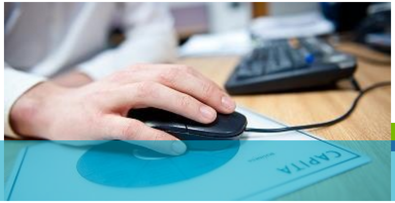
CAPITA



Network Function Verification using Open Source Solutions

3rd SIG-PMV @ NORDUnet, Copenhagen, 2017-11-28

Outline

- Who: My background
- What: Initial research on open source network function testing
- Why: If you haven’t tested it, it doesn’t work
- When: Ad-hoc not continuous, soak testing/break-fix
- Where: Production WANs and lab environments
- How: Simulate network traffic using open source software
- Examples: My findings and lessons learnt (so far)
- Future work: Planned areas of testing and tooling development

Background

- James Bensley / Platform Architect / SP NetEng + Linux + coding
- Updata Infrastructure (Capita Networking Services)
- Originally formed in 2003, CNS-WAN has 500+ employees today
- ISP and Managed WAN Provider
- UK based LLU provider (CLEC)
- Present in 1200 exchanges UK wide
- Business, Local Gov, Edu, Health Care, Fire & Rescue, Police

What: Open Source Network Function Validation

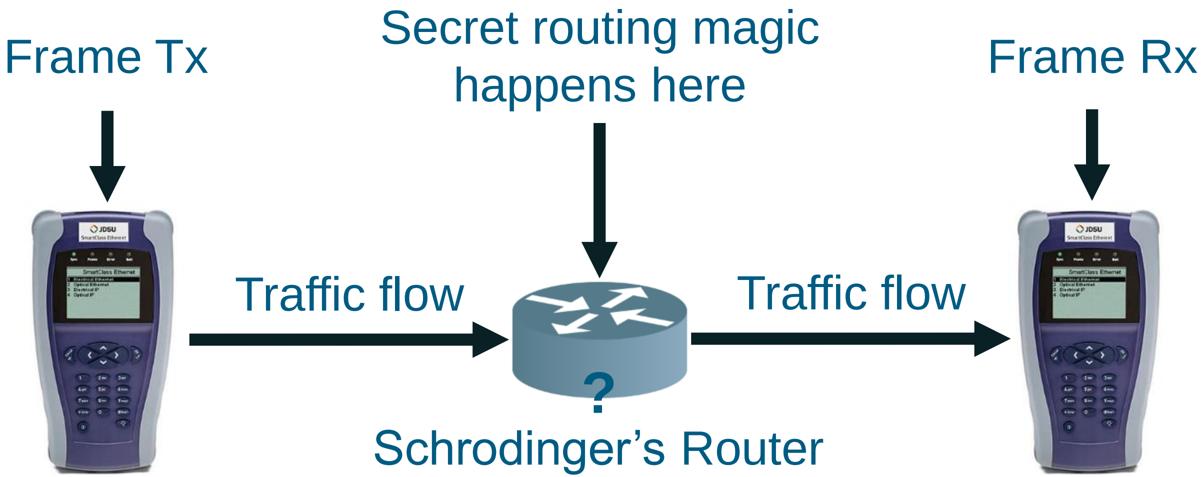
Open source, synthetic, in-band, network function validation and performance testing:

- Open Source: Specifically OSS runs on Linux (happy to pay but not to compromise on scale/features/security/standards compliance)
- Synthetic: This means the ability to simulate production traffic (or replay it) to provide reliable A/B testing
- In-Band: Focus on the data plane level (many operators are solving ctrl/mgmt plane using CI/CD processes)

What: Open Source Network Function Validation

- Network Functions: Not all traffic passes through the same set of network functions when using NFV or SR:
  - Load-balancing e.g. (un)-ECMP/LAG/Geo/Session
  - Fast-reconvergence e.g. IP FRR (r)LFA/MPLS-TE FRR/BGP PIC
  - Traffic filtering e.g. BUM filters (Ethernet)/uRPF (IP & MPLS)

Why: Fail-safe and assume it doesn’t work



Frame Tx

Secret routing magic happens here

Frame Rx

Traffic flow

Traffic flow

?

Schrodinger’s Router

Was the *exact* same frame received that was sent?

http://www.testequipmentdepot.com/viavi/images/csc-ethernet.jpg

Why: Fail-safe and assume it doesn’t work

Testing in general:

- “If you haven’t tested it, it doesn’t work!”
- Black box vendor devices with undocumented behaviour
- Troubleshooting might require TAC assistance and/or hidden commands
- Currently black box vendor devices being tested with black box testers
- “We can’t test every possibility in the lab”

Where: Ideally in the lab

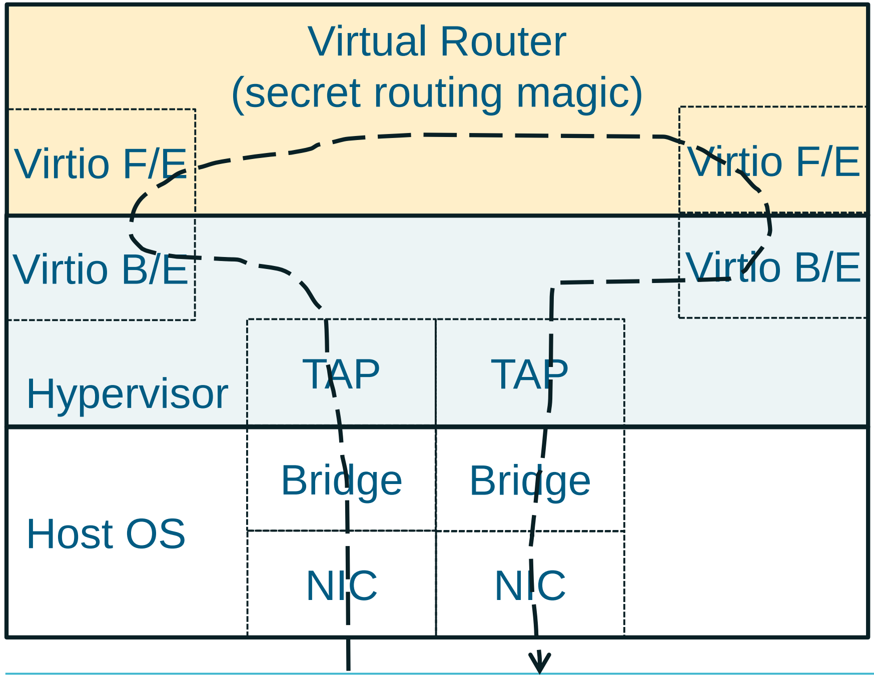
- Mainly testing physical devices in the lab (sometimes in production)
- However, we’re exacerbating the issue with the rise of NFV and COTS (SD-WAN, vCPE, vPE, vRR, vBNG)
- We’re running black box virtual network functions!
    - Obvious issue: COTS is by definition not task optimised
    - Less obvious issue: how many additional variables COTS introduces
- NFV is an interesting area for open source (OVS+DPDK, VPP)
- *Caveat: white box hardware is not open source hardware*

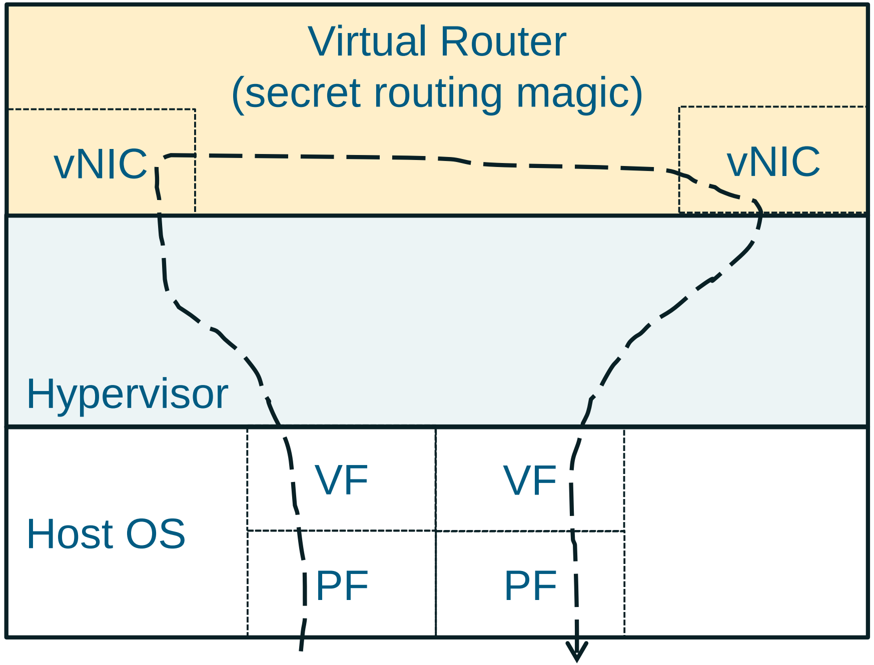
Where: Physical network devices

- Historically only expensive hardware testers could test high bandwidth links
- Initially testing layers 2/2.5 (Ethernet/MPLS) and moving into layer 2/2.5 VNFs
- Both lack security, MPLS is for transport and has no encapsulation support:
  - E.g. does it even work? Hashing on Broken Assumptions
  - E.g. ECMP with L2 and L3 VPNs is inherently flawed due to a heuristic methodology. Old problem BCP128 (2007), still an issue draft-ietf-pals-ethernet-cw-00.txt (2017!)

Where: Virtualisation paths

Traditional vs. modern VNF network path on Linux:





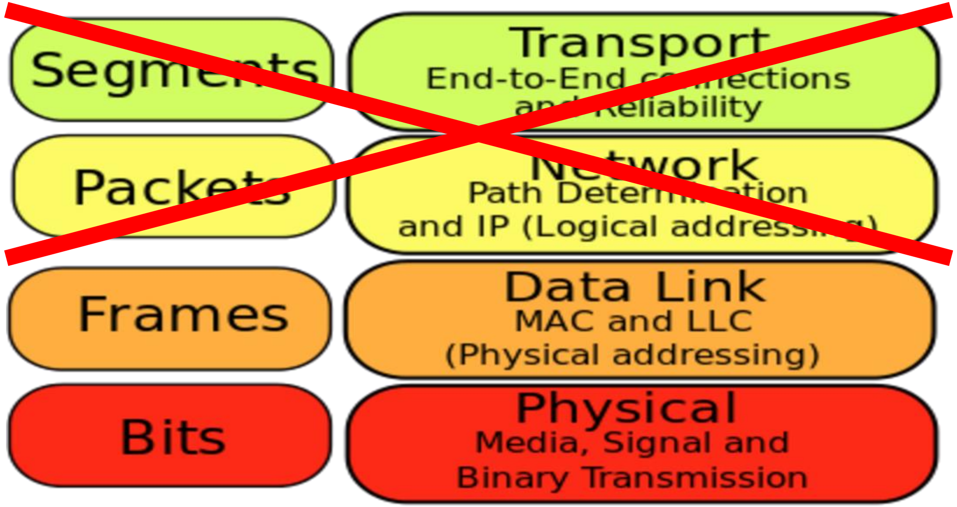
When: Ad-hoc not continuous

- Staging and soak testing phases for individual network functions
- A localised approach, device-by-device, feature-by-feature
- Support automated benchmarking and CI/CD processes with data-plan testing in virtual-labs

How: Linux & OSS PMV Tooling

- Passive, synthetic, in-band traffic generation (or replay)
- A localised approach, device-by-device, feature-by-feature
- Traffic volume (bandwidth) is rarely an issue the focus is more on functionality, but still in scope to fully move away from black box hardware testers
- Prefer open source to maximise on features/distribution/support/bug fixes etc.

How: Linux & OSS PMV Tooling



iPerf, Trex, Scapy

MoonGen, Pktgen

Pktgen, MoonGen, Etherate

CRC/FCS, ECC

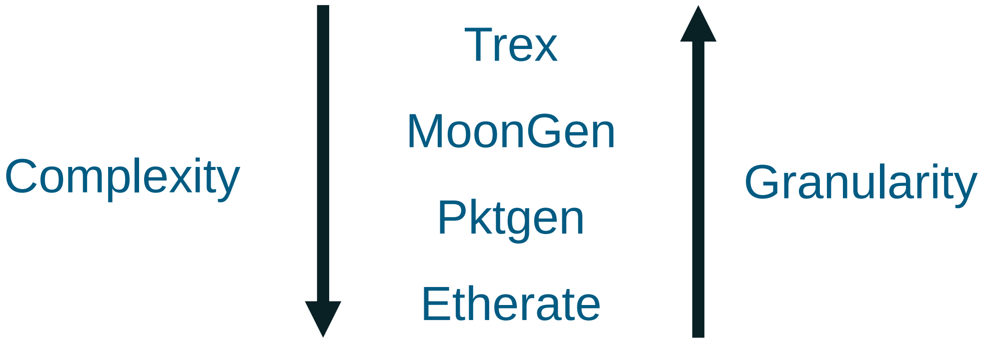
https://commons.wikimedia.org/wiki/File:Osi-model-jb.svg

How: Linux & OSS PMV

No transport layer testing:

- Including the end-user/end-device networking stack is like Russian roulette
- i.e. TCP OS implementations are too varied and too complicated
- iPerf**3** in UDP mode, *only* of if you have to
- Recent Improvements in UDP Packet Processing: UDP throughput went from 1.2 Mpps in Kernel 4.9 to ~2.25 Mpps in Kernel 4.13
- Two raw socket applications on Linux aren't even the same

How: Tooling Comparisons



Complexity

Trex

MoonGen

Pktgen

Etherate

Granularity

How: Linux & OSS PMV

Etherate: Raw socket based Ethernet and MPLS packet generator

- Any layer 2/2.5 header value (load packet-as-hex fall back) ✓
- Constraint based testing (time/speed/volume) ✓
- Easy CLI usage (no API / not scriptable) ✓
- Hardware agnostic ✓
- Lowest performance ✗
- Stateless ✗

https://commons.wikimedia.org/wiki/File:Green_tick_pointed.svg
https://commons.wikimedia.org/wiki/File:Red_X.svg

How: Linux & OSS PMV

Pktgen: DPDK based packet generator using LuaJIT

- Most layer 2-4 header options (load PCAP as fall back) ✓
- “Range” and “sequence” native features ✓
- All options in CLI and Lua API (scriptable) ✓
- Highest performance ✓
- Requires DPDK supported NIC ✗
- Stateless ✗

How: Linux & OSS PMV

MoonGen: DPDK based packet generator using LuaJIT

- Any layer 2-4 header options ✓
- No CLI options, scripted tests only (Lua API) ✓
- Partially stateful ✓
- High(er) performance ✓
- Requires DPDK support NIC ✗
- DPDK EAL settings are hidden ✗

How: Tooling Comparisons

- Etherate assumes two difference devices are being used.

  MoonGen & Pktgen assume the Tx and Rx hosts are the same device.
- Etherate can be used to test a physical device or link at layers 2/2.5.

  Pktgen & MoonGen can be used to test a physical device or link at layers 2-4 for high performing metrics (high throughput or low latency)
- Etherate can also test the raw socket path within the Kernel networking stack.

  PktGgen & MoonGen can also provide some low level NIC stats.

Examples: NF Verification using OSS

Example resources/guides:

- Evolving document: Linux and NFV Testing and Tuning
- Example MoonGen Lua script: generate every Ethertype (0x0000-0xFFFF)
- “We can test every possibility in the lab”

Examples: BUM filter accuracy

NIC: Intel I350 1G, DUT: Cisco 2960, Test: Etherate broadcast test

```
2960#show storm-control fa0/15
Interface  Filter State   Upper        Lower        Current
---------  -------------  -----------  -----------  ----------
Fa0/15     Forwarding       0.25%        0.25%        0.24%
$ sudo ./etherate -i eno2 -g -G -d FF:FF:FF:FF:FF:FF -M 250000
Seconds   Mbps Tx     MBs Tx     FrmTx/s     Frames Tx
1         0.24        0          20          20
2         0.24        0          20          40
```

Examples: Every Ethertype value

NIC: Intel I350 1G, DUT: Cisco 2960, Test: MoonGen “setType” ethertype

Rx NIC drops ~1400 frames, from etype 0x2F to 0x5DC, 0x8100, and 0x888e

Random missing Ethertype chosen and retested, 0x2F == 100% lost

Random working Ethertype chosen and retested, 0x2E == 100% received

0x2E-0x5DC are length values for 802.3 Ethernet + LLC/SNAP (802.2) framing

0x8100 (802.1q VLAN tag): 0 packets input, 65536 runts

0x888e (802.1X EAP): 65536 packets input, 0 errors/drops/runs/discards

“switchport mode access” / No 802.1X configured

Examples: Every Ethertype value

NIC: Intel X710 10G, DUT: ASR9001, Test: MoonGen “setType” ethertype

Rx NIC drops ~8500 Ethertypes

0x8808 (802.3x “pause”) 0 packets input, no NP counters

0x88a8 (802.1ad QinQ/PB), 0x9100 and 0x9200 (802.1q QinQ):

NP Counter: PARSE_DROP_IN_UIDB_TCAM_MISS

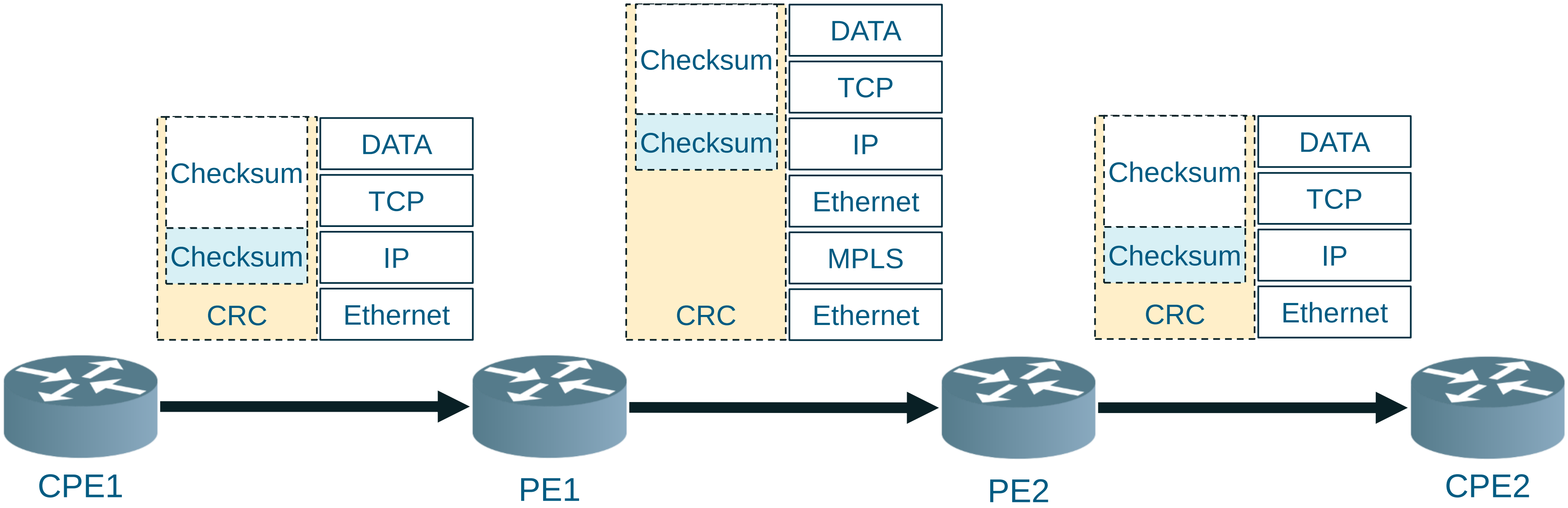
Examples: Every Ethertype value

NIC: Intel X710 10G, DUT: ASR9001, Test: Pktgen performance/size distribution

```
Pktgen:/> set 0 size 247

Pktgen:/> start 0

NP Counters:

PARSE_TOP_LOOP_RECEIVE_CNT         5987587286    6742449

MDF_PIPE_LPBK                      5987589832    6742451

MDF_PIPE_LPBK_BUFFER_PREFETCH      2963853909    3371225
```

Example: In-flight bit errors

MPLS is a transport protocol, not an encapsulation protocol:



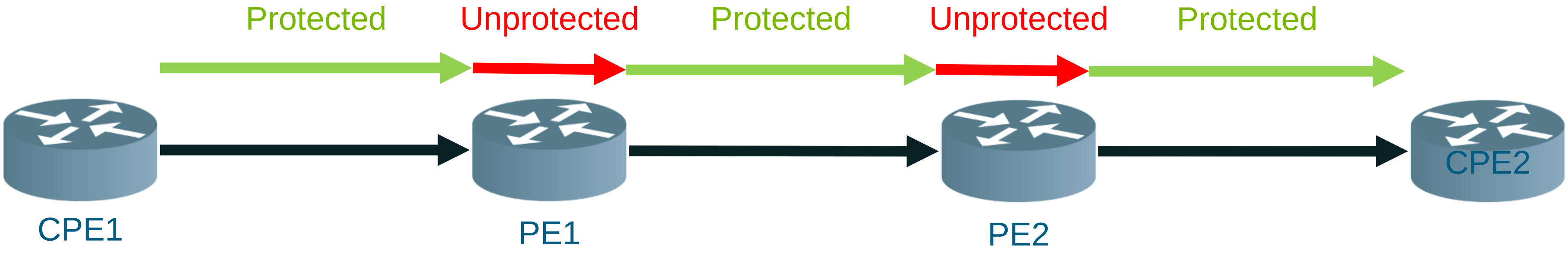
https://commons.wikimedia.org/wiki/File:Router.svg

Example: In-flight bit errors

Some vendors have NPU counters (which are exposed via SNMP):

- E.g. Cisco ASR9K: PARSE_DROP_IPV4_CHECKSUM_ERROR
- E.g. Juniper: bad-IPv4-hdr

No known tooling to test this end to end



https://commons.wikimedia.org/wiki/File:Router.svg

Example: NFV and COTS

Host VNF Paths	VNF Considerations
NAT: slow and inflexible (L3/L4 only)	These techniques are mostly agnostic to the guest VM/application however the Linux native networking stack is slow(-ish)
Linux bridging: slow but flexible	
PCI-PT: fast and flexible, but costly	
SR-IOV: fast, flexible, cost efficient (not perfect yet, e.g. VLANs/Multicast)	Requires VM/application support
Kernel-Bypass (DPDK/Snabb/NetMap); fast, flexible, cost efficient	Proven technologies, limited commercial support adoption
XDP and/or rDMA; native support for Virtio on the horizon	Bleeding edge, immature for now

Example: NFV and COTS

Even with close source solutions we can peak into their performance:

- Open Process Counter Monitor – Intel focused CPU/NUMA/PCI/RAM/Power performance profiling
- Perf “perf_events” – Kernel and application performance using Kernel tracepoints and kropes/uprobes (and more!)
- SystemTap – Kernel and application performance profiling using Kernel tracepoints and function calls/returns (and more!)

Future: Recap of work until now

Research from the past ~year has been presented

- Researched the existing problems (currently using ADE 651!)
- Evaluated the existing open source toolset
- Defining tests to detect known issues
- Trying to fill some gaps in test features

Combine all of the above into an open guide for low level testing.

https://en.wikipedia.org/wiki/ADE_651#/media/File:ADE_651_at_QEDcon_2016_01.jpg

Future: Next steps / key takeaway points

- Evolving document: Linux and NFV Testing and Tuning
- Etherate; frame pacing, bit fiddling, frame checksums
- EtherateMT; coming soon for faster kernel path testing
- MoonGen & Pktgen: Fix RFC2544 test scripts and/or implement ITU-T Y.1564
- Document SystemTap, perf and OPCM examples

Future: Next steps (long term)

- Better equipped and experienced to implement and profile NFV and COTS
- Replace P nodes with VPP+FRR (or equivalent) on COTS for “quick win”
- P4 FPGA for reliable open source hardware tester replacement?

Future: Next Steps (even longer term!)

Linux already supports: IPv6 Segment Routing, VRFs, EVPN, MPLS, LDP

Linux native improvements:

- XDP and eBPF is already being used to provide fast packet processing
- SR-IOV switchdev could be used for routing?
- ~50-60ns IPv4 lookups (single core / DDR) barely supports 10Gbps
- ~450ns IPv6 lookups (single core / DDR)

Can we develop tools for testing the performance of these features?

Questions?

Contact me using these details:

- Email: jwbensley@gmail.com / james.bensley@updata.net
- Slack: http://networktocode.slack.com/
- Skype: jameswbensley

Extra: Easily testable bugs

Example bugs which could have been easily caught with better testing:

- PPPoE unsupported over L2VPN (undocumented ASR920 core encapsulation)
- Interop LDP PWE3 label request (IOS-XR CSCux80490)
- LPTS denying OSPFv3 incorrectly (IOS-XR CSCui29635)
- NPU cache misses causes 33% performance drop (IOS-XR CSCvf44769)

Extra: SIG-PMV overlap with IETF

- IETF: Benchmarking Methodology Working Group:

  "…the BMWG is limited to the characterization of implementations of various internetworking technologies using controlled stimuli in a laboratory environment. Said differently, the BMWG does not attempt to produce benchmarks for live, operational Networks…"

https://datatracker.ietf.org/wg/bmwg/about/