OSNT: Open Source Network Tester

Gianni Antichi, Muhammad Shahbaz, Yilong Geng, Noa Zilberman, Adam Covington, Marc Bruyere, Nick McKeown, Nick Feamster, Bob Felderman, Michaela Blott, Andrew W. Moore, and Philippe Owezarski

Abstract

Despite network monitoring and testing being critical for computer networks, current solutions are both extremely expensive and inflexible. Into this lacuna we launch the Open Source Network Tester, a fully open source traffic generator and capture system. Our prototype implementation on the NetFPGA-10G supports 4 × 10 Gb/s traffic generation across all packet sizes, and traffic capture is supported up to 2 × 10Gb/s with naïve host software. Our system implementation provides methods for scaling and coordinating multiple generator/capture systems, and supports 6.25 ns timestamp resolution with clock drift and phase coordination maintained by GPS input. Additionally, our approach has demonstrated lower-cost than comparable commercial systems while achieving comparable levels of precision and accuracy; all within an open-source framework extensible with new features to support new applications, while permitting validation and review of the implementation.

Computer networks are the hallmark of 21st century society and underpin virtually all infrastructure in the modern world. Consequently, society relies on the correct operation of these networks. To achieve compliant and functional equipment, effort is put into all parts of the network equipment life cycle. Testing validates new designs, equipment is tested throughout the production process, and new deployments are rigorously tested for compliance and correctness. In addition, many owners of network equipment employ a relentless battery of testing and measurement to ensure that the infrastructure operates correctly.

The continuous innovation that is such a desirable property of the Internet has also led to a dilemma for network testing. For a typical piece of new networking equipment there are a multitude of related IEEE standards and standards-track Internet Engineering Task Force (IETF) RFCs, each requiring test cases to ensure correctness for network equipment. This has led to a multi-billion-dollar industry in network test equipment, giving rise to companies such as Ixia, Spirent, Fluke, and Emulex/Endace, among others.

However, such equipment has evolved with a number of undesirable characteristics: commonly closed and proprietary systems with limited flexibility well outside the reach of most universities and research laboratories. Even a modest two-port 10GbE network tester capable of full line-rate costs upward of $25,000, and adding support for additional protocols, large numbers of TCP streams, and nontrivial traffic profiles quickly increases this price.

This has been the case for two reasons. First, network test equipment capable of full-line rate with high-precision timestamping is a significant engineering challenge, leading to state-of-the-art and specialist physical components. Second, test equipment is often developed simultaneously with early prototype network equipment. Thus, modest numbers of units sold mean an expensive and slow time to develop test hardware and software.

This slow development cycle and high expense opens an opportunity for an open source network tester. It is no longer necessary to build network testers on top of specialized, proprietary hardware. There are multiple open source hardware platforms with the potential for line-rate across many 10GbE ports, for example, the NetFPGA-10G,[1] Xilinx VC709,[2] and Terasic DE5-Net.[3] Each of these fully reprogrammable cards purports to be capable of running at line-rate. For example, the NetFPGA-10G has 4 10GbE interfaces, is based on a Xil-

Gianni Antichi, Noa Zilberman, and Andrew W. Moore are with the University of Cambridge.

Yilong Geng, Adam Covington, and Nick McKeown are with Stanford University.

Muhammad Shahbaz and Nick Feamster are with Georgia Tech.

Marc Bruyere is with DellForce10 and Université de Toulouse, CNRS and LAAS.

Bob Felderman is with Google.

Michaela Blott is with Xilinx.

Philippe Owezarski is with Université de Toulouse, CNRS and LAAS.

[1] http://www.netfpga.org.

[2] http://www.xilinx.com/products/boards-and-kits/EK-V7-VC709-CES-G.htm.

[3] http://www.de5-net.terasic.com.

[4] http://www.osnt.org.

inx field programmable gate array (FPGA), and is available to the research and teaching community for less than $2000, including firmware and software.

We therefore present the Open Source Network Tester (OSNT),[4] primarily for the research and teaching community. Such a tester needs to be able to achieve full line-rate, provide sufficiently accurate timestamping, and be flexible enough to allow new protocol tests to be added to the system.

We believe that, as an open source community grows, a low-cost open source network tester will also prove valuable to the networking industry. We also envisage the enabling of new testing and validation deployments that are simply financially impractical using commercial testers. Such deployments may see the use of hundreds or thousands of testers, offering previously unobtainable insights and understanding.

In this article we present an architecture for OSNT, describe our first prototype based on the NetFPGA open source hardware platform, and present early-day benchmarks illustrating the tester in operation. OSNT is portable across a number of hardware platforms, maximizing reuse and minimizing reimplementation costs as new hardware, physical interfaces, and networks become available. By providing an open source solution we invite everyone from the community to audit (and improve) our implementation as well as adapt it to their needs.

Related Work

Network testers, and open source networks, are not new; uniquely, OSNT brings the incorporation of designs that operate intimately with the hardware. Our efforts ride the established tradition of network measurement and testing that exists in the network research and academic communities.

A small sample of open source and community projects include Iperf [1] and later Netperf [2], developed to provide performance tests of throughput and end-to-end latency. Traffic loads from previously captured *pcap* files could be transmitted using Tcpreplay [3]. Netalyzer [4] uses bespoke server and client infrastructure to measure many aspects of Internet performance and behavior. Swing [5] provided a closed-loop traffic generator: first monitoring and characterizing, and then regenerating system load replicating the measured characteristics. Early attempts at both flexible and feature-rich traffic generation led to the Ostinato [6] traffic generator. The netmap [7] achieves near-optimal host throughput, but is still restricted by the underlying hardware for timestamps, traffic shaping, and maximum rate capacity. As a final example, Bonelli *et al.* [8] describe near-line-rate traffic on a 10 Gb/s link that uses multi-core multi-queue commodity hardware, albeit without the flexibility or guarantee of full line-rate throughput, precise traffic replay timing, and sufficient packet capture timestamp accuracy and precision.

Commercial network testers are provided by a number of companies: Ixia and Spirent dominate, but other test equipment manufacturers also have network test offerings. Despite their ability to perform at high line-rate, a criticism common to all these systems regards cost and inflexibility. Supporting newly designed protocols is often expensive, while supporting a newly designed physical line standard can result in an entirely new system.

In the measurement community the ubiquitous *pcap* program, *tcpdump*, has been the tool of choice for network capture. However, capture system performance (and rates of loss) are dictated by the underlying host: a combination of hardware, operating system, device drivers, and software. Additionally, it is rare for these software systems to provide any common clock across the captures, making end-to-end latency measurements complicated and inaccurate. There have been software/hardware efforts in the past to incorporate GPS-coordinated high-precision hardware timestamps and use device-driver designs intended to mitigate loss under load [9]. However, this work was limited to 1GbE and serves now only to provide a motivating example. NTP is a mature time synchronization method; however, it can only achieve an accuracy better than 1 ms under limited conditions [10], making it unsuitable for high-precision traffic characterization.

In contrast to the large range of commercial offerings available to generate traffic, the high-precision capture market has few commercial systems and is dominated by the Endace DAG card.

Several previous NetFPGA-based projects using the previous generation NetFPGA 4 × 1GbE platform have also provided traffic generation [11] and traffic monitoring [12]. The architecture of OSNT has been heavily informed by the designs, limitations, and experience with these systems.

The OSNT Architecture

The OSNT architecture is motivated by limitations in past work: closed source/proprietary solutions, high costs, lack of flexibility, and the omission of important features such as timestamping and precise packet transmission.

Alongside flexibility there is a need for scalability; while our prototype work has focused on single-card solutions, our desire to reproduce real operating conditions means we must have a system that can test beyond single network elements. A production network needs to be tested as close as possible to its real operating conditions; this means the OSNT system must also be able to recreate such real operating conditions.

From the outset it has been obvious that flexibility must be a key part of the OSNT approach. This flexibility is needed to accommodate the variety of different uses for OSNT. Four distinct modes of use have become clear.

OSNT traffic generator: a single card, capable of generating and receiving packets on four 10GbE interfaces. By incorporating timestamps into each outbound packet, information on end-to-end delay and loss can be computed. Such a system can be used to test a single networking element (e.g., switch or router) or a network encompassed within a sufficiently small area that different inputs and outputs from the network can be connected to the same card.

OSNT traffic monitor: a single card, capable of capturing packets arriving through four 10GbE ports and transferring them to the host software for analysis and further processing. Alongside a range of techniques utilized to reduce the bottleneck of PCIe bandwidth (packet batching, ring receivers, and pre-allocated host system memory), packets are optionally hashed and truncated in hardware. The card is intended to provide a loss-limited capture system with both high-resolution and high-precision timestamping of events in a live network.

Hybrid OSNT system: Our architecture allows the combination of the traffic generator and traffic monitor into a single FPGA device and a single card. Using high-precision timestamping of departing and arriving packets, we can perform full line-rate per-flow characterization of a network (device) under test.

Scalable OSNT system: our approach to coordinating large numbers of multiple traffic generators and traffic monitors synchronized by a common time base to provide the resources and port -count to test larger network systems. While still largely untested, such a coordinated system has been a design objective from the outset.

The OSNT architecture is designed to support these needs for network testing using a scalable architecture that can utilize multiple OSNT cards. Using one or more synchronized OSNT cards, our architecture enables a user to perform measurements throughout the network, characterizing aspects such as end-to-end latency and jitter, packet loss, congestion events, and more.

It is clear that our approach must be capable of full line-rate operation. To this end we built our prototype on the NetFPGA-10G platform, an open source hardware platform designed to be capable of full line-rate. We describe our prototype implementation later.

While there is a clear need for one or both of the traffic capture and traffic generator cores in our OSNT system to be present in each use case, these two subsystems have orthogonal design goals: the capture system is intended to provide high-precision inbound timestamping with a loss-limited path that gets (a subset of) captured packets into the host for further processing, whereas the traffic generator requires precision transmission of packets according to a generator function that may include closed-loop control (e.g., TCP) and even (partial) application protocol implementation.

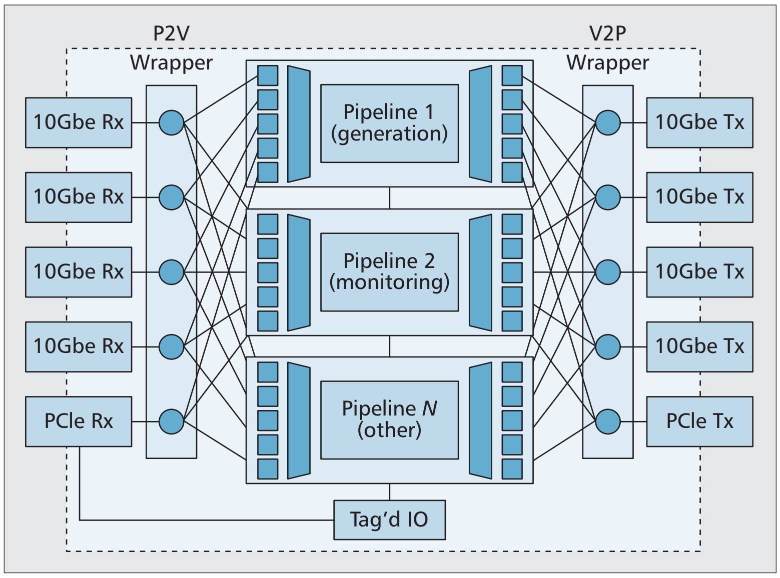

Figure 1. NetV — an approach for NetFPGA virtualization.

Given we already had a proven starting design for both generator and capture engines [11, 12], along with a keen desire to employ component reuse, we were led to develop the *NetV* approach that virtualizes the underlying hardware platform.[5] The approach, shown in Fig. 1, extends a hardware xlatform such as the NetFPGA, using *P2V*: physical to virtual and *V2P*: virtual to Pphysical wrappers. The V2P hardware wrapper is a per-port arbiter that shares access among each of the 10GbE and PCIe interface pipelines. This permits multiple NetFPGA pipelines within a single FPGA fabric on a single board, in turn providing support for seamless integration of existing pipelines with strong isolation characteristics. For example, a traffic generator can coexist with a high-precision capture engine. Each pipeline is tagged with a unique ID to ensure that register accesses can be distinguished among different pipelines. In this manner, traffic generation and monitoring can be implemented as either standalone units or a combined system on a single card. Using multiple pipelines in the same design does not affect the overall performance as long as they do not share data structures. The only limitation is given by the available FPGA resources.

Our design has focused on one particular architectural approach; this direction was selected to maximize code reuse at the expense of potential redundant gate logic. Other OSNT architectures may be appropriate but are not explored here for sake of brevity.

Traffic Generation

The OSNT traffic generator both generates packets and analyzes return statistics. It is designed to generate full line-rate per card interface, and is scalable in a manner that allows for multiple traffic generators to work in parallel within a single OSNT environment. Traffic generation features include:

- Support for a large number of different traffic flows
- Flexible packet header specification over multiple headers
- Support of several standard protocols
- Sufficient flexibility to test future protocols
- Simulation of multiple networking devices/end systems (e.g., routers running BGP)
- Allowing timestamping of in- and out-bound packets
- Allowing per-packet traffic-shaping
- Statistics gathered per-flow or flow-aggregate
- Support for negative testing through malformed packets

In addition to the above features, OSNT can be customized to support different protocols, numbers of flows, and many other features in each given application context.

Figure 2 illustrates the high-level architecture of the traffic generation pipeline. The center of the pipeline is a set of micro-engines, each used to support one or more protocols at network and transport layers such as Ethernet, TCP, and UDP, and application protocols such as BGP. Each micro-engine either generates synthetic or replays captured traffic for one or more of the selected egress interfaces. A basic micro-engine is a simple packet replay: a set of predefined packets are sent out a given number of times as configured by the software. Each micro-engine contains three building blocks: traffic model (TM), flow table (FT), and data pattern (DP). The TM contains information about the network characteristics of the generated traffic, such as packet size and inter-packet delay (IPD). It is a compiled list of these characteristics, extracted by the host software and installed into the hardware. Each parameter is software defined, permitting arbitrary rate distribution patterns such as constant bit rate (CBR) or Poisson distribution. The FT contains a list of header template values used by the micro-engine when generating a packet. Each packet - header is defined by the FT. In this manner, multiple flows with different header characteristics can be generated by a single micro-engine. The micro-engine takes each header field and manipulates it in one of several ways before setting it: a field may remain constant, incrementally increase, interleave, or be set randomly or algorithmically. The number of flows supported by the FT depends on the trade-off between trace complexity and the number of fields to be manipulated. The DP module sets the payload of a generated packet. The payload can be set to a random or prespecified pattern. A prespecified pattern allows a user to set the payload of packets to a unique pattern so that the user can execute specific network tests such as continuous-jitter measurement. It also provides in-payload timestamping of departing packets and capabilities for debugging/validating received packets.

Packets generated by the micro-engine are sent to a per-port arbiter. The arbiter selects among all the packets des-

[5] Our reference prototype is the NetFPGA, but we believe that the architecture including approaches such as NetV will be generic across a range of hardware platforms.

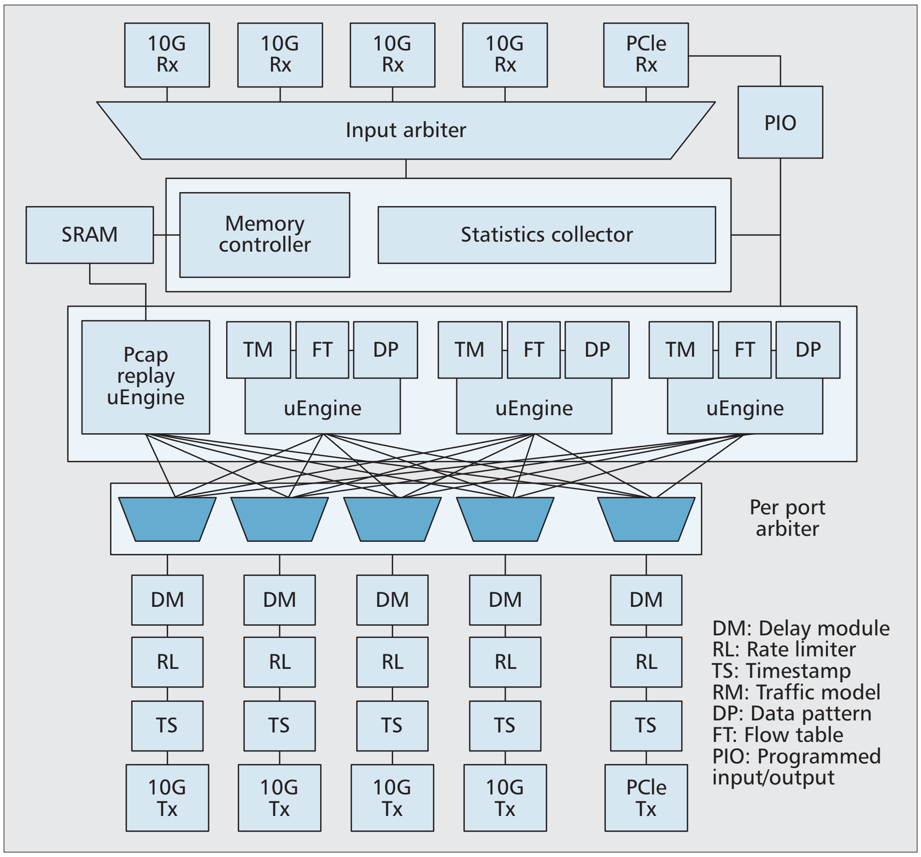

Figure 2. The architecture of the OSNT traffic generation system.

tined for a port from each micro-engine. Ordering is based on the required packet departure time. A delay module (DM) located after the arbiter will delay packets by each flow's IPD. A rate limiter (RL) guarantees that no flow exceeds the rate assigned to it at each port. Lastly, the packet goes to the (10GbE) medium access control (MAC), from which it is transmitted to its destination.

The traffic generator implementation can also receive incoming packets and provide statistics on them at either the port or flow level. This allows use of the traffic generation subsystem as a standalone unit without an additional external capture subsystem. To this end, packets entering the card through a physical interface are measured, the statistics gathered, and the received packets discarded. The gathered statistics are relayed to host software using the programmed input/output (PIO) interface.

The traffic generator has an accurate timestamping mechanism located just before the transmit 10GbE MAC. The mechanism, identical to the one used in the traffic monitoring unit and described next, is used for timing-related measurements of the network, permitting characterization of measurements such as latency and jitter. The timestamp is embedded within the packet at a preconfigured location and can be extracted at the receiver as required.

As for the software side, we provide an extensible graphical user interface (GUI) to interact with the hardware (load a PCAP trace to replay in hardware, define the per-packet inter-departure time, etc.).

Traffic Monitoring

The OSNT traffic monitor provides four functions:
- Packet capture at full line-rate
- Packet filtering, permitting selection of traffic of interest
- High-precision, accurate packet timestamping
- Statistics gathering

Figure 3 illustrates the architecture of the monitoring pipeline that provides the functionality enumerated above. The 5-tuple (protocol, IP address pair, and layer four port pair) extraction is performed using an extensible packet parser able to recognize both virtual LAN (VLAN) and multiprotocol label switching (MPLS) headers along with IP-in-IP encapsulation. Further flexibility is enabled by extending the parser implementation code as required.

A module positioned immediately after the physical interfaces and before the receive queues timestamps incoming packets as they are received by hardware. Our design is an architecture that implicitly copes with a workload of full line-rate per port of minimum sized packets. However, this will often exceed the capacity of the host (processing, storage, etc.), or may contain traffic of no practical interest. To this end we implement two traffic-thinning approaches. The first of these is to utilize the 5-tuple filter implemented in the "core monitoring" module. Only packets that are matched to a rule are sent to the software, while all other packets are dropped. The second mechanism is to record a fixed-length part of each packet (sometimes called a *snap-length*) along with a hash of the entire original packet. The challenge here is that if a user is interested in all packets on all interfaces, it is possible to exhaust the host resources. We quantify the PCIe bandwidth and the trade-off for snap-length selection mentioned later.

As for the software side, we provide a Python-based GUI that allows the user to interact with the hardware components (e.g., enable cut/hash, set filtering rules, check statistics). A C-based application that comes with it records the received traffic in both PCAP or PCAPNG format. This allows offline use of common libpcap-based tools (e.g., TCPDump, Wireshark.) These tools do not work directly with OSNT: the device driver secures performance by bypassing the Linux TCP/IP stack. We refer the reader to the OSNT website for further information about the software application programming interface (API).

Timestamping

Providing an accurate timestamp to (incoming) packets is a critical objective of the traffic monitoring unit. Packets are timestamped as close to the physical Ethernet device as possible in order to minimize first-in first-out (FIFO)-generated jitter and permit accurate latency measurement. A dedicated timestamping unit stamps packets as they arrive from the physical (MAC) interfaces. Each packet is appended with a 64-bit timestamp.

Motivated by the need to have minimal overhead while also providing sufficient resolution and long-term stability, we have chosen to use a 64-bit timestamp divided into two parts: the upper 32 bits count seconds, while the lower 32 bits provide a fraction of a second with a maximum resolution of approximately 233 ps; the practical prototype resolution is 6.25 ns.

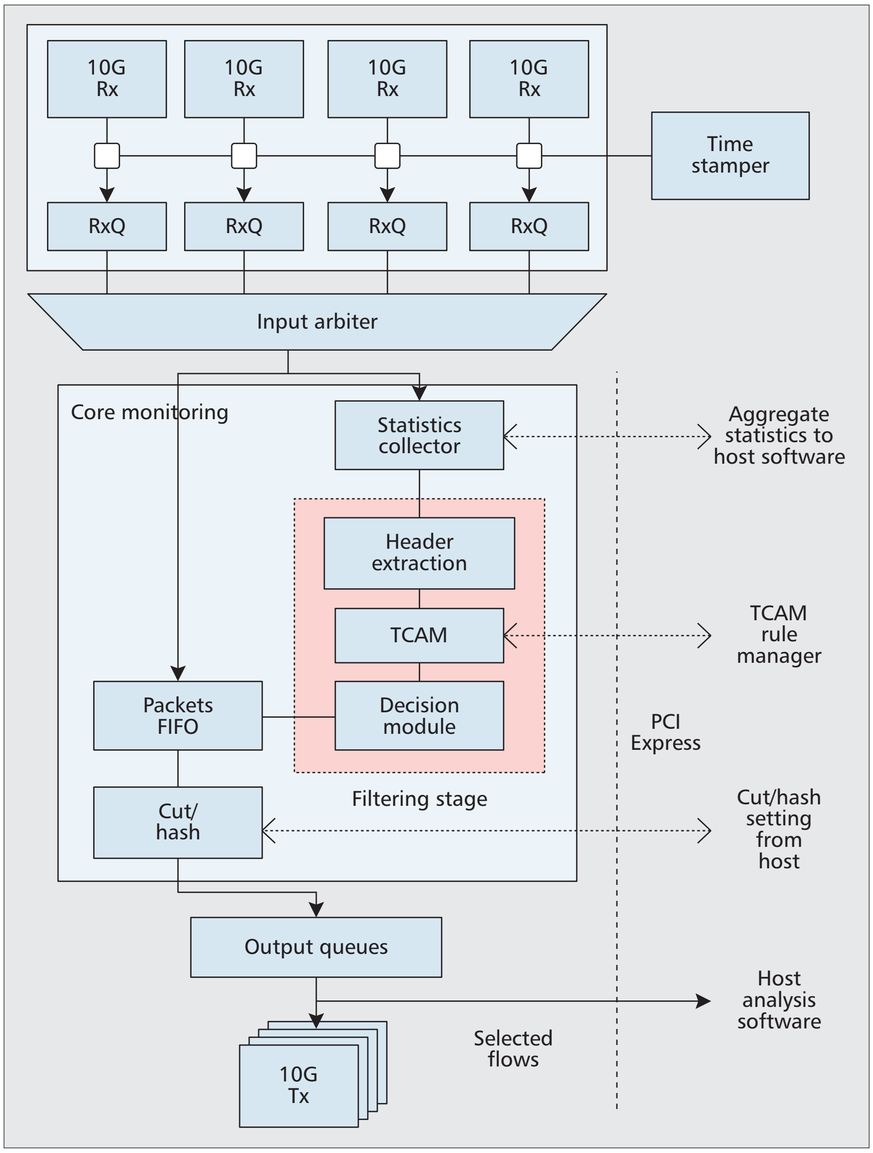

Figure 3. The architecture for OSNT traffic monitoring system.

Integral to accurate timekeeping is the need to correct the frequency drift of an oscillator. To this end, we use direct digital synthesis (DDS), a technique by which arbitrary variable frequencies can be generated using synchronous digital logic [13]. The addition of a stable pulse-per-second (PPS) signal such as that derived from a GPS receiver permits both high long-term accuracy and the synchronization of multiple OSNT elements. The selection of a timestamp with this precision was a conscious effort on our part to ensure that the abilities of the OSNT design are at least as good as the currently available commercial offerings.

OSNT NetFPGA-10G Prototype

Our prototype implementation of the OSNT platform has been on the NetFPGA-10G open source hardware platform. The NetFPGA system provides an ideal rapid prototyping target for the work of OSNT. Since its original inception as an open source high-speed networking platform for the research and education community [14] and through its second generation [15], the NetFPGA has proven to be an easy-to-use platform. The NetFPGA project supplies users with both basic infrastructure and a number of pre-worked open source designs intended to dramatically simplify a user's design experience.

The NetFPGA-10G card, as shown in Fig. 4, is a 4-port 10GbE PCIe adapter card incorporating a large FPGA fabric. At the core of the board is a Xilinx Virtex-5 FPGA: XC5VTX240T-2 device. Additionally, there are five peripheral subsystems that complement the FPGA: four 10 Gb/s SFP+ Ethernet interfaces, a Gen1 PCIe subsystem providing the host-bus adapter interface, and memory consisting of a combination of both SRAM and DRAM devices. The memories were selected to provide minimal latency and maximal bandwidth over the available FPGA I/Os. The fourth and fifth subsystems are expansion interfaces and the configuration subsystem. The board is implemented as a three-quarter-length PCIe adapter, but can also operate as a standalone unit outside the server environment.

Experiences with Our Prototype

By building our prototype on the NetFPGA-10G platform, we have inherited several platform constraints. Despite having a large FPGA device, design decisions must trade resources. One example of this is in the sizing of TCAM tables for filtering. Table size is traded directly against overall design size. In our prototype implementation, the tuple-based filtering table is limited to 16 entries.

While the internal NetFPGA datapath has been designed to accommodate full line-rate, minimum-sized packets, the PCIe interface lacks the bandwidth to transmit all traffic to or from the host. The NetFPGA-10G provides a first-generation 8-lane PCIe implementation. This interface uses an maximum transmission unit (MTU) of 128 bytes, and without careful packing a naïve implementation of DMA and device driver may achieve as low as 33.5 percent utilization (for transactions of 129-byte packets). Furthermore, even for an ideal scenario, this interface imposes a limit of around 13.1 Mpkts/s for an MTU of 128 bytes or a little over 15 Gb/s. It is clear that capture-to-host of all four interfaces when operating at 10 Gb/s into the host is not practical. Alongside flow filtering, the traffic-thinning technique of selecting a snap-length places a known limit on the maximum amount of data that needs to be transferred over the PCIe to the host.

The option to add a hash of the original packet, along with a fixed snap-length, means that we can reduce the potential number of bytes per packet to a known upper bound. Although the hash adds an overhead of 128 b/pkt, it permits practical packet identification, which in turn means we can perform end-to-end latency measurements as well as identifying specific loss events. The ability to do bandwidth limiting in this way allows us to achieve a maximum rate of approximately 21.7 Mpkts/s provided we use non-naïve DMA and device driver mechanisms.

Fortunately, there has been considerable progress in non-

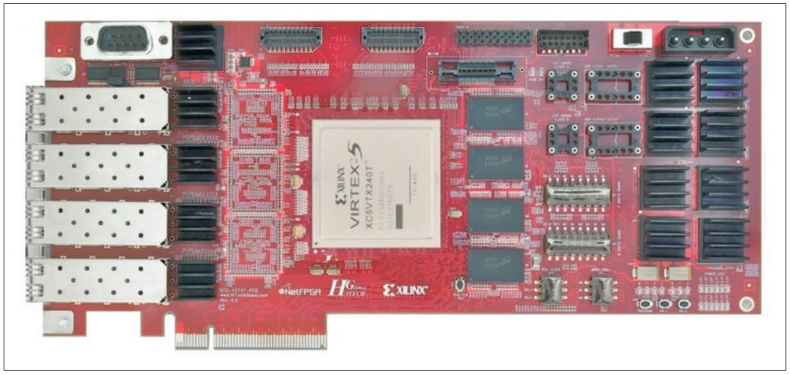
Figure 4. The NetFPGA-10G board.

naïve DMA and device-driver mechanisms to reduce the bottleneck of PCIe bandwidth; packet batching, ring receivers, and pre-allocated host system memory have all seen use in past dedicated capture systems [9]. Recent efforts such as netmap achieve rates of 14.8 Mpkt/s into user space for single-port commodity 10GbE interface cards. Our architecture is not limited to a current hardware implementation; the OSNT system, when running on more advanced hardware such as the Xilinx VC709, using the third generation PCIe, has sufficient bandwidth to support full size payloads for all four 10GbE ports. In fact, the open source nature of OSNT means that having this system operate effectively on any future NetFPGA platform, or other platforms from Xilinx or indeed from other FPGA vendors, is no more complicated than the porting of any open source project.

Figure 5 shows the capture engine performance results. The system has been validated for one and two ports against 100 percent line utilization (packets sent back to back) across a range of packet sizes. In the first case, OSNT is able to record all received traffic, without loss, independent of packet length. Additionally, using two ports at the same time, the system is able to record traffic without experiencing any kind of loss up to 14 Gb/s (PCIe Gen1 limitation); the impact of the cut/hash feature at reducing traffic across the PCIe is clear.

We validated the OSNT performance against the IXIA 400T and similtaneously confirmed these results via a parallel capture using optical-port splitters to an Emulex Endace DAG 9.2, each equipped with 2x10G ports. IXIA provides the capability of both generating full line-rate traffic and full line-rate monitoring, permitting validation of both capture and generation capabilities. The Endace DAG provides full line-rate capture and high-precision, and offers a further confirmation mechanism.

Testing of the traffic generator confirmed to our satisfaction that the OSNT traffic generator is able to generate full line-rate over two ports independent of the packet length. Tests were conducted over a range of packet sizes with results compared directly against IXIA-based generators. In all experiments data was generated (and measured) on all four NetFPGA ports with a combination of IXIA and Endace packet capture and measurement.

Conclusions

In this article we have introduced OSNT, an open source network tester. We have described the OSNT architecture, which permits a flexible combination of multiple packet processing pipelines using a new virtualization technique, NetV. While the NetV virtualization approach was designed with the NetFPGA in mind, this technique is not bound to that hardware and should be able to provide flexibility and versatility across a range of uses. Using the NetV approach, we have shown how the OSNT system can implement both traffic generator and network monitor functions. We have also described our prototype implementation using the rapid-prototyping NetFPGA platform and characterized aspects of that implementation.

The OSNT platform provides a network tester that is able to combine desirable software flexibility with the advantages of being built on an open source hardware platform. The versatility of OSNT is in its suitability for a range of applications, from the testing of single items of networking equipment to the characterizing of large distributed networks.

The OSNT system is available to the research community through the NetFPGA project. Any user who owns a NetFPGA card can simply use it, with no additional hardware expense. We envisage the project being extended and enhanced by the research community, and users are encouraged to contribute further features and capabilities, as well as to share their own experience using OSNT.

The promise of OSNT is exciting. In the field of network measurement alone, high-precision loss-limited capture has led to remarkable progress in the characterization and understanding of the modern Internet. The OSNT traffic monitor overcomes the two biggest s with these capture deployments to date — the cost and lack of flexibility — while also, by virtue of being open source, providing an auditable test system that encourages repeatability in network science.

Acknowledgments

We thank the NetFPGA community, David Fermor, Scott Whyte, and Richard Hay for inspiring and assisting with this work. The language and form of this article has been improved immeasurably by feedback from Jon Crowcroft and the anonymous reviewers. This project is jointly supported by the NSF CRI program under contract 0855268 and by the EPSRC INTERNET Project EP/H040536/1.

References

[1] iperf, TCP and UDP Bandwidth Performance Measurement Tool, http://code.google.com/p/iperf.
[2] Netperf, http://www.netperf.org.

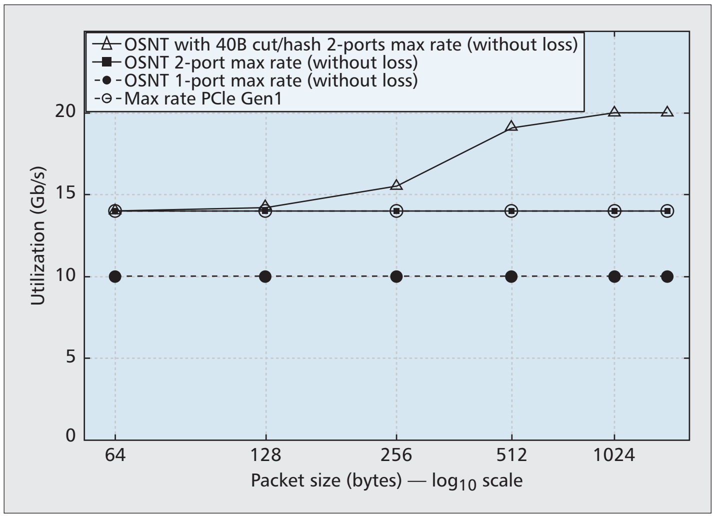

Figure 5. The OSNT per-packet capture engine performance for various presented traffic loads.

[3] Tcpreplay, https://github.com/synfinatic/tcpreplay.
[4] C. Kreibich *et al.*, "Netalyzr: Illuminating The Edge Network," *ACM Internet Measurement Conf.*, 2010.
[5] K. V. Vishwanath and A. Vahdat, "Swing: Realistic and Responsive Network Traffic Generation," *IEEE/ACM Trans. Net.*, vol. 17, no. 3, 2009.
[6] P. Srivats, "OSTINATO: An Open, Scalable Packet/Traffic Generator," *FOSS.IN*, 2010.
[7] L. Rizzo, "Netmap: A Novel Framework for Fast Packet I/O," *USENIX Annual Technical Conf.*, 2012.
[8] N. Bonelli *et al.*, "Flexible High Performance Traffic Generation on Commodity Multi-Core Platforms," *Traffic Monitoring and Analysis*, Springer, 2012.
[9] A. Moore *et al.*, "Architecture of a Network Monitor," *Passive & Active Measurement Wksp.*, 2003.
[10] M. Ussoli and G. Prytz, "Sntp Time Synchronization Accuray Measurements," *IEEE Int'l. Conf. Emerging Technologies and Factory Automation*, 2013.
[11] A. Covington *et al.*, "A Packet Generator on the NetFPGA Platform," *IEEE Symp. Field Programmable Custom Computing Machines*, 2009.
[12] G. Antichi *et al.*, "Enabling Opensource High Speed Network Monitoring on NetFPGA," *IEEE/IFIP Network Operations and Management Symp.*, 2012.
[13] P. Saul, "Direct Digital Synthesis," *Circuits and Systems Tutorials*, 1996.
[14] J. W. Lockwood *et al.*, "Netfpga — An Open Platform for Gigabit-Rate Network Switching and Routing," *IEEE Int'l. Conf. Microelectronic Sys. Education*, 2007.
[15] M. Blott *et al.*, "FPGA Research Design Platform Fuels Network Advances," *Xilinx Xcell J.*, no. 73, 2010.

Biographies

GIANNI ANTICHI received B.E. and M.E. degrees in telecommunications engineering and a Ph.D. degree in information engineering from the University of Pisa, Italy, in 2005, 2007, and 2011, respectively. He is currently a research associate at the Computer Lab of the University of Cambridge. His research interests are in the area of hardware accelerated networking systems, network design, network monitoring, packet classification, and software defined networks.

MUHAMMAD SHAHBAZ is a Ph.D. student in the Department of Computer Science at the Georgia Institute of Technology. His research focuses on the application of software-defined networking in large-scale IP networks, SDN performance optimization, network testing, and programmable hardware. Previously, he worked as a research assistant at the University of Cambridge on the CTSRD and MRC2 projects, and is currently a core member of the NetFPGA-10G project initiated by Stanford University.

YILONG GENG is a Ph.D. student in the Electrical Engineering Department of Stanford University. During his first year at Stanford he worked with Prof. Nick Mckeown on the NetFPGA-10G project and the Open Source Network Tester project. After that he joined Prof. Balaji Prabhakar's group and started working on the Societal Networks projects, which try to influence the behavior of users in practical networks (e.g., traffic networks) by applying incentives.

NOA ZILBERMAN received her B.Sc., M.Sc. (both magna cum laude), and Ph.D. degrees in electrical engineering from Tel-Aviv University, Israel. Since 1999 she has filled several development, architecture, and managerial roles in the telecommunications and semiconductor industries. She is currently a research associate in the Systems Research Group, Computer Laboratory, University of Cambridge.

ADAM COVINGTON is a research associate in Nick McKeown's group at Stanford University. He has been working on the NetFPGA project since 2007. He has been helping run the NetFPGA project, both 1G and 10G, since 2009. His current research interests include reconfigurable systems, open source hardware and software, artificial intelligence, and dynamic visualizations of large-scale data. Previously, he was a research associate with the Reconfigurable Network Group (RNG) at Washington University in St. Louis, Missouri.

MARC BRUYERE is a technical consultant at DellForce10 and a Ph.D. student at the CNRS, LAAS at Toulouse. He started his carrier in 1996 working for Club-Internet.fr, and for Cisco, Vivendi Universal, Credit Suisse First Boston, Airbus/DimensionData, Force10 Networks, and Dell. He is a Cisco Certified Internetwork Expert #16651. He has been involved in the NetFPGA project for a few years, and his thesis is about measurement in an IXP OpenFlow/SDN environment.

NICK FEAMSTER is an associate professor in the College of Computing at Georgia Tech. He received his Ph.D. in computer science from MIT in 2005, and his S.B. and M.Eng. degrees in electrical engineering and computer science from MIT in 2000 and 2001, respectively. His research focuses on many aspects of computer networking and networked systems, with a focus on network operations, network security, and censorship-resistant communication systems.

NICK MCKEOWN [F] is a professor of electrical engineering and computer science at Stanford University. In recent years most of his time has been devoted to SDN, including research, transfer of ideas to industry, and evangelism. He co-founded the Open Networking Foundation (ONF), the Open Networking Lab (ON.Lab), and Nicira and, more recently, Barefoot. He is a member of the U.S. National Academy of Engineering (NAE), the U.K. Royal Academy of Engineering, and a Fellow of the ACM.

BOB FELDERMAN spent time at both Princeton and the University of California at Los Angeles (UCLA) before venturing out into the real world. After a short stint at the Information Sciences Institute he helped to found Myricom, an early leader in cluster computing networking technology. Later he applied high-performance computing ideas to the Ethernet/IP space while working at Packet Design and Precision I/O. That experience led him to Google, where he has spent the past eight years working on issues in data center networking and general platforms system architecture.

MICHAELA BLOTT graduated from the University of Kaiserslautern in Germany. She worked in both research institutions (ETH and Bell Labs) as well as development organizations, and was deeply involved in large-scale international collaborations such as NetFPGA-10G. Today, she works as a senior research scientist at the Xilinx labs in Dublin, Ireland, heading a team of international researchers with a focus on FPGAs in data centers, high-speed networking, and emerging memory technologies.

ANDREW W. MOORE is a senior lecturer with the Computer Laboratory, University of Cambridge, where he is part of the Systems Research Group on issues of network and computer architecture. His current research interests include open network research and education using the NetFPGA platform, low-power energy-aware networking, and novel network and systems data center architectures.

PHILIPPE OWEZARSKI is director of research at CNRS (the French center for scientific research), working at LAAS (the Laboratory for Analysis and Architecture of Systems), Toulouse, France. He got a Ph.D. in computer science in 1996 from Paul Sabatier University, Toulouse III. His main interests deal with high-speed networking and, more specifically, IP network monitoring, and quality of service enforcement and security based on measurements.