MNK Labs & Consulting

Novel
P4c Backend: P4 to VPP

MNK Labs & Consulting
https://mnkcg.com

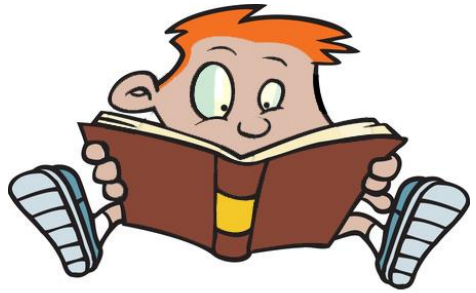
Background

- Any operator asks for virtualized product on server machines first or concurrently with hardware product
- e.g. 5G UPF on server vs. a 5G UPF running on network switch
- MNK has several products running on Intel Tofino switch
- Wouldn’t it be nice to write P4 code once and have code run on both Tofino and server machine as softswitch?
- For server machines, there is DPDK but certain networking apps (e.g. NSH) run significantly faster using Cisco VPP or Fd.io

Need for P4toVPP Compiler

- It took Cisco several years to port Starent cellular hardware product to server machine and VPP.
- If a p4c existed to emit VPP plugin, one writes P4 code once for asic and have code also run on server with VPP – header files change between asic (Tofino/CiscoONE) and server P4 programs
- Server product development velocity goes through the roof
- Virtualized and hardware products are bug-for-bug compatible

Prior Work



- PVPP (Stanford U., Cornell U., Barefoot, Cisco), around 2017, used p4c bmv2 JSON and with scripts and COG generator, produced VPP plugin source code
- If JSON is modified or bugs in JSON generation are fixed, PVPP fixes bugs in two places – bmv2 and PVPP compiler
- PVPP tested IPv4 router forwarding which uses small subset of P4 and VPP features

P4toVPP p4c Backend

- A p4c backend to generate VPP plugin fixes issues in one place – MNK has a new p4c VPP backend
- Supports v1model.p4
- All optimizations (e.g. single metadata, caching table pointers, single-node) in PVPP also exist in MNK's p4c VPP backend
- MNK added another optimization – with compiler flag, run 2, 4, etc. packets in VPP vector
- New backend tested with 5G UPF P4 program using comprehensive P4 and VPP features

Benefits of p4c Backend

- Writing repetitive loop unrolling code causes bugs, e.g., see https://tinyurl.com/y6wl4bjr
- See `vpp/src/plugins/gtpu/` plugin bug history caused by changing 2 to 4 packets in VPP vector
- A compiler solves such bugs. Ours includes a flag to ask user for how many packets to process in vector
- VPP plugin generator only generates feature enable/disable CLI/API
- Our p4c backend is better: generates all control plane CLI and API for VPP plugin, e.g., table entry add/delete/show

Performance

- Our p4c backend generated UPF VPP plugin performs as well as NAT44 VPP plugin
- Another manually-written UPF as VPP plugin didn’t work for us to compare with
- In the test, ipv4 was used with 5G 6-tuple lookup, gtpu decap, gtpu encap and checksum computation
- Traffic tests at 2.65 Gbps with IPv6 GTPU tests described earlier on modest hardware (we don’t have latest Intel machines) with one cpu
- VPP IPv4 FIB lookup and mac rewrite on same hardware runs at 3 Gbps with one cpu

Thank you