ENDEAVOUR

Horizon 2020 Programme

Open-Source Network Tester on NetFPGA-SUME Platform

22nd June

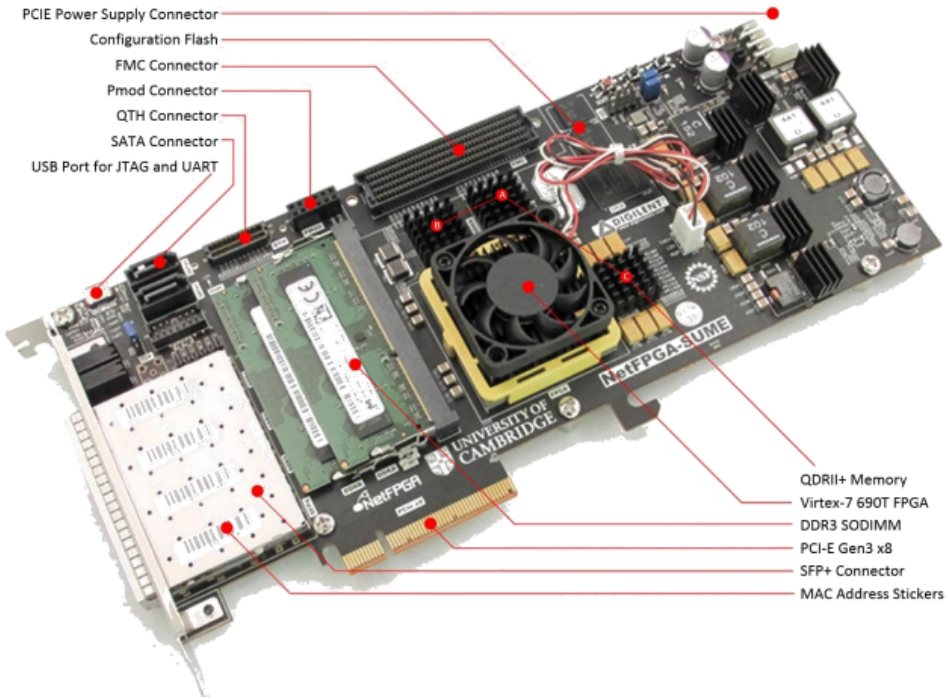
Jong Hun, Han

Computer Laboratory, University of Cambridge

Queen Mary University of London · UCL Université catholique de Louvain · University of Cambridge · LAAS CNRS · Institut Carnot LAAS CNRS · DE-CIX · IBM

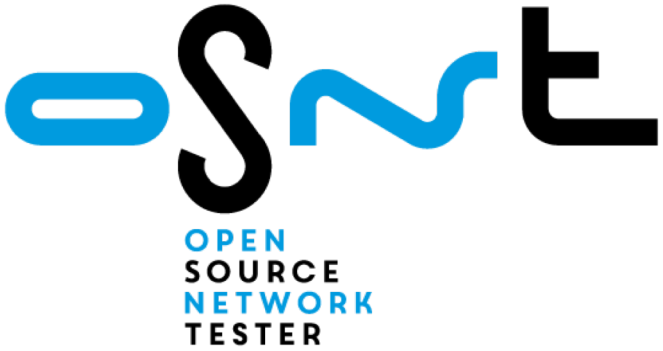
NetFPGA-SUME Specification

- NetFPGA-SUME platform features



- Xilinx Virtex-7 XC7V690T FFG1761-3
- Two 4GB DDR3 SODIMMs 64 bit wide buses clocked at 850 MHz
- Three 72Mbit QDRII+ SRAMs 36 bit wide buses clocked at 500 MHz
- Two SATA III ports
- Micro-SD Card Slot
- Two 512Mbit flash modules
- PCI-E Gen3 x8 supporting 8Gbps/lane
- Four SFP+ interfaces supporting 10Gbps
- HPC FMC Connector

Open-Source Network Tester



- Collaboration project 2014 – Cambridge, Stanford, Princeton, CNRS, and Google.
- Available on the NetFPGA-10G platform
- NetFPGA-SUME port released Feb 2017 with new features

- Open source hardware and software platform for network test, publicly available

  https://osnt.org

  https://github.com/NetFPGA/OSNT-Public/wiki

- Low cost, low jitter, flexible to update, scale-out, no CPU usage, nano-second resolution measurement

Commercial Network Testers

…this has led to a multi-billion dollar industry in network test equipment…

- Commonly closed and proprietary systems
- Limited flexibility
- Well outside the reach of most universities and research laboratories

Network Tester Comparison

- OSNT is flexible, high resolution, and full line rate performance at low cost

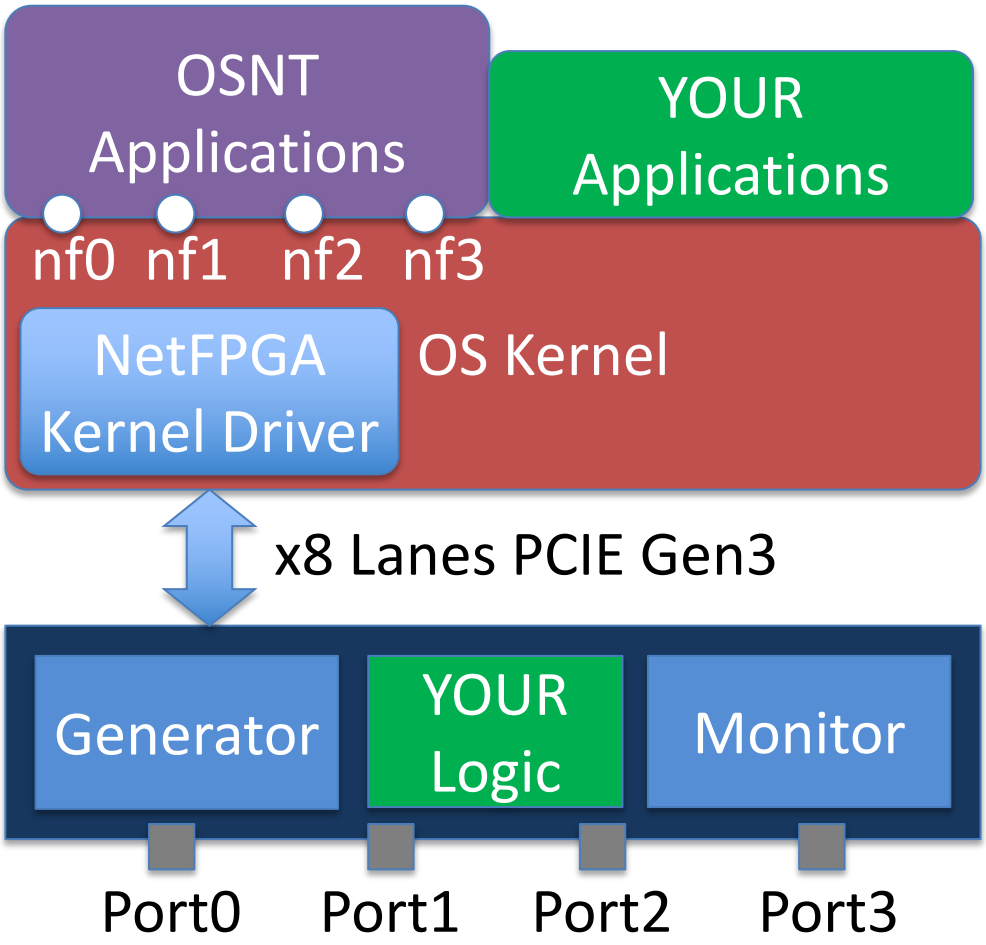
	Cost	Flexibility	Resolution	Line Rate
ixia SPIRENT endace power to see all	$$$ $$$	✗	✓	✓
DPDK, SW tools	($)	✓	✗	(✓)
OSNT	$	✓	✓	✓

OSNT-NetFPGA-SUME Main Features

- Open-Source Network Tester for HW packet traffic generation and monitor.
- Available on NetFPGA-10G and NetFPGA-SUME.
- 4x10Gbps traffic generation and monitor.
- High resolution timestamp at 6.4nsec (156.25MHz).
- GPS synchronized measurement system.
- QDR and DDR3 memory based traffic generator.
- Cut, Hash, and Batch processes to manage capture workload.
- HW and SW for the implementation are open-source.
- Flexible to modify and add more features…

OSNT System Configuration

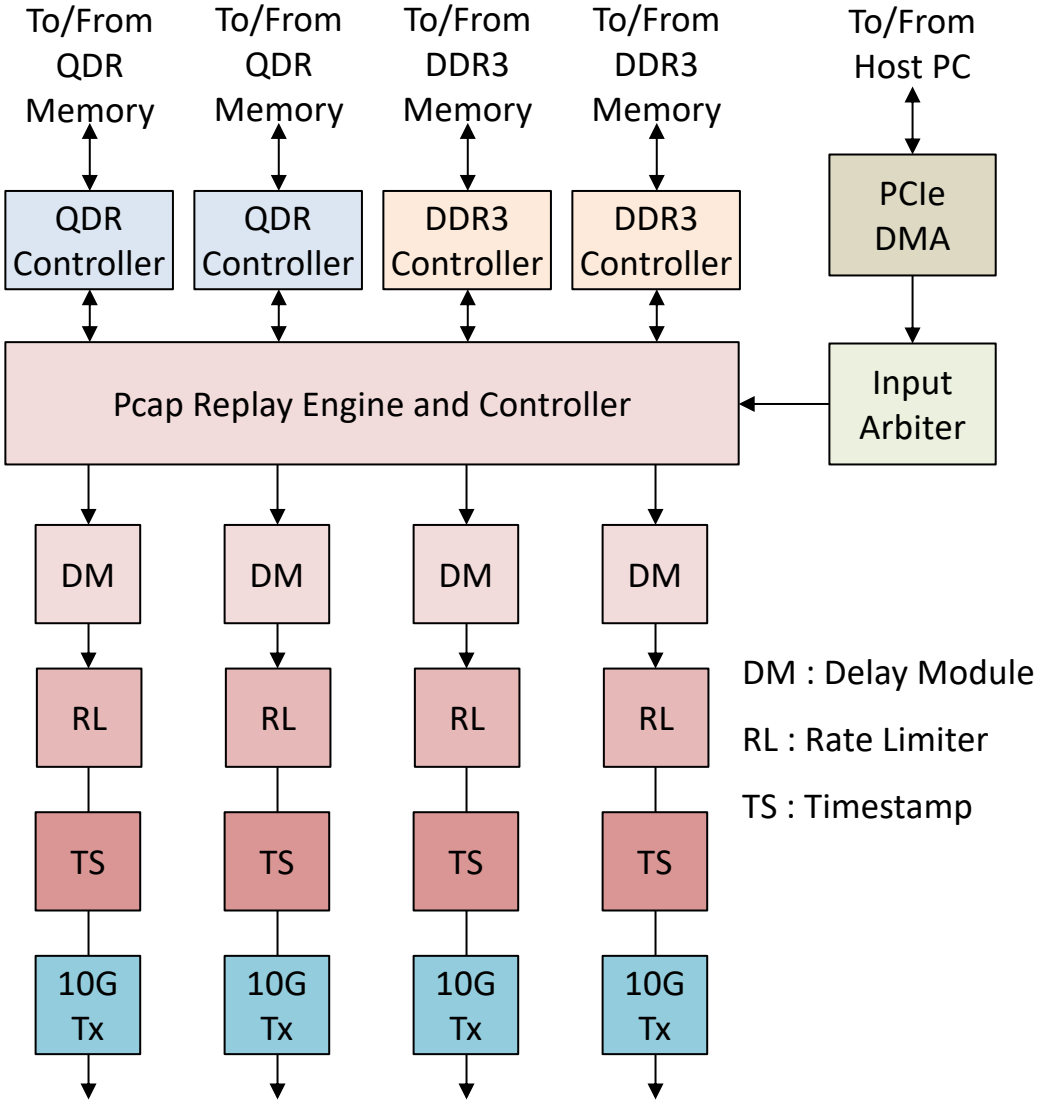
- Configuration of OSNT Systems on the NetFPGA-SUME platform



- Written in C, Python
- Open API and registers

- Written in Verilog HDL
- Users can add and modify the HW modules

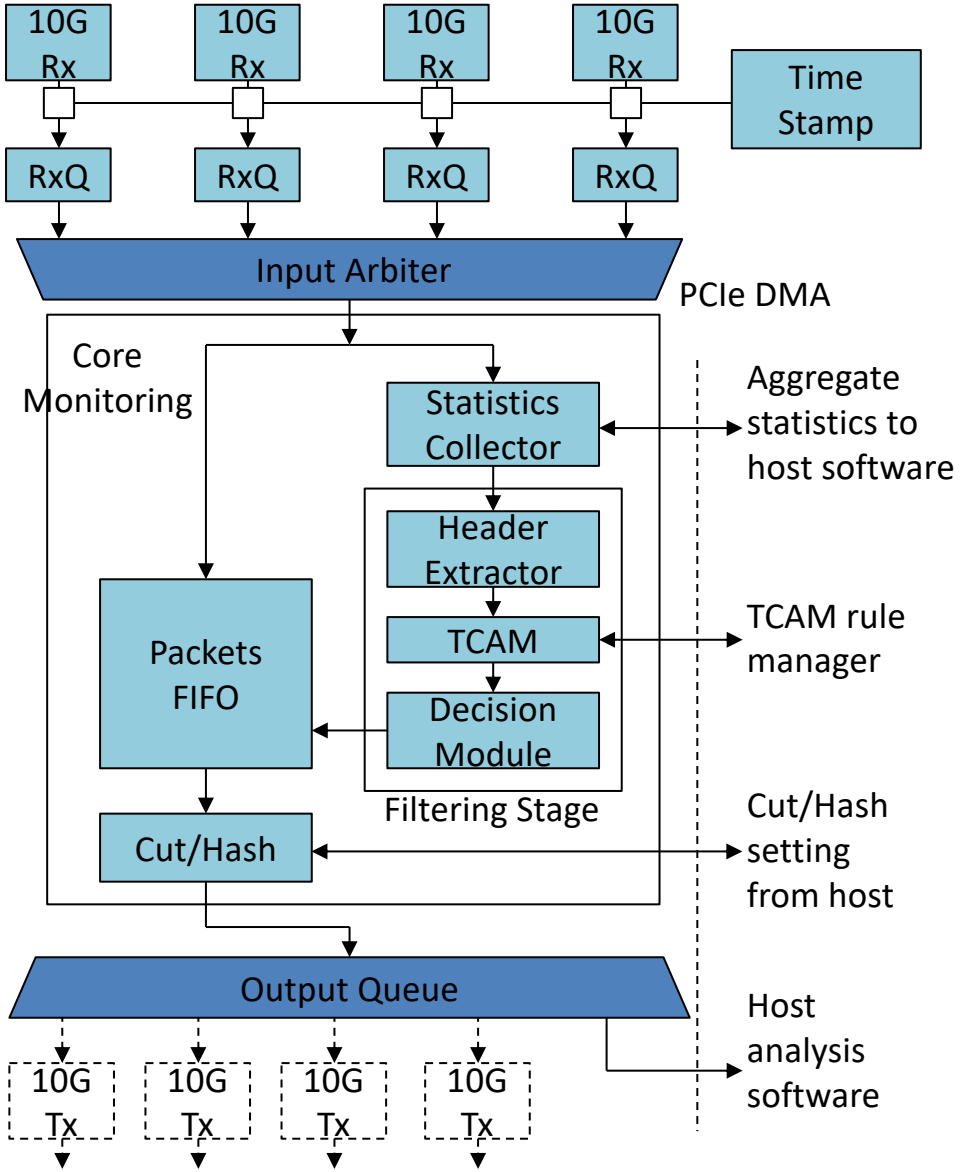
OSNT - Traffic Generator



Traffic generator FPGA HW structure

- 4x10G Pcap replay engine
- Pcap file controller
- Delay module
- Rate limiter
- Tx Timestamping

OSNT-Traffic Monitor

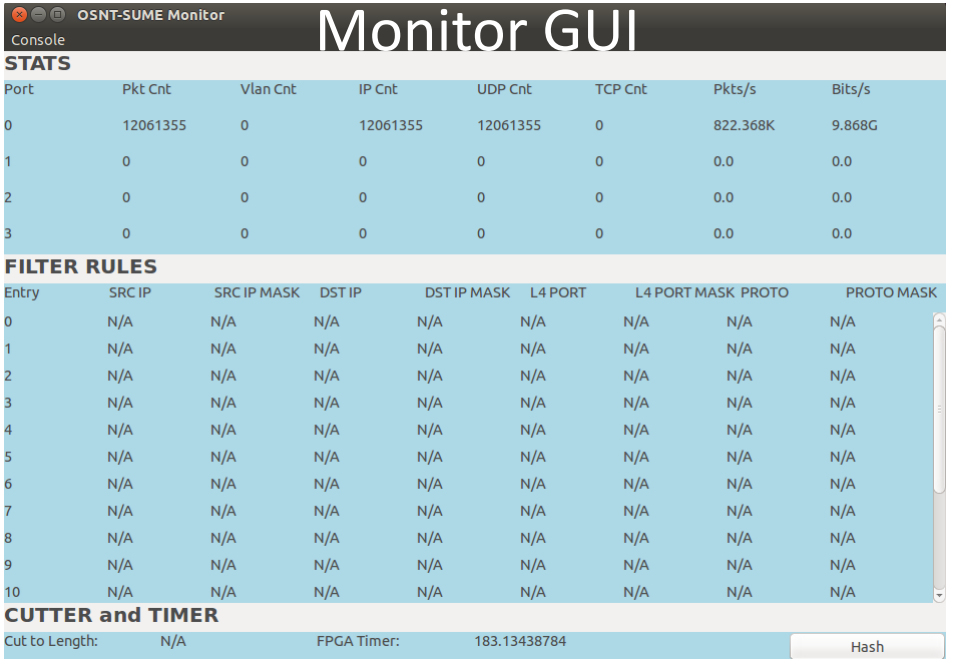


Traffic monitor FPGA HW structure

- Stats collector
- 4x10G packet capture
- TCAM-based Packet Filter
- Cut-Hash function

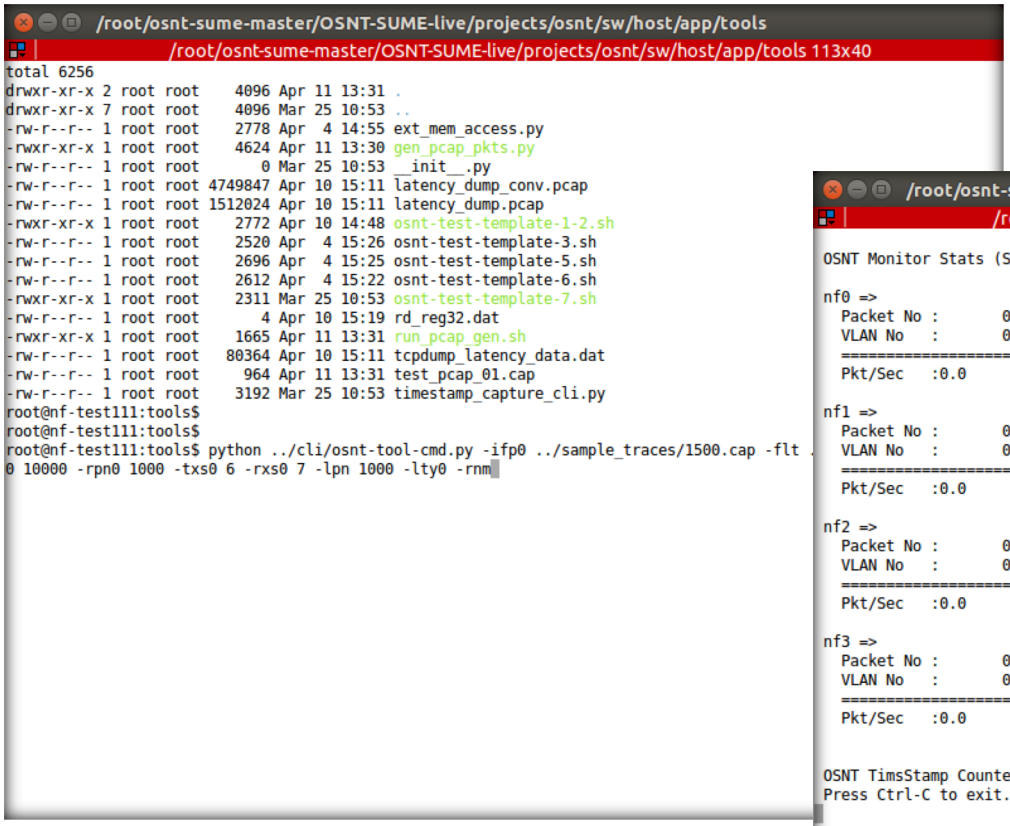
OSNT-Graphic User Interface

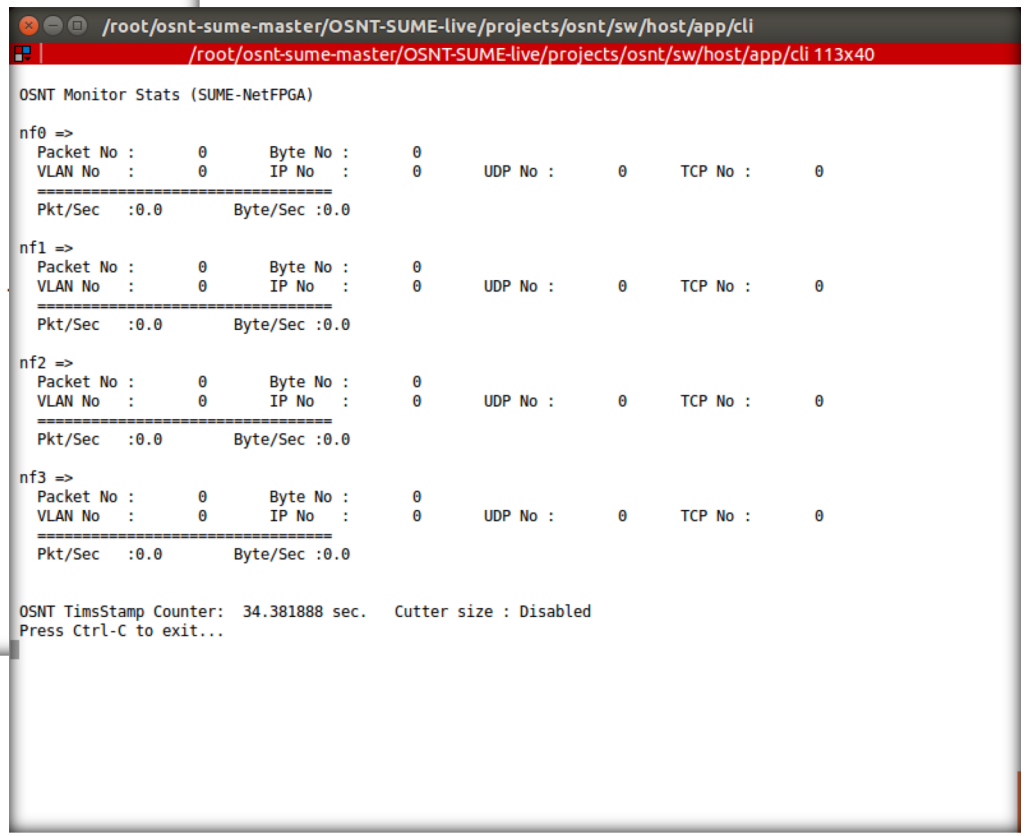
- OSNT GUI – Extensible Generator and Monitor GUI in Python.



OSNT-Command Line Interface

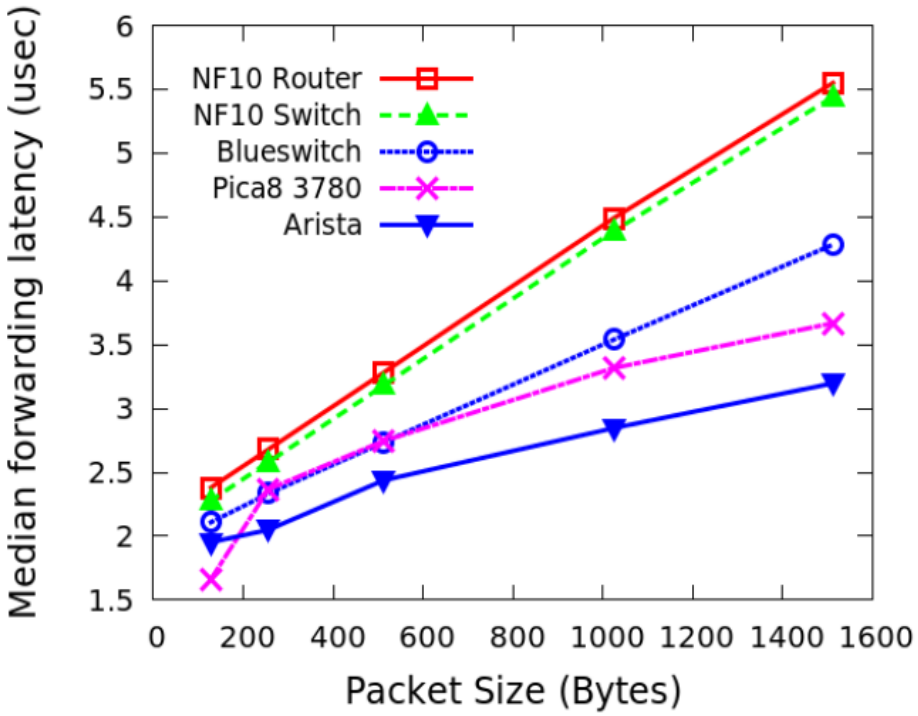
- Command-Line-Interface is also available to create a script automating the test process.

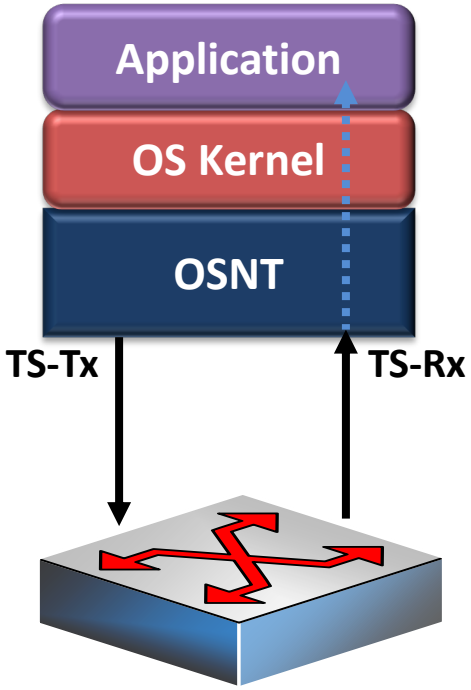




OSNT-Switch Latency Measurement

- Switch latency measurement results with OSNT against different packet lengths

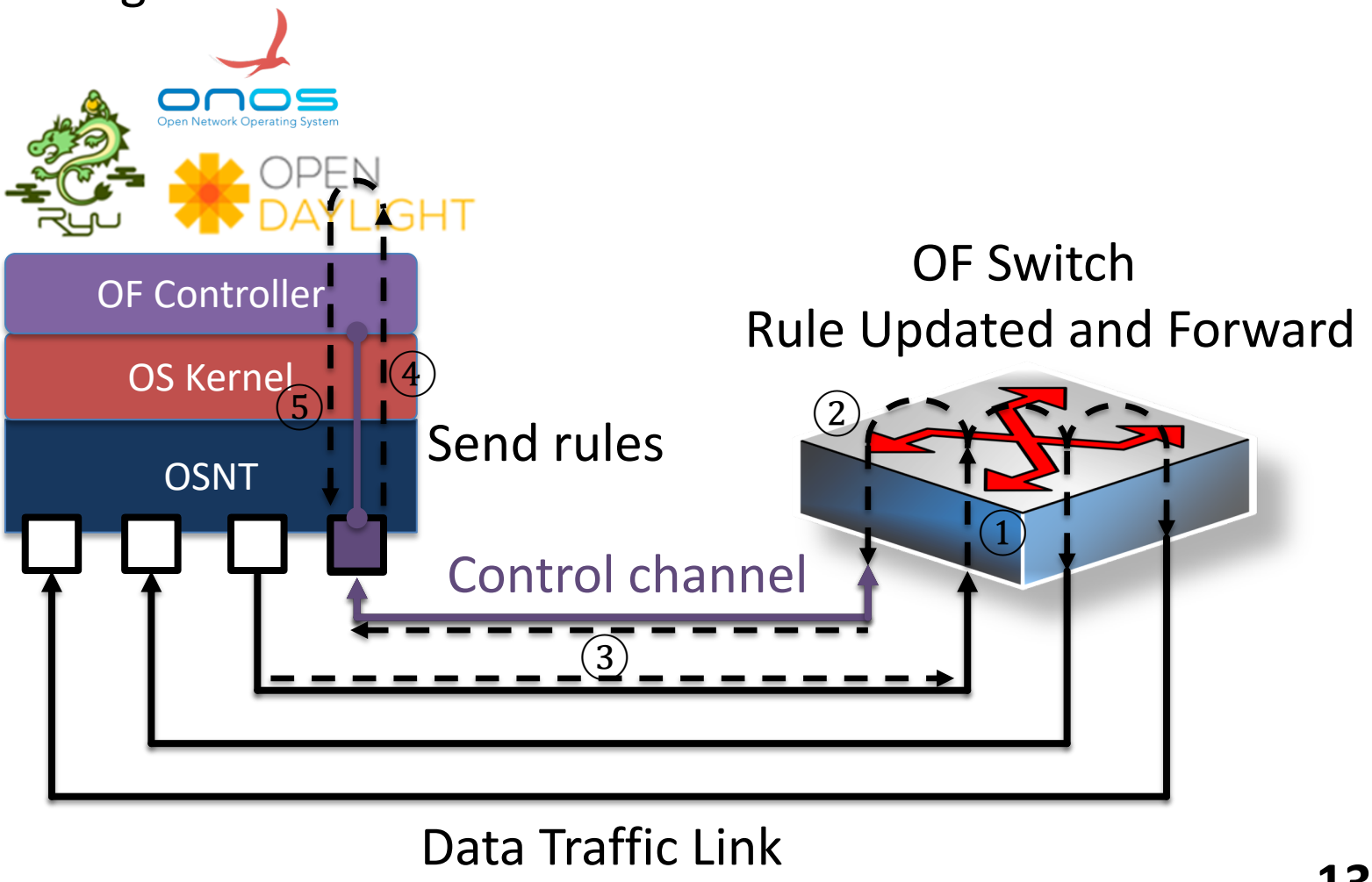




Han J. H., Mundkur P., Rotsos C., Antichi G., Dave, N., Moore A. W, and Neumann P. G., *Blueswitch: Enabling Provably Consistent Configuration of Network Switches*, ANCS 2015

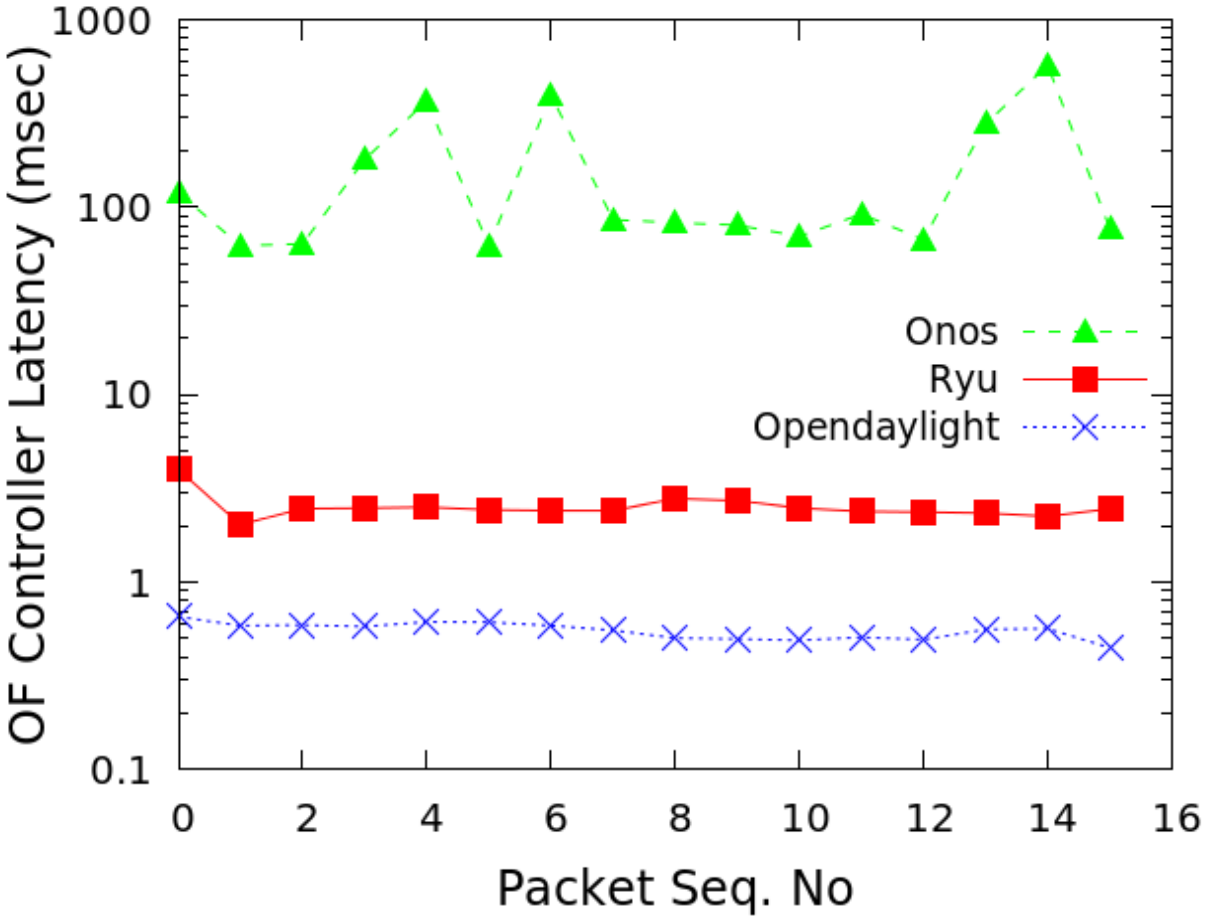
OSNT-OF Controller Latency

- Example Experimental Setup for Fine-Grain SDN measurement – L2 Learning Switch



OSNT-OF Controller Latency

- The latency results of the OF controllers measured by the OSNT – L2 Learning Switch latency between 4 → 5 by the controllers



OSNT-Reproduce Packet Traces

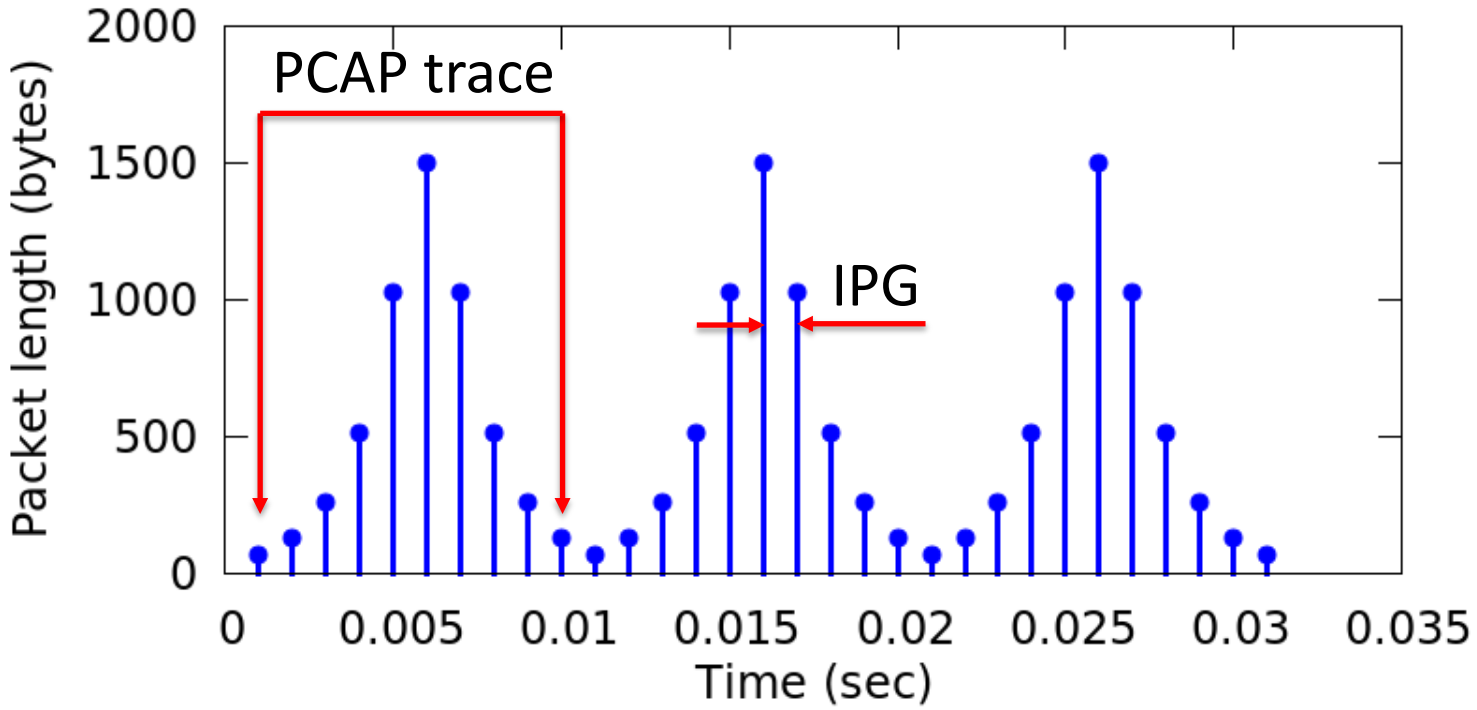
- Traffic generation with the timestamp in the PCAP traces
- Use created or dumped the PCAP traces with the timestamp
- Able to replicate and reproduce the same IPG traffic

Timestamp

```
01:00:00.100000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 22
01:00:00.200000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 86
01:00:00.300000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 214
01:00:00.400000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 470
01:00:00.500000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 982
01:00:00.600000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 1458
01:00:00.700000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 982
01:00:00.800000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 470
01:00:00.900000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 214
01:00:01.000000 IP 192.168.1.1.100 > 192.168.1.2.101: UDP, length 86
```

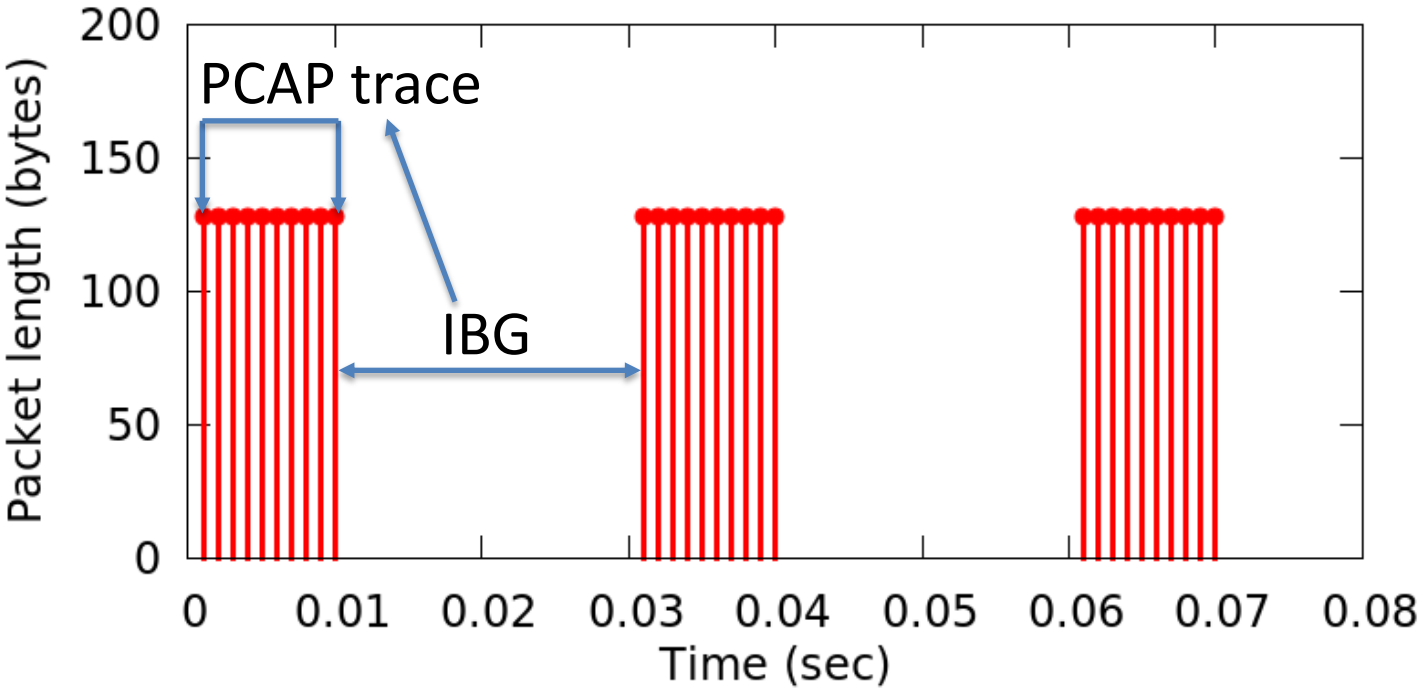
OSNT-Reproduce Packet Traces

- Traffic generation with the timestamp in the PCAP traces
- Use created or dumped PCAP traces with the timestamp
- Able to replicate and reproduce the same IPG traffic



OSNT-Reproduce Packet Traces

- Burst traffic generation with the timestamp in the PCAP traces
- Use created or dumped the PCAP traces with the timestamp
- Able to replicate and reproduce the same IPG and Inter-Burst-Gap (IBG) traffic



OSNT-SUME-live Github

- OSNT-SUME-live is publicly available.

NetFPGA / OSNT-SUME-live Private

Unwatch 20 | Star 0 | Fork 2

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

OSNT for NetFPGA-SUME board

Edit

New Add topics

47 commits | 1 branch | 4 releases | 3 contributors

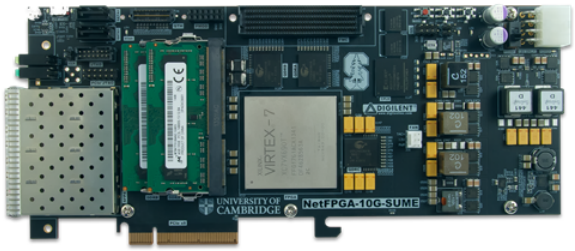
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jhhan Merge branch 'master' of https://github.com/NetFPGA/OSNT-SUME-live Latest commit 1a058a2 7 days ago

contrib/challenge2017	NetFPGA design challenge 2017, added the first test	12 days ago
lib	Update the extmem packet replay controller and add the qdr i/f contro...	7 days ago
projects	Update minor in 10g rx and tx cores. Add comments to the timestamp mo...	a month ago
scripts	Add the first clean repo.	8 months ago
util	update pcap_gen to support configurable source and dest MAC addresses	12 days ago
Makefile	Add Makefile for ip core and sw driver generation.	
README.md	Update the release not link.	

README.md

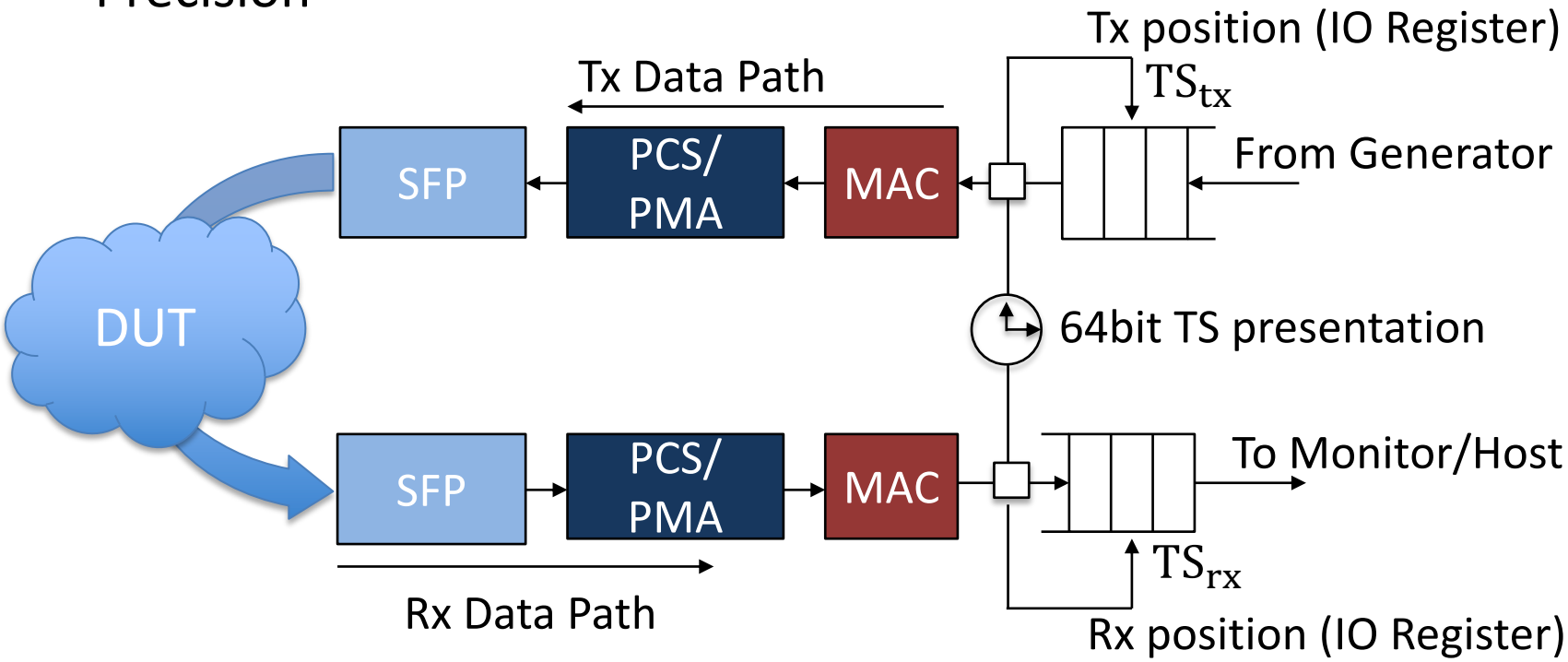
OSNT



Q & A

OSNT-Latency Measurement

- Switch latency measurement with timestamp mechanism
- Timestamp - 64bit Fixed-Point representation with 32bit Precision



DUT Latency

$$\Delta TS = TS_{rx} - TS_{tx}$$