) can efficiently offset the well-known difficulties of FPGA development.

Certainly, the benefits of using FPGAs in multi-Gb/s networks are well known. However, previous works in the network testing area are scarce and mainly focus on replacing configurable traffic generators [3], [4]. Probably the closest proposal to ours is [5], which uses FPGAs to test networking equipment according to RFC2544 [6]. However, such work considers only 1 Gb/s networks, and no previous state-of-art considers accurate time synchronization mechanisms such as GPS, which is needed to accurately measure one-way delay and jitter in a distributed way. Finally, the benefits of using high-level languages for networking applications

*All authors are with the Department of Electronics and Communication Technologies, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Francisco Tomás y Valiente, 11, 28049 Madrid, Spain*

*The last three authors are also with Naudit High Performance Computing and Networking, a spin-off company of the latter university.*

*E-mail addresses: {mario.ruiz,javier.ramos,gustavo.sutter}@uam.es, {jorge,sergio,javier}@naudit.es*

*This work was partially supported by the Spanish Ministry of Economy and Competitiveness under the project PackTrack (TEC2012-33754) and by the European Union through the Integrated Project (IP) IDEALIST under grant agreement FP7-317999*

*Manuscript submitted July 31st, 2015. Revision received December 1st, 2015.*

*Acceptance December 7th, 2015*

are beginning to be recognized, as shown in [7] where such methodology has been used to implement a 10 Gb/s TCP/IP stack in FPGA.

Hence, we propose the use of novel FPGA SoCs as a means for comprehensively testing network equipment with packet trains. The most remarkable novelties of this work are: First of all, we show that very accurate results can be obtained at both 1 and 10 Gb/s, using two proof-of-concept designs. Secondly, FPGA SoCs can be used to implement very cost-effective testing appliances, featuring a minimal component footprint and a reduced power consumption, which could easily be deployed across the whole network. Moreover, we also quantify how inaccurate software-based solutions can be at multi-Gb/s speeds, unless the corresponding hardware aid comes into play. Finally, we also show the benefits of open-source platforms and high-level synthesis in order to reduce FPGA development time and cost, thus making programmable logic competitive to software in terms of design productivity.

The rest of this paper is structured as follows. First, we show two network testing use cases where the presented solutions have proven useful. Then, the packet-train measurement technique is explained. Next, both the software and hardware approaches are described in detail. After that, a performance evaluation follows, whereby both implementations are compared and discussed. Finally, some conclusions and an outlook of future works are provided.

## 2 USE CASES

The range of application of high-speed network testing tools is very diverse. The typical use case focuses on testing the capabilities of network equipment. However, this testing can also be extended to other scenarios, where it is necessary to perform distributed measurements along a network path. Moreover, such testing must be performed continuously in time to monitor the network QoS parameters. Next subsections provide two examples where this type of distributed continuous testing has been applied, apart from the usual testing of network equipment.

### 2.1 Service level verification

Nowadays we are witnessing a fierce competition between operators to provide more bandwidth in the residential access link for the less possible price. In this competition for market share the regulators are playing their role as referees to enforce a given quality of service level.

Of particular interest is the case of bandwidth reselling between operators, at the access and metro link level. There, the regulator must ensure that the Quality of Service (QoS) provided by the incumbent operator to the hiring operator meets a QoS level that allows the transmission of interactive multimedia services. Since the number of potential users in the metro network is very large and the residential bandwidth is growing at a significant pace, we note that the metro network switches, working at multi-Gb/s data rates, must be carefully tested both before deployment and also during operation.

### 2.2 Measurement of next-generation elastic optical network equipment

Elastic optical networks are being developed to offer the possibility of dynamically changing the signal modulation format and/or the spectrum allocation of optical data links. This dynamic reconfiguration capability paves the way for new operation models, whereby links are no longer statically provisioned, but dynamically adapted to traffic demands. Therefore, the link capacity must be continuously monitored in these elastic optical networks, to check if the underlying network has reconfigured the provided bandwidth or not.

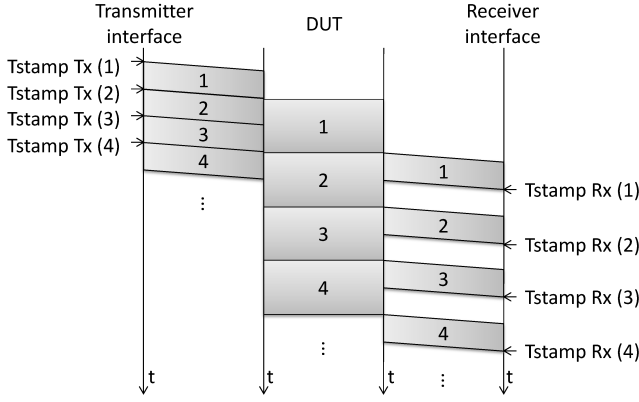
## 3 PACKET-PAIR AND PACKET-TRAIN TECHNIQUES FOR NETWORK MEASUREMENTS

Once we have motivated the need of high-speed network testing, we propose to use the packet-train technique, which is an evolution from the previous packet-pair technique. Packet-pair [8] is an active measurement method based on sending multiple packet pairs from a source to a destination endpoint in order to estimate the corresponding QoS parameters. Each pair is composed of equally sized packets sent back-to-back at the maximum allowed speed in a link or end-to-end path. At the receiver side, packet dispersion is analyzed to estimate the capacity. As it turns out, packet-pair techniques are prone to both capacity underestimations and overestimations due to interfering traffic, because only two packets are used in the measurements. However, packet-trains [9] provide better accuracy and robustness, simply because more packets are involved in the measurement and the resulting train is less sensible to cross-traffic interference than the corresponding pair.

In packet-train techniques, a group of  $N$  packets is sent back-to-back from a sender to a receiver and the average dispersion of the  $N$  packets is used to calculate the capacity, as shown in Figure 1. Additionally, One-Way Delay (OWD), jitter and PLR may also be estimated by including timestamps and sequence numbers on the packets. Increasing the number of packets in the train provides immunity against interfering traffic but also increases the measurement load in the measured network. We note that this technique is based on flooding the link, and, consequently, the measurement time must be kept at a minimum, not to interfere with the rest of traffic. Typically, train lengths range from 100 to 1000 packets. Regarding packet sizes, OWD is better measured using minimum-sized packets to reduce the impact of the transmission time on the estimation.

## 4 SOFTWARE-BASED SOLUTIONS

Traditionally, network measurements have been performed using specialized hardware designed for such a task. In the recent years, several software-based solutions that run on top of commercial off-the-shelf (COTS) systems have also been applied for network measurement and testing tasks. The latter provide a cost-effective and flexible solution for the development of network testing probes. For instance, *pktgen* [10] is a Linux kernel module that enables the generation of traffic with different packet headers and payloads



**Figure 1.** Representation of the packet-train technique.

defined by user (source and destination MAC and IP addresses, UDP ports, etc.) and also with specific statistical features, namely inter-arrival time and number of flows. Additionally, this kernel module adds a sequence number and the departure timestamp for each packet, which makes this tool suitable for throughput, OWD, jitter and PLR measurements. The main drawback is that the departure timestamp is taken in the Linux kernel and not in the Network Interface Card (NIC) itself, which adds measurement noise due to the transit time from the Linux kernel to the NIC. In high-speed links (10 Gb/s and beyond), we note that more packets per second must be copied to the kernel, so the measurement noise is more significant. Thus, all traditional software traffic generators cannot exactly mimic the transmission pattern defined by the user, which severely biases the measurement in a high-speed scenario, as stated in [11]. Even if a real-time operating system is used, the interruption timer accuracy is in the order of milliseconds, that is far too coarse for 10 Gb/s networks.

In addition, vanilla network drivers cannot cope with minimum-sized packets at 10 Gb/s rates neither in transmission nor in reception, which is essential for testing. Recently, high-speed network engines have been developed [12] to solve this issue. For instance a software traffic generator called *PktGen-DPDK* built on top of Intel's DPDK<sup>1</sup> framework has recently been released. Such traffic generator is able to transmit either random generated packets or PCAP traces. Although this traffic generator provides 10 Gb/s rates, it cannot add sequence numbers or transmission timestamps to the packets, so it is not able to measure OWD. For this reason, only the Linux *pktgen* module will be considered later in our comparative analysis.

## 5 HARDWARE-BASED SOLUTIONS

For years, the use of programmable logic devices (more specifically, FPGAs) has democratized hardware design for low volume users. Nevertheless, the complexity of designing specialized hardware still resides in the FPGA design flow, which is based on the demanding Hardware Description Languages (HDL). To circumvent this issue, several promising HLS (High-Level Synthesis) tools are appearing. HLS typically uses C/C++ as design entry, instead of the lower-level RTL (Register Transfer Level) descriptions used

by HDLs. HLS tools not only improve design productivity, but also bring FPGA technology closer to networking engineers.

In order to develop a hardware-based solution that implements the packet-train technique exploiting HLS design tools, we have worked with two hardware platforms. The first one is used to demonstrate the feasibility of building accurate, affordable, low power and portable 1 Gb/s network testing appliances on top of a FPGA SoC device. The second one, provided as proof-of-concept for 10 Gb/s networks, is based on the NetFPGA<sup>2</sup> project. For simplicity, we will name them HwP1 and HwP2 respectively.

### 5.1 HwP1: ZedBoard

ZedBoard<sup>3</sup> is a low cost board based on a Xilinx Zynq SoC (an XC7Z020-CLG484-1) device that encompasses in a single device a microcontroller based on a dual core ARM Cortex-A9 (named as PS, Processing System) and an FPGA (named as PL, Programmable Logic). The board has plenty of input/output connectors, standing out an FPGA Mezzanine Card (FMC) connector that allows plugging complex peripherals to the system. Furthermore, an operating system such as Linux can be run in the PS, thus enabling complex applications to be developed. Besides, the PL is based on a modern and fast FPGA technology (Xilinx 7-Series), which allows building complex hardware peripherals in the form of hardware modules (also known as IP-Cores, from Intellectual Property core). To develop the project, two external Gigabit Ethernet interfaces were used, connected to the PL through the FMC connector.

### 5.2 HwP2: NetFPGA-10G

NetFPGA is an open hardware and software project, developed by Stanford and Cambridge Universities in collaboration with Xilinx. It is intended for rapid prototyping of computer network devices. The NetFPGA-10G is based on Xilinx Virtex-5 FPGA (an XC5VTX240TFFG1759-2). It provides four SFP+ interfaces and has an 8X PCI Express Gen 1 interface to the host. Even though it can work standalone, it is typically attached to a host PC, as in our testbed. To ease the development process, we leveraged on the current NetFPGA environment, using the Open Source Network Tester (OSNT) project [13] as development framework.

### 5.3 Hardware architectures description

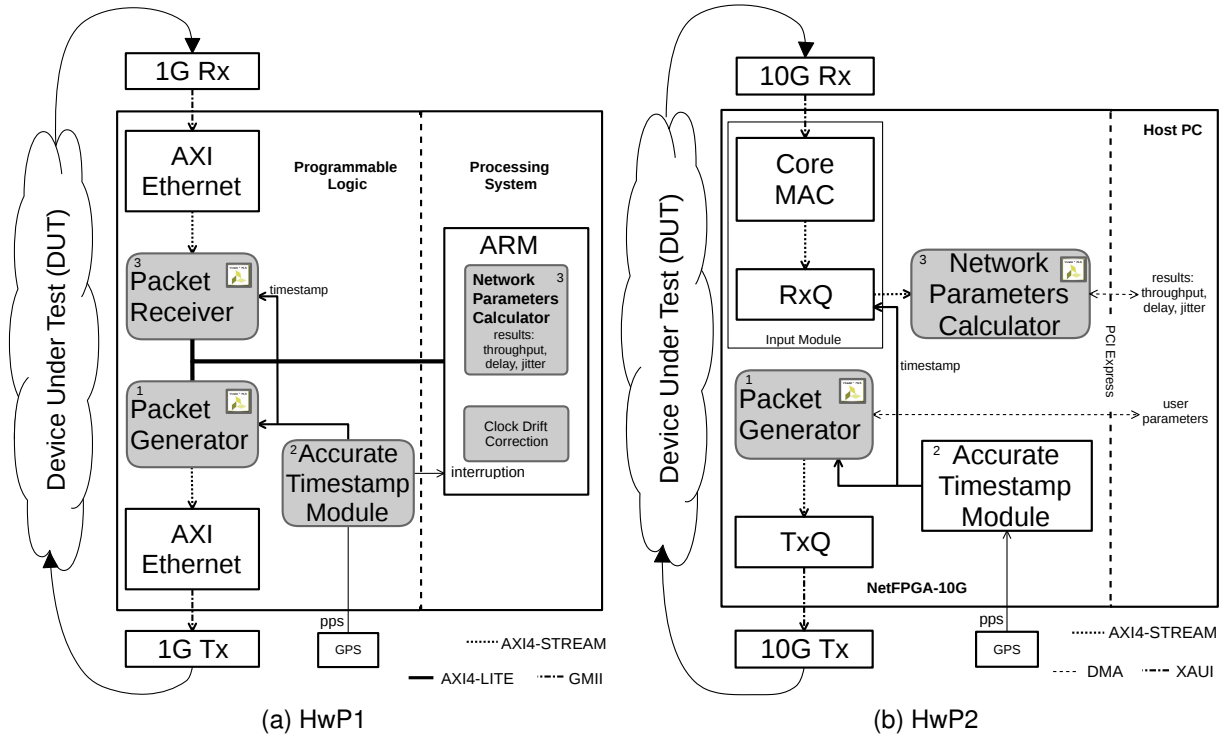
With the aim of speeding up the hardware development cycle and bringing it as close as possible to the network application engineers, we have used HLS tools to design the prototypes, as described in [14]. Our tool of choice is Vivado-HLS<sup>4</sup>, which generates synthesizable HDL code from a C/C++ source along with synthesis directives. On the other hand, modules where timing is critical (operations need to be done in an exact number of clock cycles) were implemented using the traditional FPGA design flow based on HDLs (VHDL or Verilog).

2. <http://netfpga.org/>

3. <http://zedboard.org/>

4. <http://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html>

1. <http://dpdk.org/browse/apps/pktgen-dpdk/>



**Figure 2.** Implemented hardware architectures. (a) ZedBoard at 1 Gb/s. (b) NetFPGA and OSNT at 10 Gb/s.

The designs communicate with the external Physical Layer (PHY) chip by means of an AXI4-Stream bus. The difference between both prototypes is in the size of the bus transactions and its frequency of operation (32 bits at 100 MHz for HwP1, and 256 bits at 156.25MHz for HwP2).

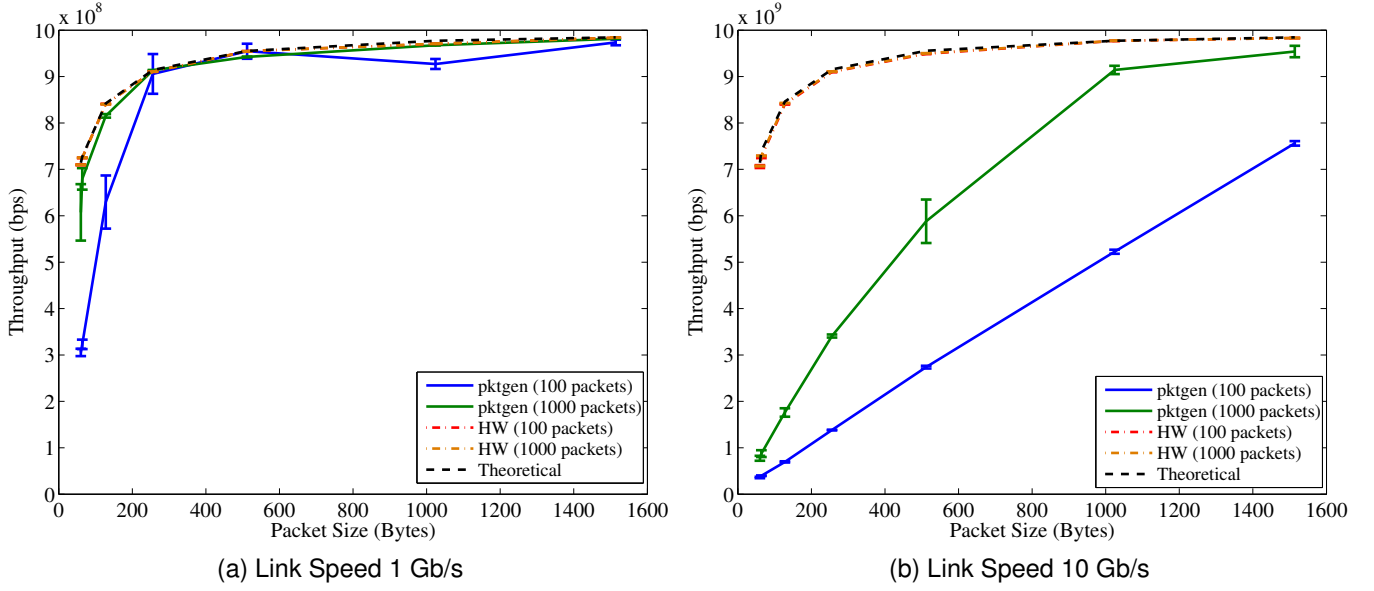
In both architectures, depicted at Figure 2, there are two key IP-Cores with similar behavior in the heart of the system. On the one hand, the **Packet Generator** (1) was developed for both systems with the same HLS code, the only difference being the AXI4-Stream size and frequency of operation. Customizable options include source and destination MAC and IP addresses, UDP ports, packet size and train length. In addition, each generated packet contains a sequence number, and it is timestamped as close as possible to the physical layer chip. On the other hand, the **Accurate Timestamp Module** (2) is in charge of correcting the clock drift. Both implementations use the Pulse Per Second (PPS) signal from a GPS receiver as a reference to compensate the clock drift. At HwP1, this module is split in two parts following a hybrid hardware-software approach: the first part is a variable-rate counter implemented in hardware, which runs at 100 MHz. The second part is an algorithm that runs in the ARM processor. Such algorithm uses the sum of the previous errors to correct the rate of the counter, and sends it back to hardware. Therefore, we can obtain a remarkable timestamp resolution of 10 ns, with an extremely low clock drift, thanks to the GPS-based error compensation. At HwP2, we have leveraged on the OSNT project functionality, which implements a Direct Digital Synthesizer (DDS) to correct the clock drift [13]. The design operates at 156.25 MHz, with a 6.4 ns resolution.

There are some differences in the architectures mainly because HwP1 has a tightly coupled processor near the FPGA (enabling a hybrid hardware-software approach, as mentioned before) and HwP2 provides a framework with a library of pre-designed modules. In HwP1, following the hybrid hardware-software approach, the **Packet Receiver** (3), developed in HLS, receives the packets and filters them according to user-defined rules. Then, a software program running on the ARM processor computes the **Network Parameters**. On the other hand, HwP2 uses the infrastructure available at the NetFPGA framework for packet reception and a custom-developed HLS module to compute the **Network Parameters** (3).

In summary, our designs are good to send packet trains with the aim of measuring the required quality parameters (delay, jitter, loss and throughput). Additionally, both designs use a PPS signal from an external GPS in order to support one-way delay and jitter measurements when testing in a distributed infrastructure. The code of both projects is freely available at GitHub<sup>5</sup>.

In the near future, we expect to port our implementation to a low cost 10 Gb/s version based on Zynq SoC. Finally, it is worth remarking that both hardware implementations occupy less than 55 % of most of the available resources in the FPGA. Absolute figures of used resources in both solutions are shown in Table 2.

5. [https://github.com/hpcn-uam/hardware\\_packet\\_train](https://github.com/hpcn-uam/hardware_packet_train)



**Figure 3.** DUT throughput with different packet sizes. Mean and standard deviation estimated both in software and hardware.

## 6 PERFORMANCE EVALUATION

### 6.1 Evaluation testbeds

Two different performance evaluation scenarios have been considered for both software and hardware solutions. The first scenario, used for calibration, is based on sending measurement packets through an interface and receiving them in another interface of the same testing device, in a loopback fashion. The second scenario is based on sending the measurement packets through an interface that is connected to the DUT—in our case, a Cisco Catalyst 2960-S. In the DUT, the measurement traffic is forwarded from one SFP+ port to another SFP+ port that is connected to the traffic receiver interface of the testing device. This scenario addresses the 10 Gb/s case. For the sake of completeness, the performance analysis has been repeated for the 1 Gb/s case, using the same setup but connected to 1 Gb/s DUT ports.

To evaluate the software solution, the *pktgen* module has been executed on a server running an Ubuntu Linux 14.04 with a 3.16.0 kernel. The server has two Intel Xeon E5-2620 processors with 6 cores each, 32 GB of RAM and an Intel 82599 10 Gb/s NIC. For all tests the *ixgbe* vanilla driver has been used along with *pktgen* 2.75 module.

All the experiments featured packet trains of 100 and 1000 packets with frame sizes of 60, 64, 128, 256, 512, 1024 and 1514 bytes, excluding frame preamble and check sequence. The experiment was repeated ten times to obtain mean and standard deviation for throughput and OWD. In the case of *pktgen* module, traffic has been generated using a single transmission queue since packet sequence numbers must be correlative for throughput and delay measurements, and using multiple queues produces packet disorder [15].

### 6.2 Experimental results

The throughput measurements in the 1 Gb/s scenario using the testing setup with the DUT are shown in Figure 3a. As it can be observed, results obtained with the hardware HwP1 prototype are fairly close to theoretical throughput value and the standard deviation is very small. In the case of measurements using the software approach, if the train length is 100 packets and the packet sizes are lower than 256 bytes, the results are pretty far from the theoretical values, and present larger deviations. With sizes of 256 bytes and above, we observe that the empirical values approach the theoretical ones. For 1000-packet trains, measurements are more accurate and present less deviation.

Figure 3b shows the throughput measurements for 10 Gb/s. As in the 1 Gb/s case, the results for the hardware systems are very similar to the theoretical values, with low deviation. However, the results for the software systems significantly depart from the theoretical ones, but improve as the packet size and train length increase. Similar results were obtained in the calibration setup, measuring the loopback for 1 and 10 Gb/s scenarios.

Note that, for some scenarios, regulatory bodies require that link measurements are performed using minimum-sized packets, which implies that software solutions are unfeasible. Additionally, a thorough network device testing should generate traffic with packet sizes ranging from the minimum size to the MTU.

Regarding OWD measurements, Table 1 shows the estimation results for both loopback and switch setups at 1 Gb/s and 10 Gb/s. As it can be observed, software measurements are far from the theoretical values, adding up to 150  $\mu$ s of error in the worst-case scenario—loopback at 1 Gb/s. If accuracy below thousands of  $\mu$ s is needed, the hardware solution is the most suitable option. It is also worth noting that the hardware approach not only shows the most accurate results, but it also presents extremely low

Packet Size (bytes)	# of packets	1Gb/s				10Gb/s			
		Loopback		Switch		Loopback		Switch	
		OWD pktgen ( $\mu$ s)	OWD HwP1 ( $\mu$ s)	OWD pktgen ( $\mu$ s)	OWD HwP1 ( $\mu$ s)	OWD pktgen ( $\mu$ s)	OWD HwP2 ( $\mu$ s)	OWD pktgen ( $\mu$ s)	OWD HwP2 ( $\mu$ s)
60	100	29 $\pm$ 5	1.890 $\pm$ 0.003	38 $\pm$ 14	5.149 $\pm$ 0.003	17 $\pm$ 6	0.883 $\pm$ 0.000	17 $\pm$ 8	3.79 $\pm$ 0.003
	1000	30 $\pm$ 7	1.889 $\pm$ 0.000	34 $\pm$ 5	5.149 $\pm$ 0.001	17 $\pm$ 5	0.883 $\pm$ 0.000	17 $\pm$ 6	3.80 $\pm$ 0.002
64	100	33 $\pm$ 9	1.941 $\pm$ 0.002	34 $\pm$ 6	5.236 $\pm$ 0.003	17 $\pm$ 7	0.883 $\pm$ 0.000	18 $\pm$ 5	3.81 $\pm$ 0.001
	1000	30 $\pm$ 5	1.945 $\pm$ 0.003	33 $\pm$ 4	5.235 $\pm$ 0.002	18 $\pm$ 6	0.884 $\pm$ 0.000	18 $\pm$ 6	3.80 $\pm$ 0.002
128	100	34 $\pm$ 5	2.772 $\pm$ 0.001	43 $\pm$ 6	6.720 $\pm$ 0.005	20 $\pm$ 5	0.945 $\pm$ 0.000	20 $\pm$ 3	3.97 $\pm$ 0.007
	1000	37 $\pm$ 5	2.773 $\pm$ 0.003	44 $\pm$ 6	6.728 $\pm$ 0.002	28 $\pm$ 1	0.946 $\pm$ 0.000	29 $\pm$ 2	3.97 $\pm$ 0.008
256	100	53 $\pm$ 8	4.437 $\pm$ 0.005	59 $\pm$ 8	9.425 $\pm$ 0.003	35 $\pm$ 8	1.069 $\pm$ 0.000	34 $\pm$ 7	4.20 $\pm$ 0.001
	1000	53 $\pm$ 8	4.438 $\pm$ 0.007	59 $\pm$ 7	9.426 $\pm$ 0.002	35 $\pm$ 8	1.069 $\pm$ 0.000	34 $\pm$ 9	4.20 $\pm$ 0.000
512	100	83 $\pm$ 13	7.763 $\pm$ 0.003	92 $\pm$ 13	14.793 $\pm$ 0.004	59 $\pm$ 12	1.317 $\pm$ 0.000	58 $\pm$ 13	4.65 $\pm$ 0.000
	1000	84 $\pm$ 11	7.765 $\pm$ 0.006	92 $\pm$ 11	14.796 $\pm$ 0.002	61 $\pm$ 11	1.317 $\pm$ 0.000	60 $\pm$ 10	4.65 $\pm$ 0.000
1024	100	122 $\pm$ 11	14.423 $\pm$ 0.002	134 $\pm$ 11	25.556 $\pm$ 0.003	90 $\pm$ 11	1.812 $\pm$ 0.000	89 $\pm$ 10	5.56 $\pm$ 0.001
	1000	123 $\pm$ 3	14.424 $\pm$ 0.002	135 $\pm$ 4	25.555 $\pm$ 0.002	91 $\pm$ 4	1.812 $\pm$ 0.000	92 $\pm$ 8	5.56 $\pm$ 0.000
1514	100	170 $\pm$ 17	20.798 $\pm$ 0.001	172 $\pm$ 47	35.854 $\pm$ 0.003	130 $\pm$ 16	2.322 $\pm$ 0.064	129 $\pm$ 15	6.48 $\pm$ 0.000
	1000	171 $\pm$ 6	20.796 $\pm$ 0.001	188 $\pm$ 9	35.853 $\pm$ 0.000	132 $\pm$ 6	2.317 $\pm$ 0.007	131 $\pm$ 9	6.48 $\pm$ 0.000

**Table 1.** Switch and loopback estimated OWD with different packet sizes and link speeds. Mean and standard deviation.

variation (less than 0.01%) which makes it well suited for jitter measurements.

### 6.3 Calibration

We note that the measured latency in the hardware developments is significantly larger than the theoretical minimum (frame transmission time) in the loopback scenario. This is due to the different elements in the transmission chain. In the 10 Gb/s case, the FPGA features three IP-cores that add up latency to transmission and reception: 10G MAC core, 10G Attachment Unit Interface (XAUI) core and multi-gigabit transceivers (MGT). Particularly, the reception MGT has an elastic buffer in order use the same clock for transmission and reception, so that the design is simplified. Such elastic buffer will also add uncertainty to the latency measurement. Moreover, we note that this buffer is not the only source of latency; the 10G MAC and XAUI cores can add up to 200 ns (adding transmission and reception latencies). Nevertheless, the main source of latency in the NetFPGA-10G board is the physical medium chip, which performs a conversion from XAUI to the 10 Gb/s serial electrical interface, as well as Electronic Dispersion Compensation (EDC). Such operations add a significant latency to the transmission/reception path. Similar considerations should be taken into account for 1 Gb/s case.

Considering the results of Table 1, we can empirically infer that the aggregate delays (due to the reasons discussed above) linearly depend on the packet size. Therefore, we can use the loopback scenario to extract a calibration function from the delay measurements. To obtain such a function, we have firstly represented the scatter plot of measured data set (70 points) as shown in Figures 4a for 1 Gb/s and 4b for 10 Gb/s. As a second step we have fitted the data using a linear regression, as it is the simplest method to calculate such function. Finally, we have subtracted this function from the theoretical one to obtain the calibration function. Such calibration function has been later applied to the Catalyst 2960-S switch delay measurements, obtaining comparable results to those reported<sup>6</sup> for a similar device.

6. <http://miercom.com/pdf/reports/20130917.pdf>

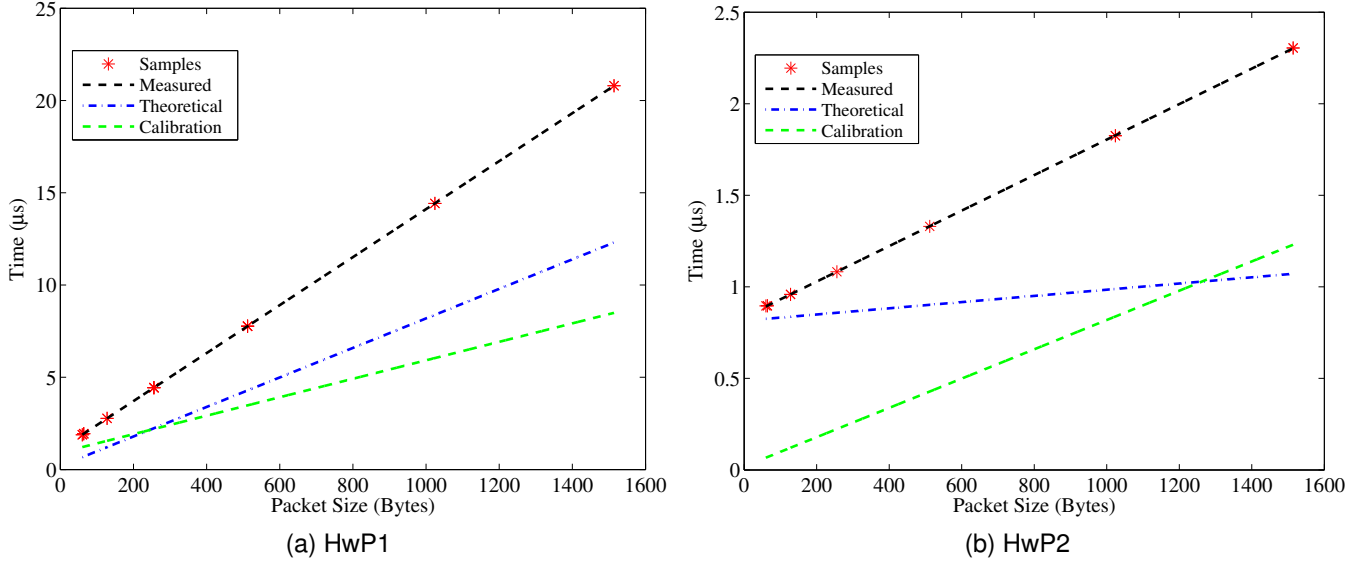
## 7 CONCLUSIONS AND FUTURE DIRECTIONS

The increasing speed of communication networks poses a serious challenge for testing. In this paper we presented the advantages of FPGA technology to implement accurate and affordable testing appliances for high-speed networks. Although software-based solutions are undoubtedly the most convenient approach in terms of deployment and development costs, we have shown that the non-determinism of software severely limits the accuracy of such solutions at multi-Gb/s speeds. Moreover, non-determinism arises in the NIC itself and its connection to the system (PCIe, chipset), so using GP-GPU accelerators or multi-/many-core architectures does not help to solve such an issue.

The solution proposed in this paper is using the new FPGA SoC devices. On the one hand, the FPGA fabric can be used to implement an accurate network measuring system. But an accurate network measuring is not enough to create a fully-fledged network testing appliance. A microprocessor where to run the testing software and its underlying operating system is needed. This is where the high-performance microcontroller embedded in the FPGA SoC device plays an essential role, providing a hybrid hardware-software approach. On the other hand, FPGA SoCs only require a reduced number of external components in order to create a complete appliance, and its price and power consumption is low enough to consider a massive deployment of these appliances in the whole network. For example, the price of a Xilinx XC7Z015 device is around 100 USD, and this device has the capability of implementing a 10 Gb/s Ethernet network port by means of a XAUI interface.

In order to assess the benefits of FPGA measuring systems for high-speed networks, we have developed two proof-of-concept designs. The first one runs at 1 Gb/s and is based on the ZedBoard development platform. The second one runs at 10 Gb/s on top of NetFPGA-10G. We have chosen the packet-train technique to perform the measurements due to its well-known features of accuracy, interference immunity and low network overhead during testing. With the help of these two proof-of-concept designs, we have shown the advantages of hardware solutions in terms of





**Figure 4.** Regression on hardware platforms to calibrate delay measured. (a) ZedBoard at 1 Gb/s. (b) NetFPGA at 10 Gb/s.

Feature	SW	HwP1	HwP2
Approximate cost	5000 USD	1400 USD	3500 USD
Power consumption	160 W	8.5 W	120 W
FIFO Blocks (FIFO36)	N/A	86 out of 140	11 out of 324
DSP Blocks (DSP48E)	N/A	0 out of 220	2 out of 96
Number of Slices	N/A	8210 out of 13300	28996 out of 37440
Slice LUTs	N/A	22224 out of 53200	79099 out of 149760
Slice Registers	N/A	24589 out of 106400	81646 out of 149760
Timestamp resolution	10 μs	10 ns	6.4 ns
Designability	Easy	Moderate	Moderate
Design time	Weeks	Months	Months
Engineer skills needed	Drivers	FPGA	FPGA
Maximum rate supported	10 Gb/s	1 Gb/s	10 Gb/s
Measure throughput 1 Gb/s	poor	✓	✓
Measure throughput 10 Gb/s	✗	N/A	✓
Measure OWD 1 Gb/s	✗	✓	✓
Measure OWD 10 Gb/s	✗	N/A	✓

**Table 2.** Features summary of software and hardware prototypes.

determinism and accuracy when compared to the software alternative.

Table 2 provides a qualitative and quantitative summary of this hardware vs. software comparison. As it was evidenced in Figure 3, software solutions only provide a good accuracy for throughput measurements at 1 Gb/s and when using large packet sizes. At 10 Gb/s the software accuracy is very poor, even if using Kernel-level approaches such as the one evaluated in this paper.

In Table 2, it is stated that the development time for the FPGA prototypes has been just months. It is well known that FPGA development is very costly, not being uncommon to have development times reaching one year. Another contribution of this paper is to present the benefits of open-source platforms and high-level synthesis for improving FPGA design productivity. We have shown how high-level synthesis could significantly ease the development of the network parameter calculation core. Additionally, using an open-source platform as a starting point, the development time of the 10 Gb/s prototype may be significantly reduced.

Regarding costs and power consumption, the results presented in Table 2 are the corresponding to the prototypes

that have been evaluated in this paper. For the software solution, a high-end server system with a 10 Gb/s Ethernet card was used. For HwP1, the configured system includes the ZedBoard card, a GPS receiver and the Gigabit Ethernet FMC card. For HwP2, a NetFPGA-10G card was used (academic price), along with a low-end computer attached to it. It is expected that such costs will be much lower when producing the network testing appliances at a large scale. Finally, the power consumption has been measured when performing the network measurements with packet trains. In HwP2, the NetFPGA-10G consumes less than 30 Watts standalone.

In the near future, we will see a widespread use of 40 and 100 Gb/s networks. Given the results obtained for software solutions at 10 Gb/s, we envision that using dedicated hardware will be a must for testing such networks, moreover if we consider the nanosecond-accuracy required at those speeds. Fortunately, FPGA SoC devices will be able to implement these testing solutions based on dedicated hardware. At the moment, devices such as Xilinx XC7Z030 are capable of implementing both 10 and 40 Gb/s interfaces at a cost roughly below 400 USD. For the case of future



100 Gb/s networks, the new generation of Zynq UltraScale+ devices will provide direct a connection to CFP or QSFP28 cages, while, at the same time, maintaining the low power consumption and moderating price features of the current generation of FPGA SoC devices.

## REFERENCES

- [1] Body of European Regulators for Electronic Communications, "Monitoring quality of Internet access services in the context of net neutrality," Tech. Rep. BoR (14) 117, Sep. 2014.
  - [2] J. Ramos, P. M. Santiago del Río, J. Aracil, and J. E. López de Vergara, "On the effect of concurrent applications in bandwidth measurement speedometers," *Computer Networks*, vol. 55, no. 6, pp. 1435 – 1453, Apr. 2011.
  - [3] A. Tockhorn, P. Danielis, and D. Timmermann, "A configurable FPGA-based traffic generator for high-performance tests of packet processing systems," in *6th International Conference on Internet Monitoring and Protection (ICIMP)*, Mar. 2011, pp. 14–19.
  - [4] T. Groléat, M. Arzel, S. Vaton, A. Bourge, Y. Le Balch, H. Bougdal, and M. Aranaz Padrón, "Flexible, extensible, open-source and affordable FPGA-based traffic generator," in *Proceedings of the First Edition Workshop on High Performance and Programmable Networking*, ser. HPPN '13, Jun. 2013, pp. 23–30.
  - [5] Y. Wang, Y. Liu, X. Tao, and Q. He, "An FPGA-based high-speed network performance measurement for RFC 2544," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 2, Jun. 2015.
  - [6] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices," RFC 2544 (Informational), Internet Engineering Task Force, Mar. 1999.
  - [7] D. Sidler, G. Alonso, M. Blott, K. Karras, K. Vissers, and R. Carley, "Scalable 10Gbps TCP/IP stack architecture for reconfigurable hardware," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, May 2015, pp. 36–43.
  - [8] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.*, vol. 2, Jan. 2001, pp. 905–914.
  - [9] A. Johnsson, "On the comparison of packet-pair and packet-train measurements," in *Proc. Swedish National Computer Networking Workshop*, Sep. 2003, pp. 241–250.
  - [10] R. Olsson, "Pktgen the linux packet generator," in *Proceedings of the Linux Symposium, Ottawa, Canada*, vol. 2, Jul. 2005, pp. 11–24.
  - [11] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?" *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 158–165, Sep. 2010.
  - [12] V. Moreno, J. Ramos, P. Santiago del Río, J. García-Dorado, F. Gómez-Arribas, and J. Aracil, "Commodity packet capture engines: Tutorial, cookbook and applicability," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 3, pp. 1364–1390, aug 2015.
  - [13] G. Antichi, M. Shahbaz, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. McKeown, N. Feamster, B. Felderman, M. Blott et al., "OSNT: Open source network tester," *Network, IEEE*, vol. 28, no. 5, pp. 6–12, Sep. 2014.
  - [14] M. Forconesi, G. Sutter, S. López-Buedo, J. E. López de Vergara, and J. Aracil, "Bridging the gap between hardware and software open source network developments," *Network, IEEE*, vol. 28, no. 5, pp. 13–19, Sep. 2014.
  - [15] W. Wu, P. DeMar, and M. Crawford, "Why can some advanced Ethernet NICs cause packet reordering?" *Communications Letters, IEEE*, vol. 15, no. 2, pp. 253–255, Feb. 2011.
- Mario Ruiz** (mario.ruiz@uam.es) is currently a PhD student in HPCN group at Universidad Autónoma de Madrid. He received his Electronic Engineering degree from Universidad Nacional de San Juan (UNSJ), Argentina in 2014. He has been teaching assistant at the UNSJ (2012-2014). His research interest include FPGA hardware design for high-speed network, algorithm acceleration and development of embedded system. He is co-author of one book and one international paper.
- Javier Ramos** (javier.ramos@uam.es) received the M.Sc. degree in computer science and the Ph.D. degree in computer science and telecommunications from the Universidad Autónoma de Madrid, Madrid, Spain, in 2008 and 2013, respectively. Currently he works as post-doc researcher at Universidad Autónoma de Madrid. His research interests are on the analysis of network traffic, quality of service, software defined networks and network function virtualization.
- Gustavo Sutter** (gustavo.sutter@uam.es) MSc degree in Computer Science from UNCPBA University in Tandil (Buenos Aires) Argentina, in 1997, and PhD degree from Universidad Autónoma de Madrid (UAM) Spain, in 2005. He has been assistant professor at the UNCPBA and is currently an associate professor at UAM, Spain. His research interests include algorithms and networking in reconfigurable computing, digital arithmetic, embedded and High Performance Computing. Author of three books and more than hundred papers and communications.
- Jorge López-de-Vergara** (jorge.lopez\_vergara@uam.es) is associate professor at Universidad Autónoma de Madrid (Spain), and founding partner of Naudit HPCN, a spin-off company devoted to high performance traffic monitoring and analysis. He received his MSc and PhD degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1998 and 2003, respectively. He researches on network and service management and monitoring, having co-authored more than 100 scientific papers about this topic.
- Sergio López-Buedo** (sergio.lopez-buedo@uam.es) is associate professor at Universidad Autónoma de Madrid (Spain). FPGA technology is his main research interest, especially high-performance reconfigurable computing and communication applications. He was a visiting researcher at University of British Columbia (2005), The George Washington University (2006-2007), and University of Cambridge (2015). Dr. Lopez-Buedo holds 3 patents and more than 50 publications, and he is also co-founder of the spin-off company Naudit HPCN.
- Javier Aracil** (javier.aracil@uam.es) received the M.Sc. and Ph.D. degrees (Honors) from Technical University of Madrid in 1993 and 1995, both in Telecommunications Engineering, and the Licenciatura (five-years degree) in Mathematics from UNED in 2009. In 1995 he was awarded with a Fulbright scholarship to pursue post-doctoral research at University of California, Berkeley. He is currently a full professor at Universidad Autónoma de Madrid, Spain and a co-founder of the spin-off company Naudit HPCN.