Enabling Open-Source High Speed Network Monitoring on NetFPGA

Gianni Antichi, Stefano Giordano
Email: <first.last>@iet.unipi.it
Department of Information Engineering
University of Pisa

David J. Miller, Andrew W. Moore
Email: <first.last>@cl.cam.ac.uk
Computer Laboratory
University of Cambridge
Introduction

• Why monitor
  – Security
  – Billing
  – Performance and provisioning
  – Research

• Active (eg NLANR AMP project) vs passive monitoring solutions

• Challenges
  – Packet loss
  – System load
  – Cost
Challenges (con’d)

- Series vs parallel monitoring
  - Parallel
    - Copper network links require an expensive active tap
    - Passive optical splitters are inexpensive
  - Series
    - Cheap
    - Offers possibility of building an Intrusion Prevention System
    - Extra latency
    - Risk of interruption of the link
“The problem ... is choice.”

- Characteristics of an ideal measurement/monitoring solution:
  - Accurate timestamps [better]
  - Guarantee no loss of information (or at least records exactly where records have been lost) [faster]
  - Inexpensive [cheaper]

- Pick any two!
  - Software solutions are
    - cheap
    - adequate for low-speed networks or when timestamp accuracy not critical
    - but don’t scale.
  - Hardware solutions are
    - very accurate (timestamps and packet loss)
    - scale well
    - but can be very expensive.
Motivations

• NetFPGA platform offers a third alternative
  – open and low-cost
  – but with performance of hardware-based solutions
  – including possibility of filtering unwanted traffic
  – flexibility of parallel or series installation
NetFPGA platform

- Line rate, flexible and open platform for research and classroom experimentation
- NetFPGA has:
  - Xilinx FPGA Virtex2Pro.
  - 4 GigE copper ports.
  - 4.5 MB of SRAM and 64MB of DRAM.
NetFPGA architecture

- 8 different rx/tx queues (CPU and physical)
- Unified by Input Arbiter into single packet stream
- Demultiplexed into respective port according to Output Port Lookup
Our Solution (Hardware Plane)

- Packets time-stamped at very earliest possible moment (i.e. at the RGMII interface) to minimise jitter caused by FIFOs
- The Core Monitoring module performs filtering by flow
• Timestamps pass in a side channel parallel with the main packet data path.
Free-running counter?

- We could use a 64-bit timestamp driven by the 125MHz the system clock (naïve solution).
  - Provides no means by which to correct oscillator frequency drift
  - Produces time-stamps expressed in unit of 8 ns
  - Fixed-point representation of time in seconds more useful to host

A more accurate solution…

- DDS (Direct Digital Synthesis)
  - Technique by which arbitrary variable frequencies can be generated using FPGA-friendly purely synchronous digital logic (how DAG works).
  - Need a time reference to correct DDS rate
  - Optimal solution: PPS from GPS receiver
Two problems using the GPS:
- NetFPGA has no easy GPS input.
- The logistics of installing GPS equipment can be quite difficult.

**Independent Rate Controlled Time-stamper (IRCT)**

Two oscillators will not drift at the same time
- We decide to use two local, independent oscillators to estimate drift.
- We use one oscillator as time-stamper and the second to derive a PPS.

We can implement the algorithm entirely inside the FPGA
- NTP would provide better long term stability at cost of short-term jitter and dealing with PCI latency
Filtering stage

• All packets received are retransmitted.

• While the internal NetFPGA datapath can cope with full-rate, minimum sized packets, the NetFPGA PCI interface lacks the bandwidth to record all traffic.

5-Tuple filter stage

• Packets that match one of up to 32 filter rules are also copied verbatim, with their time-stamp prepended, to the host.

• We use the TCAM modules available in Xilinx CoreGen.

• TCAMs are fast and permit on-the-fly rule updates.
Our Solution (Software Plane)

- We modified the NetFPGA kernel device driver to strip off the prepended timestamp and to store this timestamp in the packet’s `struct sk_buff` before the packet is passed on.

- A recent 2.6 kernel that contains a `ktime_t` timestamp in `struct sk_buff` must be used to assure nanosecond granularity.

- We modified libpcap (and tcpdump along with it) to return `struct timespecs` instead of `struct timeval`.

- We provide an auxiliary command-line tool for TCAM rule management which also initialises the hardware timestamp counter with the current date and time.
We also provide a Statistics Daemon (with an interface for NAGIOS) which tracks captured-traffic

- Only traffic that matches the currently loaded rule-set.
- Optionally records the traffic with nanosecond time-stamps to disc in a PCAP-compatible file.

The daemon keeps aggregate counts for:

- Total packets.
- Total bytes.
- Number of IP/non-IP packets.
- Number of TCP/UDP packets.
- Number of TCP SYN/FIN/RST and options.
- Mean inter-arrival time.
- Mean bit-rate.
Performance Evaluation

Time-stamp accuracy

- We used TCPReplay with a real traffic trace as software traffic generator
- nf2c1 stands for one of the four physical ports of NetFPGA board
- eth1 is one of the ports of the NIC connected to the PC where NetFPGA is hosted
Performance Evaluation

Comparison of the two absolute drift with the naïve time-stamping module

![Graph showing the comparison of DAG timestamp 0-based (seconds) with NetFPGA timestamp 0-based (seconds). The graph is a straight line, indicating a linear relationship between the two variables.]
Performance Evaluation

Comparison between the two oscillators with the naïve time-stamping module

![Graph showing relative drift (milliseconds) over time (seconds)]
Performance Evaluation

Comparison between the two oscillator with the IRCT module

-1.8 msec

60 sec
Conclusions and Future Works

• We have introduced a flexible yet cost-effective passive monitoring system based on a cooperative PC/NetFPGA architecture.

• Promising results and compares favourably with a widely-recognised commercial system for traffic while having a significantly lower cost.

• Future works:
  – Port to the shiny new NetFPGA 10G
  – Add support for an external time-base
  – Expand the available pool of selection filters by replacing the TCAM-based filter with a Bloom Filter.
NetFPGA 10G!
Enabling Open-Source High Speed Network Monitoring on NetFPGA

Gianni Antichi, Stefano Giordano
Email: <first.last>@iet.unipi.it
Department of Information Engineering
University of Pisa

David J. Miller, Andrew W. Moore
Email: <first.last>@cl.cam.ac.uk
Computer Laboratory
University of Cambridge