Architecture for an Open Source Network Tester

Muhammad Shahbaz, Gianni Antichi, Yilong Geng, Noa Zilberman, Adam Covington, Marc Bruyère, Nick Feamster, Nick Mckeown, Bob Felderman, Michaela Blott, et al.

Architecture for an Open Source Network Tester

Muhammad Shahbaz[†], Gianni Antichi[*], Yilong Geng[‡], Noa Zilberman[*]
Adam Covington[‡], Marc Bruyere[‖], Nick Feamster[†], Nick McKeown[‡]
Bob Felderman[§], Michaela Blott[¶], Andrew W. Moore[*], Philippe Owezarski[‖]
[†]Georgia Tech [*]University of Cambridge [‡]Stanford University [¶]Xilinx [§]Google [‖]Université de Toulouse

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management, Network monitoring*

Keywords

Open Source; Programmable Hardware; High Speed; NetFPGA; Monitoring; Traffic-Generation; Packet-Capture; Packet-Sniffing

1. INTRODUCTION

To make networks more reliable, enormous resources are poured into all phases of the network-equipment lifecycle. The process starts early in the design phase when simulation is used to verify the correctness of a design, and continues through manufacturing and perhaps months of rigorously trials. With over 7,000 Internet RFCs and hundreds of IEEE standards, a typical piece of networking equipment undergoes hundreds of conformance tests before being deployed. Finally, when deployed in a production network, the equipment is tested regularly. Throughout the process, a relentless battery of tests and measurement help ensure the correct operation of the equipment.

Not surprisingly, to support the testing effort, there is a multi-billion dollar industry building and selling network test equipment for all stages of design, development and deployment. It is common for a large network equipment vendor to spend tens of millions of dollars per year on test equipment (from companies such as Ixia, Spirent, Fluke, and Emulex/Endace) to test physical layers, protocols and applications. Researchers and educators would also like to use test equipment to understand current networks and when prototyping new ideas. Unfortunately, commercial test equipment is closed, proprietary - making it difficult to try out new ideas - and high prices place it well beyond reach of most university teaching and research laboratories.[1]

We believe that it is no longer necessary to build network testers upon specialized, proprietary hardware; it is now possible to develop open-source network testers that run at line-rate, particularly at 10Gb/s. The NetFPGA-10G, Xilinx V5TXT and Terasic DE5-Net, are all programmable, open-source hardware platforms that can be programmed to test networks at line-rate. For example, the NetFPGA-10G (developed by the authors) has 4 × 10GbE interfaces, is based on an FPGA, and is available to the research and teaching community for less than $2,000.

We therefore set out to create an open-source network tester (OSNT), primarily for the research and teaching community. Although, we believe that as an open-source community grows, a low-cost open-source network tester would be as valuable to the networking industry as a whole. A low-cost tester could also enable large-scale testing with tens to thousands of testers, that was financially infeasible before.

In this paper we propose an architecture for an extensible open-source network tester, we describe a first prototype running on the NetFPGA open-source hardware platform and present early-day benchmarks indicating the tester in operation.

2. PROPOSED ARCHITECTURE

Key design goals for the OSNT architecture were low cost, flexibility, high-precision time-stamping and packet transmission, as well as scalability. In order to satisfy these goals, we made a series of choices which are described in the following paragraphs.

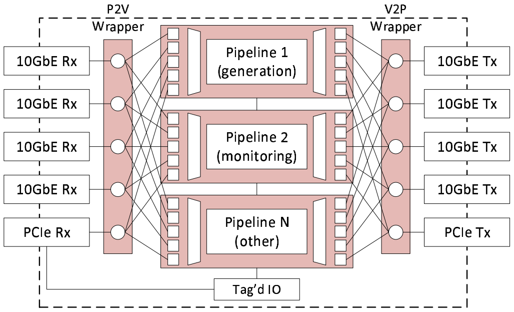

Figure 1: NetV — an approach for NetFPGA Virtualization.

At a system-level, we provide the means to join and synchronize numerous OSNT cards. This lets a user to perform measurements throughout the network, such as end-to-end latency and jitter, packet-loss, and congestion control. While we clearly need to generate traffic and capture traffic, we don't always need precisely one of each. We therefore allow the user to pick a mix of both, using an approach we call *NetV*, illustrated in Figure 1. Two wrappers let us run multiple NetFPGA pipelines inside: the *V2P* (Virtual to Physical) wrapper is a per-port arbiter that shares access among each of the 10GbE and PCIe interface-pipelines; the *P2V* (Physical to Virtual) wrapper copies every incoming packet to the ingress virtual interface of all the pipelines.

[1] Even a modest two port 10GbE network tester capable of full line-rate costs upward of $25,000.

978-1-4799-1640-5/13/$31.00 ©2013 IEEE

This approach lets us create new pipelines, while keeping them isolated from each other.

2.1 Traffic Generation

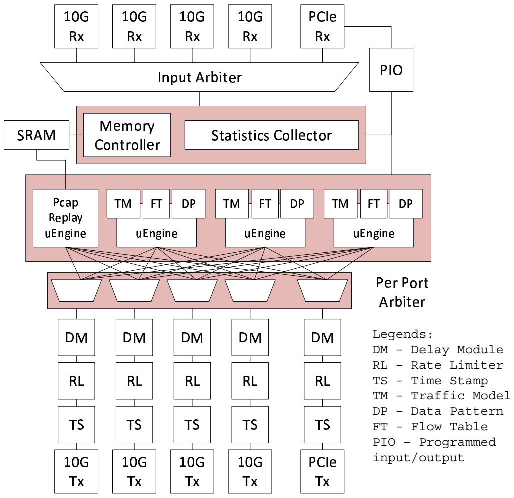

Figure 2: The architecture for OSNT traffic generation system.

The OSNT traffic generator generates packets and then analyses their statistics upon return, as illustrated in Figure 2. It consists of several micro-engines, each of which generates traffic according to a given traffic model, list of flow values and data patterns from which we generate packet size, rate and inter-packet delays. The traffic generator consists of five functional units: The Arbiter selects packets and forwards them at their departure time. The Delay Module (DM) and Rate Limiter (RL) control delay and rate for each flow, before passing packets to the Timestamping (TS) module. The timestamp is appended to the packet and is used to measure latency and jitter. Finally, the packet is passed to the (10GbE) MAC which transmits it onto the wire.

The traffic generator can also receive incoming packets and produce port- and flow-level statistics. Received packets enter the system through a MAC, and are aggregated through a round-robin input arbiter, before arriving at the statistics collection module. The resulting statistics are relayed to the software using the programmed input/output (PIO) interface.

2.2 Traffic Monitoring

Figure 3 shows the traffic monitoring pipeline which processes (and filters) packets at line-rate and generates high precision timestamps and statistics. We assume that the pipeline processes minimum sized packets at full line-rate, as you would expect from hardware. However, the host may not be able to process packets at line-rate, so we thin traffic in two ways. First, a 5-tuple (protocol, IP address pair, and layer-4 port pair) filter - in the “Core Monitoring” module - identifies flows of interest. Only packets in a *flow of interest* are sent to the software, while all other packets are dropped. Second, we truncate packets to a fixed-length (sometimes called a snap-length) along with a hash of the entire original packet.

Providing an accurate timestamp is critical to monitoring traffic. Our design stamps packets with a 64-bit timestamp as soon as they arrive from the MAC (minimize queueing jitter). The upper 32-bits count seconds, while the lower 32-bits count fractions of a second with a resolution of 233ps.

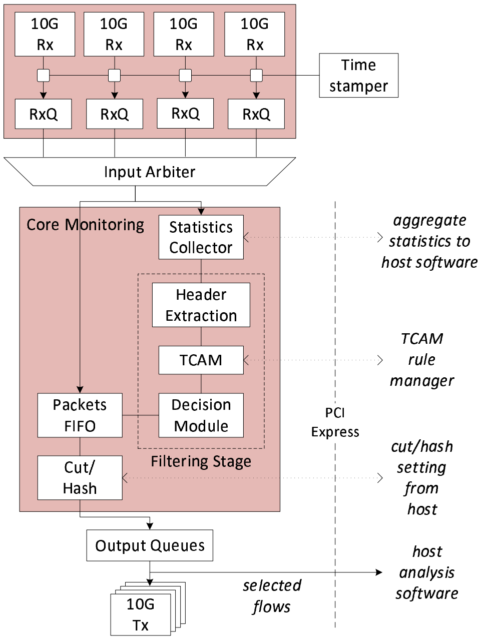

Figure 3: The architecture for OSNT traffic monitoring system.

3. PROTOTYPE AND DISCUSSIONS

By building our prototype on the NetFPGA-10G platform [1], we have inherited a number of constraints from that system. One example of this is in the table size such as the filter TCAMs; whose size is traded directly against the overall design-size.

While the internal NetFPGA datapath can accommodate full line-rate, minimum-sized packets, the PCIe interface lacks the bandwidth to pass unaggregated traffic to the host CPU. This interface uses an MTU of 128 bytes. Without careful packing, a naïve implementation of DMA and device driver may achieve as low as 33.5% utilization (for transactions of 129 byte packets). The use of an opportunistic snap-length (cut) size while adding a unique hash of the complete packet, can significantly improve the utilization of the PCIe. Despite the fact that the hash adds an overhead of 128 bits per packet, it is critical in packet identification which is required for many end-to-end measurements such as latency measurements and packet loss-events. This capability of compressing packet information allows us to scale testing to a maximum packet rate of approximately 21.7 Million packets per second.

4. ACKNOWLEDGMENTS

The project is supported by the NSF CRI program under contract 0855268 and is also partially supported by NSF grant CNS-1261462.

5. REFERENCES

[1] M. Blott, *et al.*. FPGA research design platform fuels network advances. *Xilinx Xcell Journal*, (73), 2010.