YANG model and implementation of Network Interconnect Tester

- IETF111 Hackathon
- July 19-23, 2021
- Online
The project

Specification:
* [draft-vassilev-bmwg-network-interconnect-tester-06](http://example.com)

Client side:
* Example script – rfc2544.py benchmark ([Python](https://www.python.org))

Device side:
* Software - YANG/NETCONF server instrumentation code ([C](https://www.cprogramming.com))
* Firmware - ([Verilog](https://www.verilog.com))
* Hardware – off-the-shelf FPGA module Ultra96 + 6x SFP+ network programmability kit shield ([KiCAD, Walk-through](https://www.kicad.info))
Setup

IETF Hackathon – YANG model and implementation of Network Interconnect Tester
Design and implementation

NETCONF Server (Model (YANG), Implementation Generator module (C), Analyzer module (C))

<table>
<thead>
<tr>
<th>TRAFFIC-GENERATOR-SW (C)</th>
<th>TRAFFIC-ANALYZER-SW (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket API</td>
<td>Socket API</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Kernel</td>
<td>Kernel</td>
</tr>
<tr>
<td>Sync -&gt;{RTCLOCK}(Verilog)</td>
<td>Sync -&gt;{RTCLOCK}(Verilog)</td>
</tr>
<tr>
<td>PPS</td>
<td>PPS</td>
</tr>
<tr>
<td>DMA</td>
<td>DMA</td>
</tr>
<tr>
<td>[AXI]</td>
<td>[AXI]</td>
</tr>
<tr>
<td>MAC</td>
<td>MAC</td>
</tr>
<tr>
<td>TRAFFIC-GENERATOR-HW (C, Verilog)</td>
<td>TRAFFIC-ANALYZER-HW (C, Verilog)</td>
</tr>
<tr>
<td>GMII_MUX</td>
<td>GMII_MUX</td>
</tr>
<tr>
<td>[GMII]</td>
<td>[GMII]</td>
</tr>
<tr>
<td>PHY</td>
<td>PHY</td>
</tr>
<tr>
<td>SFP+ TX</td>
<td>SFP+ RX</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* - underlined text has links to repositories
What got done

* Completed the rfc2544.py script and validated the binary search algorithm (Python)!

* Improved GPS realtime clock synchronization support (C, Verilog).

* Granted public NETCONF access to tester0, tester7 and tester8 nodes for the duration of IETF111 (link) for interoperability testing.
The report – 1 Gb/s lossless loopback connection

tester@spark:~:/litenc/tntapi/example/ietf-network-interconnect-tester# python ./rfc2544.py --config=../topology-localhost.xml --dst-node=tester0 \ 
--dst-node-interface=eth1 --src-node=tester0 --src-node-interface=eth0 --dst-mac-address="70:B3:D5:EC:20:01" --src-mac-address="70:B3:D5:EC:20:00" \ 
--dst-ipv4-address="192.0.2.2" --src-ipv4-udp-port=49184 --src-ipv4-address="192.0.2.1" --frame-size=64 --trial-time=120 --speed=1000000000 \ 
| grep "^#"

===Throughput===
#1 1488095.238095 pps, 20 octets interframe gap, 100.00% ... 178571428 / 178571428
#Result: 1488095.238095 pps

###Latency===
#Measurement style - bit forwarding
#1 896 ns (min=864 ns, max=896 ns) ... 178571428 / 178571428
...  
#20 904 ns (min=872 ns, max=904 ns) ... 178571428 / 178571428
#Result: 900.800000 nanoseconds

###Frame loss rate===
#1 100% rate, 0% loss, (100.000000% rate actual), 1488095.238095 pps (1488095.238095 pps actual), 20octets interframe gap ... 178571428 / 178571428
#2 90% rate, 0% loss, (89.361702% rate actual), 1339285.714286 pps (1329787.234043 pps actual), 30 octets interframe gap ... 159574468 / 159574468

###Back to back frames===
#1 2 back-to-back frames ... 120 / 120
#2 4 back-to-back frames ... 240 / 240
...  
#21 1488095 back-to-back frames ... 89285707 / 89285707
#The back to back search is limited to bursts below 1 second.
#Result: >= 1488095

tester@spark:~:/litenc/tntapi/example/ietf-network-interconnect-tester#
Model change from -05 to -06

```yang
module: ietf-traffic-analyzer

augment /if:interfaces/if:interface:
  +--rw traffic-analyzer! {ingress-direction}?
    |  +--rw filter! {filter}?
    |  |  +--rw type identityref
    |  |  +--rw ether-type? uint16
    |  +--rw capture {capture}?
    |  +--rw start-trigger
    |   |  +--rw (start-trigger)?
    |   |   +--:(frame-index)
    |   |   |  +--rw frame-index? uint64
    |   |   +--:(testframe-index)
    |   |   |  +--rw testframe-index? uint64
    |   +--rw stop-trigger
    |     +--rw (stop-trigger)?
    |     +--:(when-full)
    |     |  +--rw when-full? empty
  +--ro state
    +--ro pkts? yang:counter64
    +--ro octets? yang:counter64
    +--ro idle-octets? yang:counter64 {idle-octets-counter}?
    +--ro errors? yang:counter64

...
Remaining model work

* Filter configuration model for the traffic analyzer
  * ietf-access-control-list based (rfc8519)
  * bitfields based (simpler to implement and faster)
* Other