DENOG 14
GERMAN NETWORK OPERATORS GROUP

BGP/OSPF at >100Mpps (on amd64)

VPP + Linux Control Plane

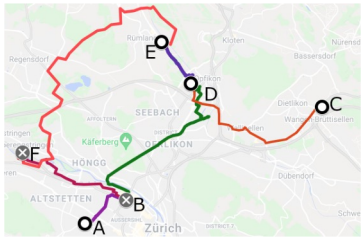
Pim van Pelt <pim@ipng.ch> • 2022-11-15 • Hamburg, DE

Introduction

Pim van Pelt

Pim van Pelt (PBVP1-RIPE)

- Member of the RIPE community since 1999 (RIPE #34)
  - Has used pim@ipng.nl for 23 years
  - And also pim@ipng.ch for 16 years
  - Incorporated ipng.ch in Switzerland in 2021

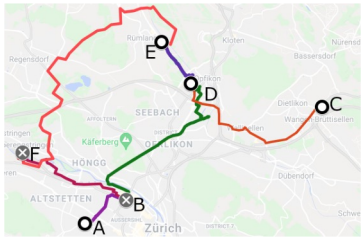


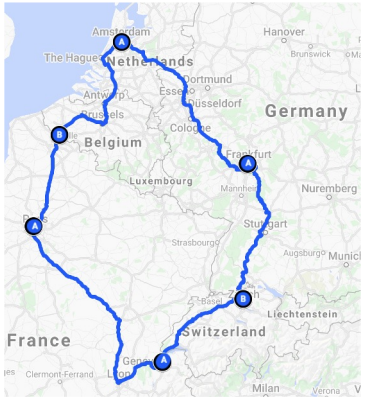
Introduction

IPng Networks GmbH

- Developer of Software Routers - VPP and DPDK [ref]
- Tiny operator from Brüttisellen (ZH), Switzerland [ref]

Introduction

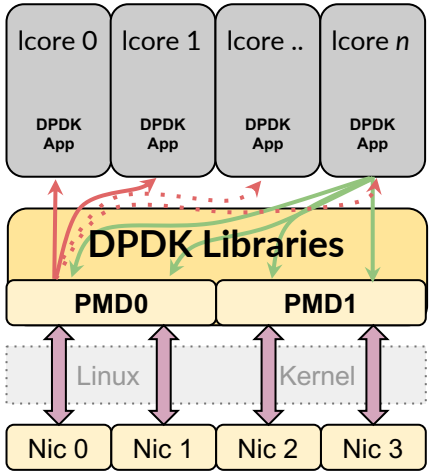




IPng Networks GmbH

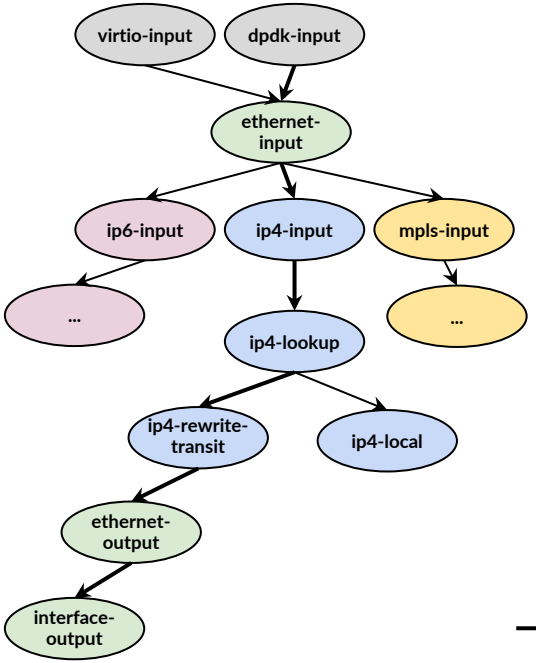
- Developer of Software Routers - VPP and DPDK [ref]
- Tiny operator from Brüttisellen (ZH), Switzerland [ref]

- Twelve VPP/Bird2 routers [ref] (UN/LOCODE names)
- European ring: *peering on the FLAP** [ref] ~1850 adjacencies
- Acquired AS8298 from SixXS [ref]



DPDK architecture

- Runs in Linux userspace (either host or VM guest)
- Kernel bypass (eg. SR-IOV, UIO, VFIO) for device access
- Fully consumes CPU (one `pthread` per logical core)
  - Implements various *Poll Mode Drivers* (PMD)
  - PMDs offer *hardware offload* capabilities
  - Lockless queues, buffers, hash tables, timers, mempools
  - *Run-To-Completion* model: Rx ⇒ process ⇒ Tx
- DPDK threads (*lcores*) subscribe to queue(s) of port(s)
  - Receive Side Scaling (RSS): n threads on one NIC
  - Hashing on IPv4, IPv6, MPLS, VXLAN, l2-payload, ...
  - Tx Queues: Each thread has an output to each NIC

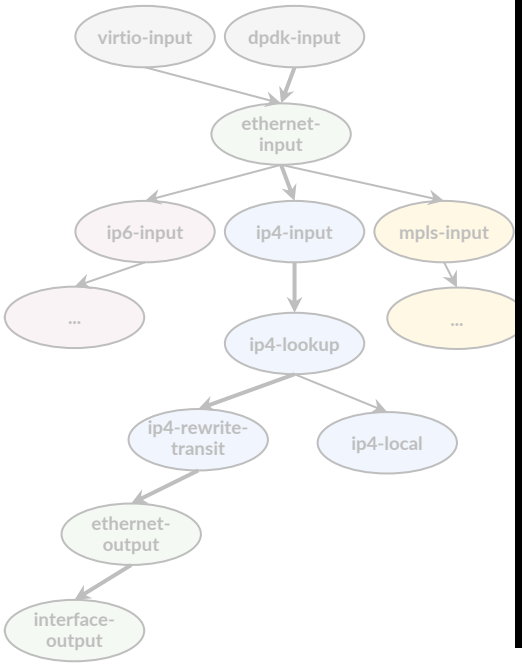


Vector Packet Processing

DPDK reads **a vector** of up to 256 packets from its interfaces:

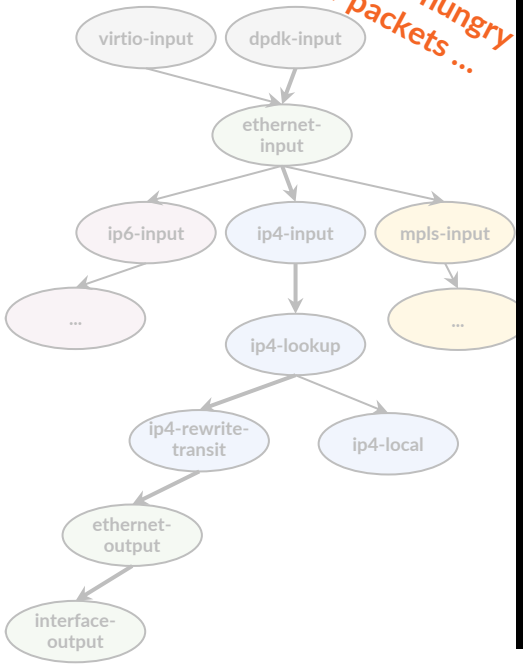
1. Packets are prefetched (or directly written) into CPU **d-cache**
2. All packets go through a directed graph
   a. First packet: graph node's code loaded into CPU **i-cache**
   b. All additional packets: fully in d/i-cache: 7-20x faster
3. Packets then traverse *as a vector* into the next node(s)
   a. Optimized with SIMD (SSE, AVX, AVX512, ...)
   b. No context switches, good TLB hit rate due to hugepages
   c. Lockless: multi-threading gives linear scaling
4. Hardware offload: use silicon if available
5. Plugins: rearrange the graph nodes and add functionality

Vector Packet Processing - example

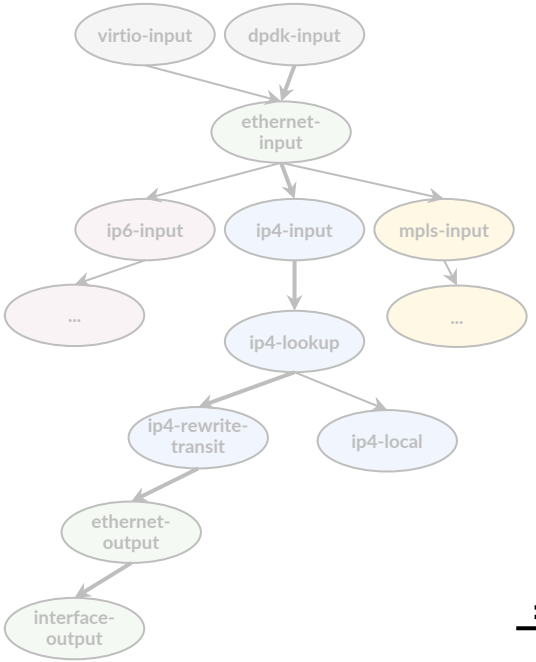



```
pim@hippo:~$ vppctl
vpp# set interface state TenGigabitEthernet3/0/0 up
vpp# set interface mtu packet 9000 TenGigabitEthernet3/0/0
vpp# set interface ip address TenGigabitEthernet3/0/0 2001:db8:0:1::2/64
vpp# set interface ip address TenGigabitEthernet3/0/0 192.0.2.2/24
vpp# ip route add 2000::/3 via 2001:db8:0:1::1
vpp# ip route add 0.0.0.0/0 via 192.0.2.1

pim@hippo:~$ vppctl show interface TenGigabitEthernet3/0/0
TenGigabitEthernet3/0/0  up  9000/0/0/0     rx packets             5969930253
                                            rx bytes            2139798549228
                                            tx packets            14517083897
                                            tx bytes            6864831067486
                                            drops                        945
                                            ip4                   3862409855
                                            ip6                   2107502378
```

Vector Packet Processing - example




```
pim@hippo:~$ vppctl
vpp# set interface state TenGigabitEthernet3/0/0 up
vpp# set interface mtu packet 9000 TenGigabitEthernet3/0/0
vpp# set interface ip address TenGigabitEthernet3/0/0 2001:db8:0:1::2/64
vpp# set interface ip address TenGigabitEthernet3/0/0 192.0.2.2/24
vpp# ip route add 2000::/3 via 2001:db8:0:1::1
vpp# ip route add 0.0.0.0/0 via 192.0.2.1

pim@hippo:~$ vppctl show interface TenGigabitEthernet3/0/0
TenGigabitEthernet3/0/0  up  9000/0/0/0     rx packets             5969930253
                                            rx bytes            2139798549228
                                            tx packets            14517083897
                                            tx bytes            6864831067486
                                            drops                        945
                                            ip4                   3862409855
                                            ip6                   2107502378
```

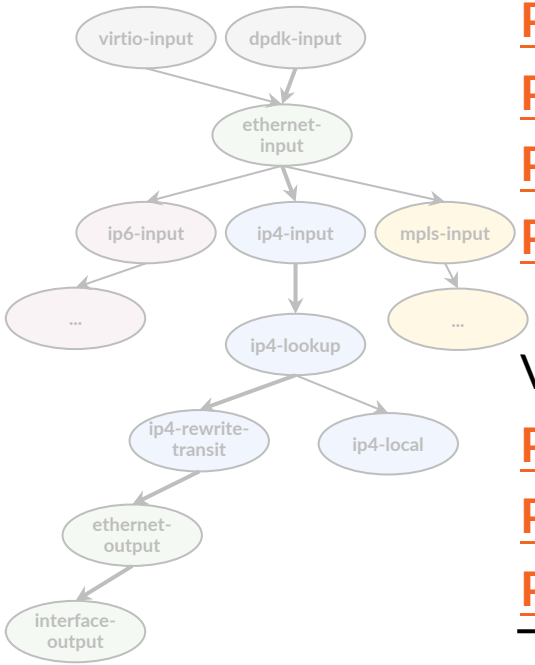


VPP: Linux Control Plane

Wrote github.com/pimvanpelt/lcpng

1. Creates tun/tap interface in Linux for a given VPP interface
   a. VPP->Linux: Traffic to ***ip4-local*** and ***ip6-local*** is punted into TAP
   b. Linux->VPP: packets into TAP are inserted into ***virtio-input***
2. Syncs interface changes in VPP into Linux
3. Listens to ***Netlink* messages** and syncs Linux changes into VPP
4. Allows operators to use VPP almost exactly as if it were Linux
   - Configure interfaces, addresses, routes by hand, or ...
   - ...using common tools like `ip(1)`, FRR, or BIRD/BIRD2

⇒ VPP is Linux's *software equivalent* of an ASIC dataplane ⇐



VPP: Linux Controlplane

For the curious …

Part1 - Create sub-interface (.1q, .1ad, q-in-q, q-in-ad) in Linux

Part2 - Sync link state, MTU and IP addresses in Linux

Part3 - Automatically create sub-interfaces in Linux

Part4 - Netlink: Sync link state, MTU, neighbor, IP addresses, create new sub-interface (.1q, .1ad, q-in-*) in VPP

Part5 - Netlink: Sync routes in VPP

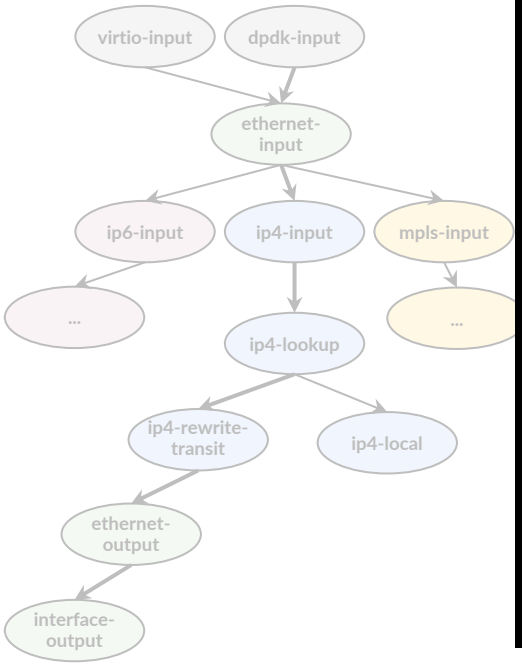
Part6 - Expose interface stats from VPP in SNMP

Part7 - HOWTO: Installation and Configuration in Production

*) *Thanks to Neale Ranns, Matt Smith and Jon Loeliger for the collaboration*

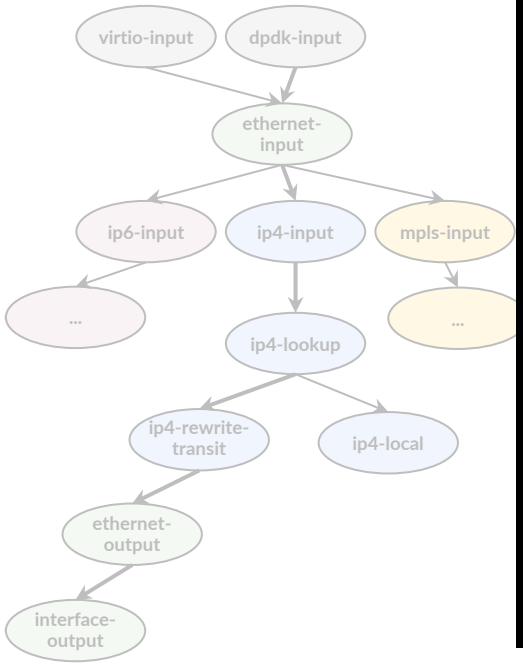
VPP: Linux Controlplane - ip

```
pim@hippo:~$ vppctl lcp create TenGigabitEthernet3/0/0 host-if xe0
```

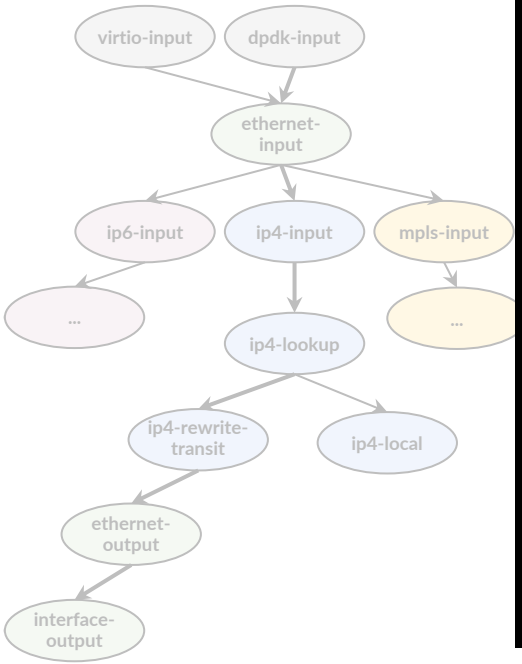


VPP: Linux Controlplane - ip

```
pim@hippo:~$ vppctl lcp create TenGigabitEthernet3/0/0 host-if xe0

pim@hippo:~$ sudo ip link set xe0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev xe0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev xe0
```

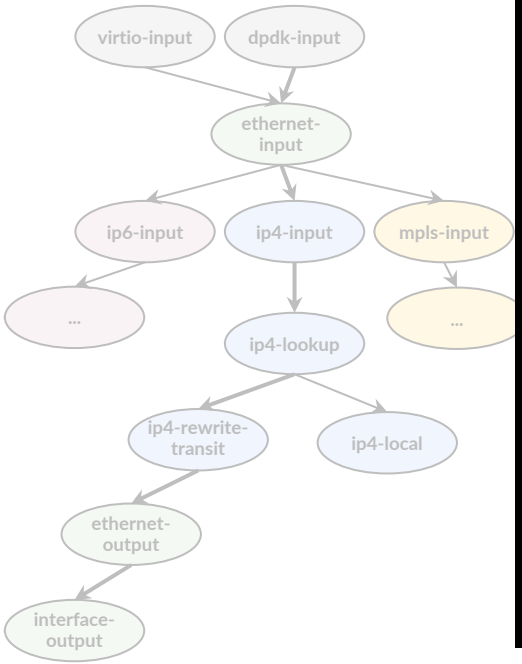


VPP: Linux Controlplane - ip

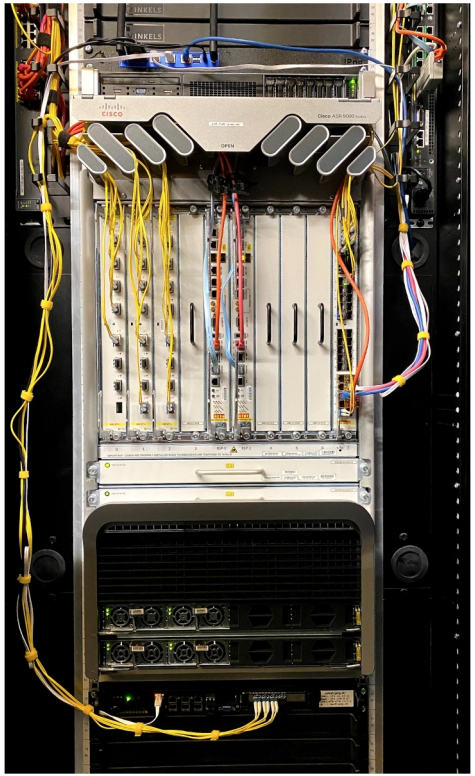



```
pim@hippo:~$ vppctl lcp create TenGigabitEthernet3/0/0 host-if xe0

pim@hippo:~$ sudo ip link set xe0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev xe0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev xe0

pim@hippo:~$ sudo ip link add link xe0 name ipng type vlan id 101
pim@hippo:~$ sudo ip link set ipng mtu 1500 up
pim@hippo:~$ sudo ip addr add 2001:678:d78:3::86/64 dev ipng
pim@hippo:~$ sudo ip addr add 194.1.163.86/27 dev ipng
pim@hippo:~$ sudo ip route add default via 2001:678:d78:3::1
pim@hippo:~$ sudo ip route add default via 194.1.163.65
```

VPP: Linux Controlplane - ip

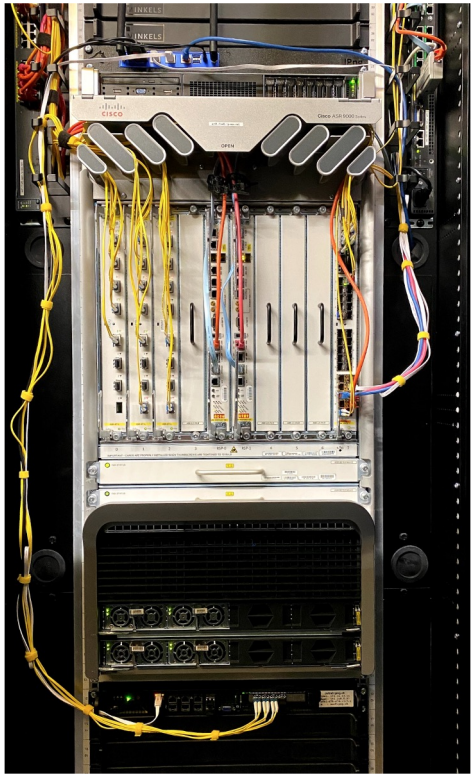



```
pim@hippo:~$ vppctl lcp create TenGigabitEthernet3/0/0 host-if xe0

pim@hippo:~$ sudo ip link set xe0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev xe0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev xe0

pim@hippo:~$ sudo ip link add link xe0 name ipng type vlan id 101
pim@hippo:~$ sudo ip link set ipng mtu 1500 up
pim@hippo:~$ sudo ip addr add 2001:678:d78:3::86/64 dev ipng
pim@hippo:~$ sudo ip addr add 194.1.163.86/27 dev ipng
pim@hippo:~$ sudo ip route add default via 2001:678:d78:3::1
pim@hippo:~$ sudo ip route add default via 194.1.163.65

pim@hippo:~$ ping dns.quad9.net
PING dns.quad9.net(dns.quad9.net (2620:fe::fe)) 56 data bytes
64 bytes from dns.quad9.net (2620:fe::fe): icmp_seq=1 ttl=63 time=0.711 ms
...
```



Running VPP in production

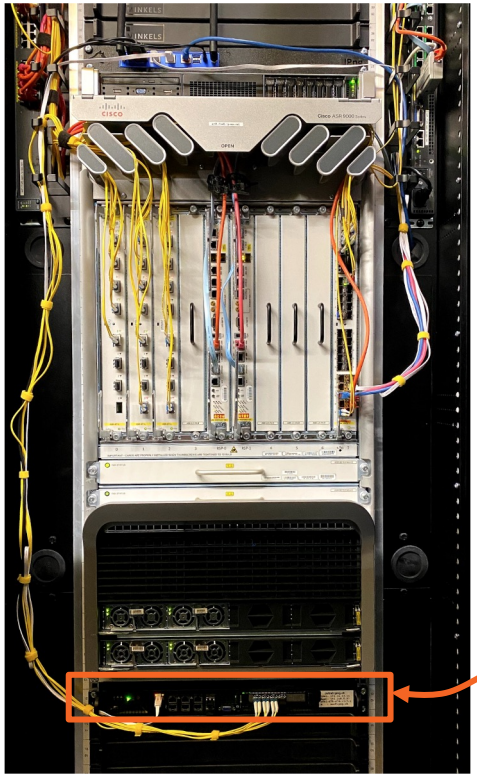
1. Deployed 12x Supermicro 5018D-F8NT
   a. 6x 10G (Intel X710 / X552)
   b. 6x 1G (Intel i350 / i210)
   c. Dedicated IPMI for OOB via AS25091
   d. Throughput ~37.8Mpps at ~45Watt
2. Interconnect 10G EoMPLS: Network Map
3. Uplink IXP and IP Transit 10G: Peering DB
4. Monitor with SNMP: Wrote vpp-snmp-agent
5. Dataplane Management: Wrote vppcfg
6. Controlplane Management: Wrote Kees



Not this one ...
er01.fra05.ip-max.net

Running VPP in production

1. Deployed 12x Supermicro 5018D-F8NT
   a. 6x 10G (Intel X710 / X552)
   b. 6x 1G (Intel i350 / i210)
   c. Dedicated IPMI for OOB via AS25091
   d. Throughput ~37.8Mpps at ~45Watt
2. Interconnect 10G EoMPLS: Network Map
3. Uplink IXP and IP Transit 10G: Peering DB
4. Monitor with SNMP: Wrote vpp-snmp-agent
5. Dataplane Management: Wrote vppcfg
6. Controlplane Management: Wrote Kees



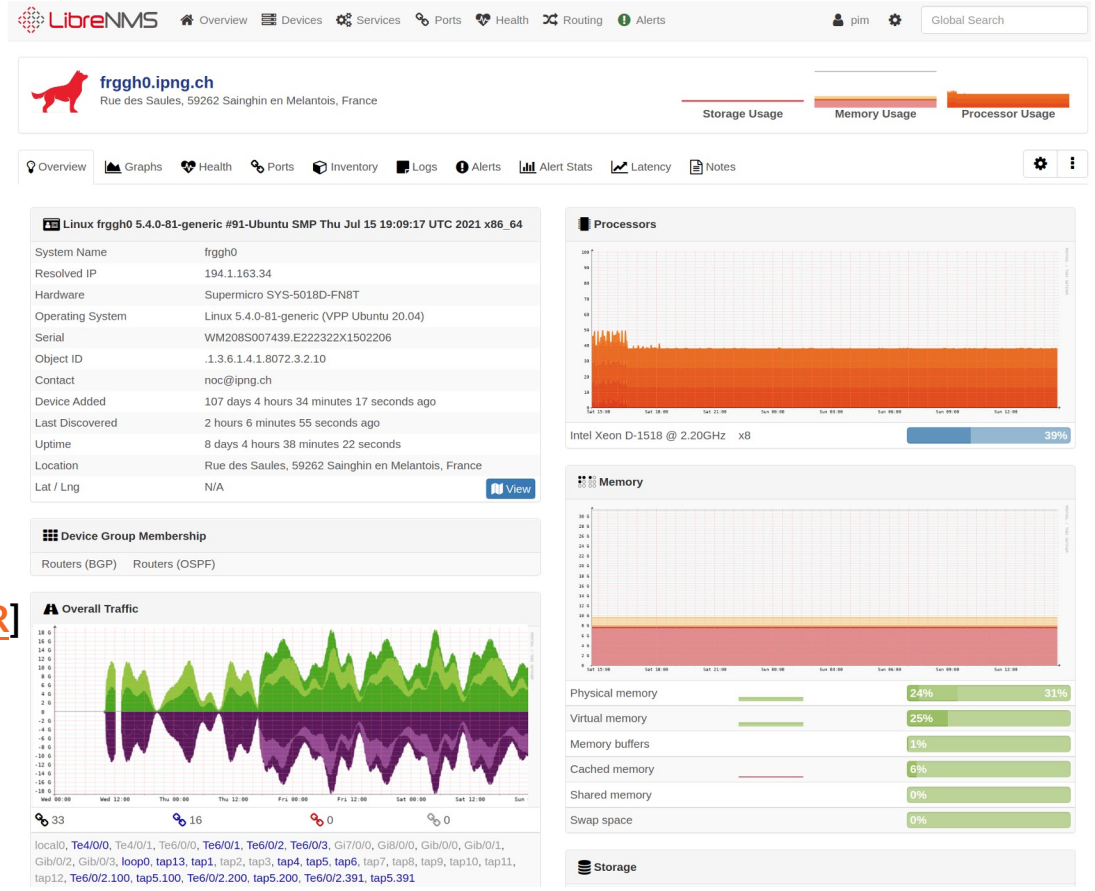
Running VPP in production

1. Deployed 12x Supermicro 5018D-F8NT
   a. 6x 10G (Intel X710 / X552)
   b. 6x 1G (Intel i350 / i210)
   c. Dedicated IPMI for OOB via AS25091
   d. Throughput ~37.8Mpps at ~45Watt
2. Interconnect 10G EoMPLS: Network Map
3. Uplink IXP and IP Transit 10G: Peering DB
4. Monitor with SNMP: Wrote vpp-snmp-agent
5. Dataplane Management: Wrote vppcfg
6. Controlplane Management: Wrote Kees

Adding SNMP to VPP

github.com/pimvanpelt/vpp-snmp-agent

1. Wrote the SNMP Agent
2. Added *logo* to LibreNMS [PR]
3. Added *distro* to LibreNMS Agent [PR]

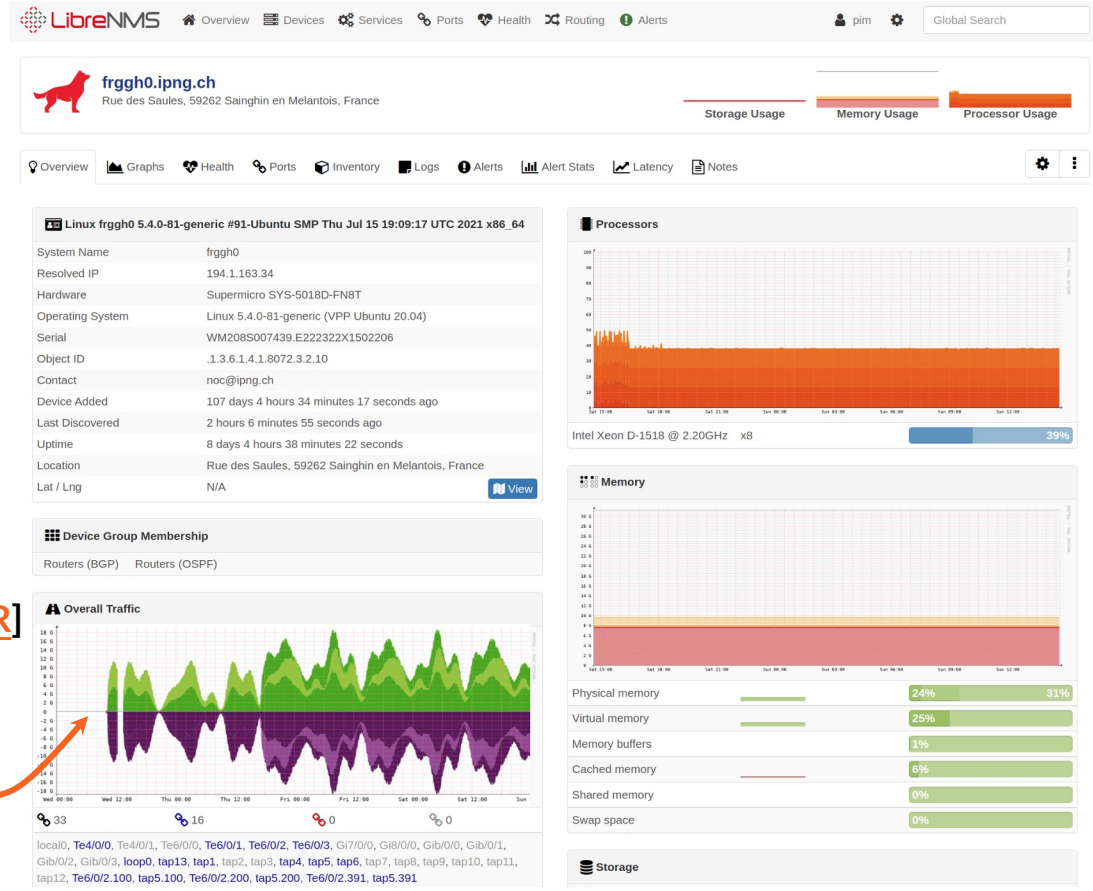


Adding SNMP to VPP

github.com/pimvanpelt/vpp-snmp-agent

1. Wrote the SNMP Agent
2. Added *logo* to LibreNMS [PR]
3. Added *distro* to LibreNMS Agent [PR]

Casually forwarding 18Gbps

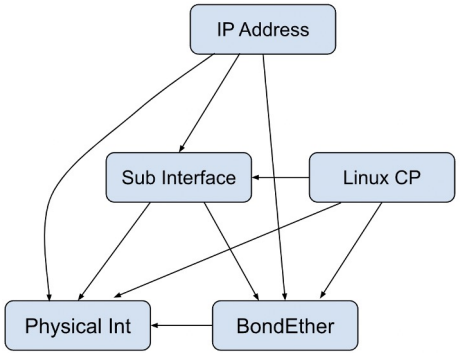


Configuration - Dataplane - vppcfg

Wrote github.com/pimvanpelt/vppcfg

- Reads a YAML configuration file [user guide] [config guide]
  - Checks it for syntax using Yamale
  - Checks it for semantics using a constraints language
- Dumps running state into a YAML file, using VPP API
- Plans a path from the running state to the required state
  - Uses declarative sequencing with a DAG
- Applies any new configuration to VPP using API or CLI

Targeting inclusion upstream in VPP 23.02



Configuration - Dataplane - vppcfg

```
pim@chplo0:~$ vppcfg dump -o chplo0.yaml
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.vppapi.write: Wrote YAML config to chplo0.yaml

pim@chplo0:~$ cp chplo0.yaml chplo0-new.yaml
pim@chplo0:~$ vim chplo0-new.yaml
```

Configuration - Dataplane - vppcfg

```
pim@chplo0:~$ vppcfg dump -o chplo0.yaml
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.vppapi.write: Wrote YAML config to chplo0.yaml


…
interfaces:
  GigabitEthernetb/0/1:
    description: 'Infra: test interface'
    mtu: 9216
    lcp: e1-1
    sub-interfaces:
      100:
        lcp: e1-1.100
        description: 'Cust: demo customer'
        mtu: 1500
        addresses: [ 192.0.2.1/24, 2001:db8::1/64 ]
      200:
        description: 'Cust: demo L2 cross connect'
        mtu: 9000
        l2xc: GigabitEthernetb/0/2
  GigabitEthernetb/0/2:
    description: ''
    mtu: 9000
    l2xc: GigabitEthernetb/0/1.200
…
```

Configuration - Dataplane - vppcfg

```
pim@chplo0:~$ vppcfg dump -o chplo0.yaml
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.vppapi.write: Wrote YAML config to chplo0.yaml

pim@chplo0:~$ cp chplo0.yaml chplo0-new.yaml
pim@chplo0:~$ vim chplo0-new.yaml


pim@chplo0:~$ vppcfg plan -c chplo0-new.yaml -o vpp.exec
[INFO    ] root.main: Loading configfile chplo0-new.yaml
[INFO    ] vppcfg.config.valid_config: Configuration validated successfully
[INFO    ] root.main: Configuration is valid
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.reconciler.write: Wrote 22 lines to vpp.exec
[INFO    ] root.main: Planning succeeded
```

Configuration - Dataplane - vppcfg

```
pim@chplo0:~$ vppcfg dump -o chplo0.yaml
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.vppapi.write: Wrote YAML config to chplo0.yaml


comment { vppcfg prune: 1 CLI statement(s) follow }
lcp delete GigabitEthernetb/0/2
comment { vppcfg create: 4 CLI statement(s) follow }
create sub GigabitEthernetb/0/1 100 dot1q 100 exact-match
create sub GigabitEthernetb/0/1 200 dot1q 200 exact-match
lcp create GigabitEthernetb/0/1 host-if e1-1
lcp create GigabitEthernetb/0/1.100 host-if e1-1.100
comment { vppcfg sync: 14 CLI statement(s) follow }
set interface l2 xconnect GigabitEthernetb/0/1.200 GigabitEthernetb/0/2
set interface l2 tag-rewrite GigabitEthernetb/0/1.200 pop 1
set interface l2 xconnect GigabitEthernetb/0/2 GigabitEthernetb/0/1.200
set interface l2 tag-rewrite GigabitEthernetb/0/2 disable
set interface mtu 9216 GigabitEthernetb/0/1
set interface mtu packet 9216 GigabitEthernetb/0/1
set interface mtu packet 1500 GigabitEthernetb/0/1.100
set interface mtu packet 9000 GigabitEthernetb/0/1.200
set interface ip address GigabitEthernetb/0/1.100 192.0.2.1/24
set interface ip address GigabitEthernetb/0/1.100 2001:db8::1/64
set interface state GigabitEthernetb/0/1 up
set interface state GigabitEthernetb/0/1.100 up
set interface state GigabitEthernetb/0/1.200 up
set interface state GigabitEthernetb/0/2 up
```

Configuration - Dataplane - vppcfg

```
pim@chplo0:~$ vppcfg dump -o chplo0.yaml
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.vppapi.write: Wrote YAML config to chplo0.yaml

pim@chplo0:~$ cp chplo0.yaml chplo0-new.yaml
pim@chplo0:~$ vim chplo0-new.yaml


pim@chplo0:~$ vppcfg plan -c chplo0-new.yaml -o vpp.exec
[INFO    ] root.main: Loading configfile chplo0-new.yaml
[INFO    ] vppcfg.config.valid_config: Configuration validated successfully
[INFO    ] root.main: Configuration is valid
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.reconciler.write: Wrote 22 lines to vpp.exec
[INFO    ] root.main: Planning succeeded


pim@chplo0:~$ vppcfg apply -c chplo0-new.yaml
[INFO    ] root.main: Loading configfile chplo0-new.yaml
[INFO    ] vppcfg.config.valid_config: Configuration validated successfully
[INFO    ] root.main: Configuration is valid
[INFO    ] vppcfg.vppapi.connect: VPP version is 22.10-rc0~33-gd6c3b1f1f
[INFO    ] vppcfg.reconciler.plan: Path planning complete
[INFO    ] vppcfg.applier.write: Will send 19 API commands to VPP
[INFO    ] root.main: Applying succeeded
```

Configuration - Controlplane - Kees

Wrote git.ipng.ch/ipng/kees that:

- Reads set of YAML configuration file
  - Augments with PeeringDB and IRRDBs (bgpq4)
- Constructs per-router configuration
- Uses Jinja2 (Ansible) to emit configuration files
- Rsync to router, safe reloads of controlplane
  - Unbound, Firewall, Bird2, Borgmatic, SNMP Agent, VRRP
  - And of course: VPP configs


Public Peering Exchange Points

Filter

Exchange IPv4	ASN IPv6	Speed	RS Peer
CHIX-CH 185.1.59.150	8298 2001:7f8:cc:333::150	10G	✓
Community-IX.ch 185.1.105.16	8298 2001:7f8:bf:1::10	10G	✓
DE-CIX Dusseldorf 185.1.171.43	8298 2001:7f8:9e::206a:0:1	1G	✓
DE-CIX Frankfurt 80.81.197.38	8298 2001:7f8::206a:0:1	100M	✓
DE-CIX Hamburg 185.1.210.235	8298 2001:7f8:3d::206a:0:1	1G	✓
DE-CIX Munich 185.1.208.84	8298 2001:7f8:44::206a:0:1	1G	✓
EVIX 206.81.104.24	8298 2602:fed2:fff:ffff::24	1G	✓
FCIX 206.80.238.92	8298 2001:504:91::92	10G	✓
FogIXP 185.1.147.43	8298 2001:7f8:ca:1::43	10G	✓
FogIXP ⚠ 185.1.147.44	8298 2001:7f8:ca:1::44	1G	○
France-IX Marseille 37.49.232.119	8298 2001:7f8:54:5::119	200M	✓
France-IX Paris	8298	200M	✓

***) Originally written by Coloclue AS8283**

Configuration - Controlplane - Kees


```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml
```

Configuration - Controlplane - Kees


```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml

ebgp:
  groups:
    decix_ham:
      bgp_local_pref: 200
      peeringdb_ix: 74
      ixp_community: 1070
      sessions:
        43252:
          description: DE-CIX Hamburg Routeserver
          ixp_community: 1071
        6939: {}
        39572: {}
        16509: {}
```

Configuration - Controlplane - Kees


```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml
pim@spongebob:~/src/ipng-kees$ vim config/defra0.ipng.ch.yaml
```

Configuration - Controlplane - Kees

```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml
pim@spongebob:~/src/ipng-kees$ vim config/defra0.ipng.ch.yaml

ebgp:
  groups:
    decix_ham:
      local-addresses: [ 185.1.210.235/23, 2001:7f8:3d::206a:0:1/64 ]
```

Configuration - Controlplane - Kees

```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml
pim@spongebob:~/src/ipng-kees$ vim config/defra0.ipng.ch.yaml
pim@spongebob:~/src/ipng-kees$ ROUTERS=defra0.ipng.ch kees-build
```

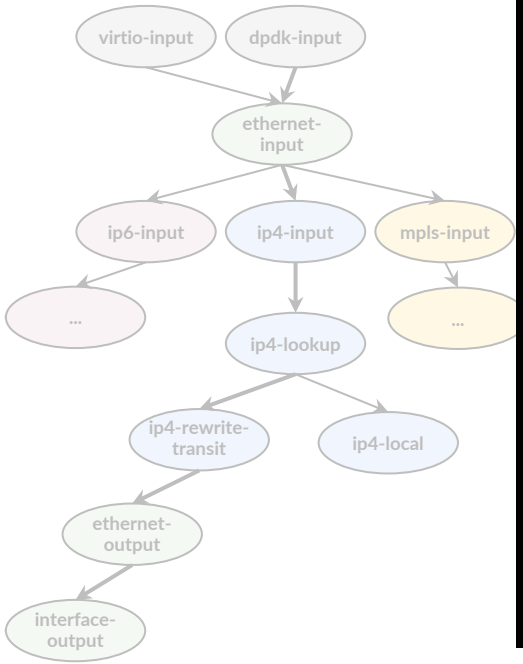
Configuration - Controlplane - Kees


```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml
pim@spongebob:~/src/ipng-kees$ vim config/defra0.ipng.ch.yaml
pim@spongebob:~/src/ipng-kees$ ROUTERS=defra0.ipng.ch kees-build

[INFO] generate - main           : Generating host defra0.ipng.ch
…
[INFO] generate.pdb - fetch          : Fetching https://peeringdb.com/api/ixlan?id=74&depth=1
[INFO] generate.pdb - fetch          : Fetching https://peeringdb.com/api/net?asn__in=6939,16509,39572,43252
[INFO] generate.pdb - fetch          : Fetching https://peeringdb.com/api/netixlan?ixlan_id=74&asn__in=6939,16509,39572,43252
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.210.252 asn 43252
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.211.252 asn 43252
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::a8f4:0:1 asn 43252
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::a8f4:0:2 asn 43252
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.210.55 asn 6939
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::1b1b:0:1 asn 6939
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.210.33 asn 39572
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::9a94:0:1 asn 39572
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.210.128 asn 16509
[INFO]     generate - ebgp_generate  : Rendering decix_ham 185.1.210.235 <-> 185.1.210.53 asn 16509
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::407d:0:1 asn 16509
[INFO]     generate - ebgp_generate  : Rendering decix_ham 2001:7f8:3d::206a:0:1 <-> 2001:7f8:3d::407d:0:2 asn 16509
…
[INFO]     generate - emit           : Emitting build/defra0.ipng.ch/etc/bird/ebgp/groups/decix_ham.conf
[INFO]     generate - prune          : Pruning file etc/bird/manual.conf (build/defra0.ipng.ch/etc/bird/manual.conf)
[INFO]     generate - prune          : Pruning file etc/vpp/config/manual.vpp (build/defra0.ipng.ch/etc/vpp/config/manual.vpp)

Testing build/defra0.ipng.ch/etc/bird/bird.conf - OK!
```

Configuration - Controlplane - Kees

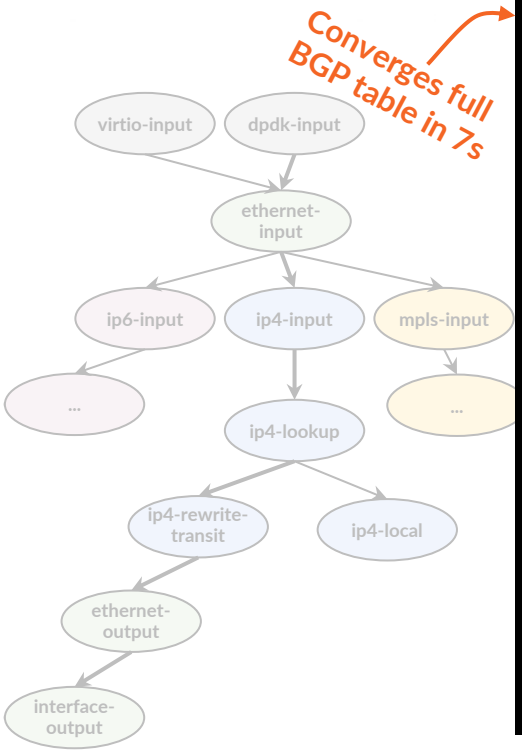
```
pim@spongebob:~/src/ipng-kees$ vim config/common/ebgp-decix.yaml

pim@spongebob:~/src/ipng-kees$ vim config/defra0.ipng.ch.yaml

pim@spongebob:~/src/ipng-kees$ ROUTERS=defra0.ipng.ch kees-build

pim@spongebob:~/src/ipng-kees$ kees-push defra0.ipng.ch
Rsyncing config to defra0.ipng.ch
Setting permissions on defra0.ipng.ch
Reloading bird on defra0.ipng.ch
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured

pim@spongebob:~/src/ipng-kees$ git commit -m “Add DE-CIX Hamburg”
```

VPP in Production: tying it all together



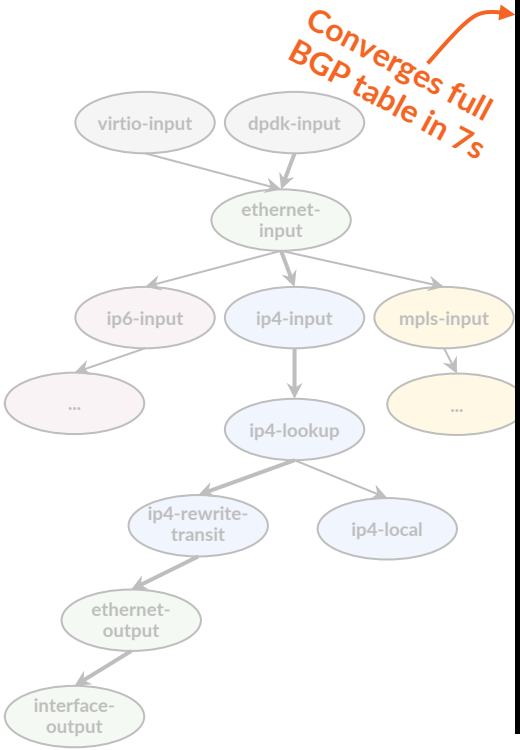

```
pim@defra0:~$ birdc show route count
BIRD 2.0.7 ready.
8195774 of 8195774 routes for 930541 networks in table master4
1585721 of 1585721 routes for 161818 networks in table master6
1263982 of 1263982 routes for 315996 networks in table t_roa4
313125 of 313125 routes for 78284 networks in table t_roa6
Total: 11358602 of 11358602 routes for 1486639 networks in 4 tables



pim@defra0:~$ birdc show ospf neighbor ospf6
BIRD 2.0.7 ready.
Router ID    Pri      State    DTime  Interface  Router IP
194.1.163.0    1   Full/PtP   38.922  xe1-0      fe80::9e69:b4ff:fe61:7682
194.1.163.32   1   Full/PtP   32.976  xe1-1      fe80::6a05:caff:fe32:3cdb
194.1.163.140  1   Full/DR    37.944  xe1-2.2006 fe80::5054:ff:feb0:442c
```

VPP in Production: tying it all together




```
pim@defra0:~$ birdc show route count
BIRD 2.0.7 ready.
8195774 of 8195774 routes for 930541 networks in table master4
1585721 of 1585721 routes for 161818 networks in table master6
1263982 of 1263982 routes for 315996 networks in table t_roa4
313125 of 313125 routes for 78284 networks in table t_roa6
Total: 11358602 of 11358602 routes for 1486639 networks in 4 tables



pim@defra0:~$ birdc show ospf neighbor ospf6
BIRD 2.0.7 ready.
Router ID    Pri      State    DTime  Interface  Router IP
194.1.163.0    1   Full/PtP   38.922  xe1-0      fe80::9e69:b4ff:fe61:7682
194.1.163.32   1   Full/PtP   32.976  xe1-1      fe80::6a05:caff:fe32:3cdb
194.1.163.140  1   Full/DR    37.944  xe1-2.2006 fe80::5054:ff:feb0:442c
```

VPP in Production: tying it all together




```
pim@defra0:~$ birdc show route count
BIRD 2.0.7 ready.
8195774 of 8195774 routes for 930541 networks in table master4
1585721 of 1585721 routes for 161818 networks in table master6
1263982 of 1263982 routes for 315996 networks in table t_roa4
313125 of 313125 routes for 78284 networks in table t_roa6
Total: 11358602 of 11358602 routes for 1486639 networks in 4 tables

pim@spongebob:~$ traceroute gripe.ipng.nl
traceroute to gripe.ipng.nl (94.142.241.186), 64 hops max, 40 byte packets
 1  chbtl0.ipng.ch (194.1.163.66)  0.291 ms  0.138 ms  0.105 ms
 2  chrma0.ipng.ch (194.1.163.17)  0.979 ms  1.068 ms  1.142 ms
 3  defra0.ipng.ch (194.1.163.25)  6.581 ms  6.573 ms  6.629 ms
 4  nlams0.ipng.ch (194.1.163.27)  12.785 ms  12.911 ms  12.838 ms
 5  gripe.ipng.nl (94.142.241.186)  13.316 ms  13.289 ms  13.212 ms
```

There is another ...

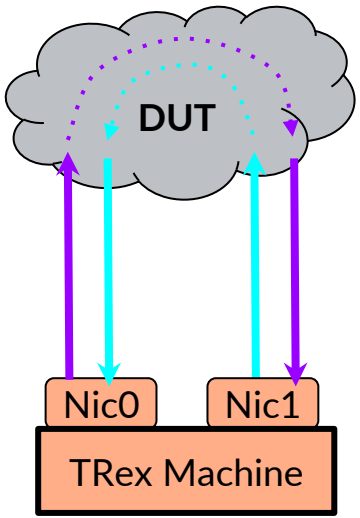
There is another ...

... super-useful DPDK app

Cisco T-Rex traffic load tester [link]

- Stateless/Stateful load testing
- Python API - fully programmable
  - Traffic streams w/ scapy [api]
  - Traffic ramp up/down [api]
  - Runtime statistics [api]
  - Interactive / CLI [docs]
- DPDK Integration
  - ~10-15Mpps per core
  - Linear scaling with cores
  - 1G/10G/25G/40G/100G

Config and Startup



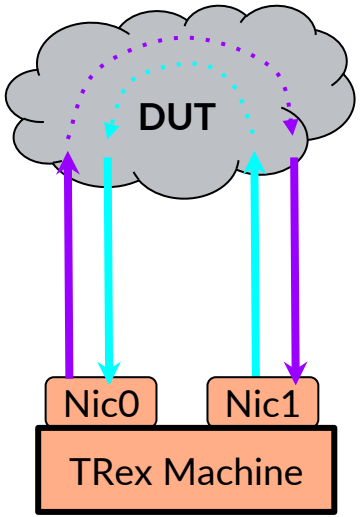
Simple configuration:

```
- version: 2
  interfaces: ['5:00.0', '5:00.1']
  port_info:
      - src_mac    : 0A:0B:0C:01:02:AA # Mac A
        dest_mac   : 0A:0B:0C:01:02:BB # Mac B
      - src_mac    : 0A:0B:0C:01:02:BB # Mac B
        dest_mac   : 0A:0B:0C:01:02:AA # Mac A
```

Startup:

```
$ sudo ./t-rex-64 -i -c 4
$ ./trex-console
```

Config and Startup



Simple configuration:

```
- version: 2
  interfaces: ['5:00.0', '5:00.1']
  port_info:
      - ip         : 100.65.1.2
        default_gw: 100.65.1.1
      - ip         : 100.65.2.2
        default_gw: 100.65.2.1
```

Startup:

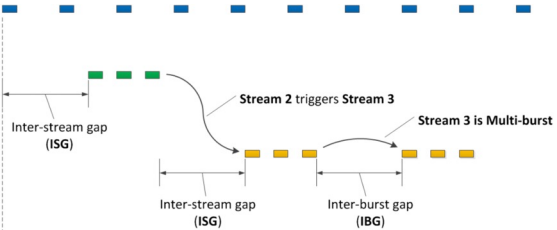
```
$ sudo ./t-rex-64 -i -c 4
$ ./trex-console
```

Stateless Traffic Profiles

Assemble packet streams with scapy:

- IPv4/IPv6 src/dst; proto; port src/dst; size; ratios; timings

```
self.ip_range = {'src': {'start': "16.0.0.1", 'end': "16.0.0.254"},
                 'dst': {'start': "48.0.0.1",  'end': "48.0.0.254"}}
```




```
# default IMIX properties
self.imix_table = [ {'size': 60,   'pps': 28,  'isg':0 },
                    {'size': 590,  'pps': 16,  'isg':0.1 },
                    {'size': 1514, 'pps': 4,   'isg':0.2 } ]
```

- Streams are *applied* on one or more ports
- Ports are configured to send a *rate* of traffic (bps, pps or % of line)

Load Testing Methodology

Method 1: VPP has one worker thread, one Rx/Tx queue

- Send *unidirectional* traffic
- Measure cycles/packet for 1kpps, 1Mpps, 10Mpps, ...
  ⇒ Report max packets/sec for one CPU thread

Method 2: VPP has n-1 worker threads with [1, 2, 3, ...] Rx queues

- Send *unidirectional*, or *bidirectional* (!) traffic
  - Warmup at 1kpps (30sec)
  - Ramp up to 100% line rate (in 600sec)
  - Keep at 100% (30sec)
- Measure point at which packet forwarding loss > 0.1%
  ⇒ Report bits/sec, packets/sec and % of line rate.

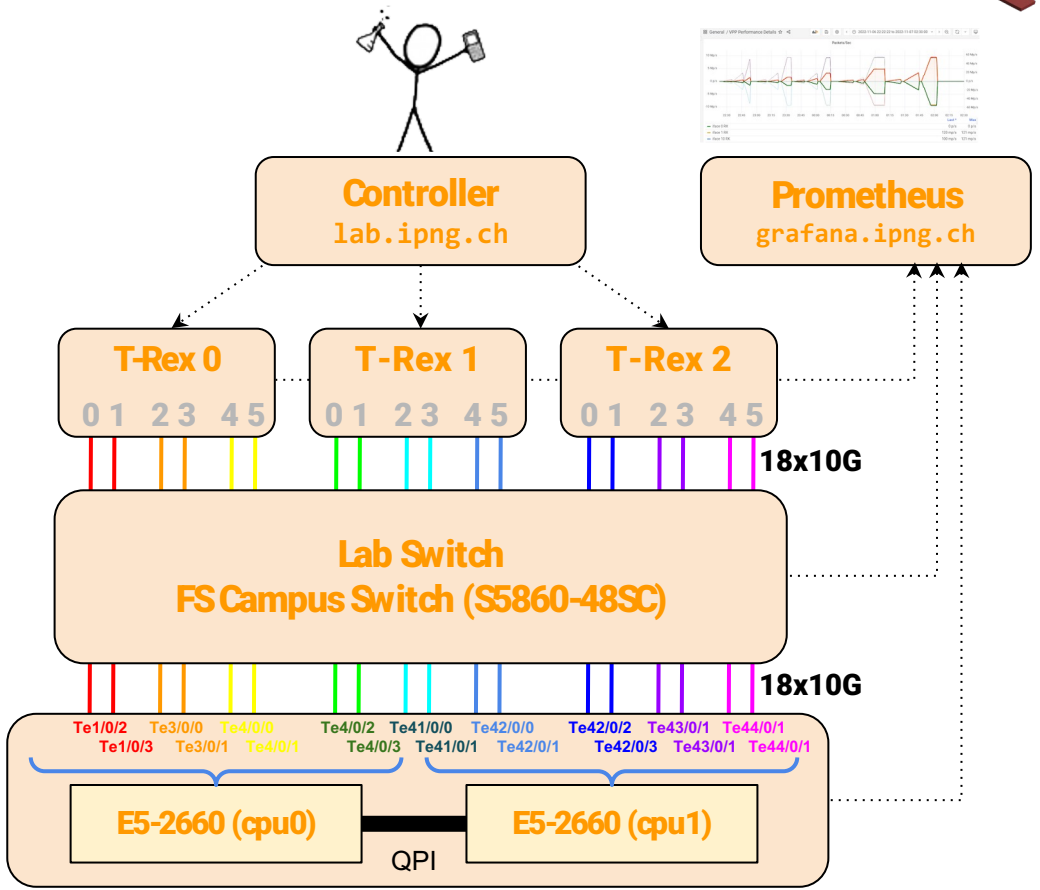
Lab Setup

Load Generator Machines

- 3x Dell R720 (E5-2620, 2.00GHz)
- 9x Dual Intel 82599ES (18x10G)
- Debian Bullseye, T-Rex v3.00
  - 4 CPUs per 10G interface pair

Device Under Test

- Dell R720XD (E5-2660 v2 @ 2.20GHz)
  - 80x PCIe v3.0 Lanes
  - 4x16GB DDR3 1866MT/s
- 3x Intel X710 + 4x 82599ES (20x10G)
  - 10x10G on cpu0, 10x10G on cpu1
- Bullseye (Kernel 5.10.0, isolcpus=2-63)
- VPP v23.02-rc0~22-g6161bba1e



Method 1 - Single Thread Saturation

Legend

1. NIC Info, T-Rex CPU utilization
2. Sent traffic (L1, L2, packets/sec)
3. Received traffic (L2, packets/sec)
4. Detailed packet/byte counters

Global Statistics

```
connection   : hvn0, Port 4501                    total_tx_L2  : 30.54 Gbps
version      : STL @ v3.00                        total_tx_L1  : 40.08 Gbps
cpu_util.    : 37.52% @ 18 cores (6 per dual port) total_rx     : 27.46 Gbps
rx_cpu_util. : 0.0% / 0 pps                       total_pps    : 59.65 Mpps
async_util.  : 0% / 115.04 bps                    drop_rate    : 3.08 Gbps
total_cps.   : 0 cps                              queue_full   : 1,552,701,642 pkts
```

Port Statistics

port	0	1	2	3	total
owner	pim	pim	pim	pim	
link	UP	UP	UP	UP	
state	TRANSMITTING	TRANSMITTING	TRANSMITTING	TRANSMITTING	
speed	10 Gb/s	10 Gb/s	10 Gb/s	10 Gb/s	
CPU util.	34.46%	34.46%	78.11%	78.11%	
--					
Tx bps L2	7.63 Gbps	7.63 Gbps	7.64 Gbps	7.64 Gbps	30.54 Gbps
Tx bps L1	10.01 Gbps	10.01 Gbps	10.03 Gbps	10.03 Gbps	40.08 Gbps
Tx pps	14.89 Mpps	14.89 Mpps	14.93 Mpps	14.93 Mpps	59.65 Mpps
Line Util.	100.08 %	100.08 %	100.33 %	100.33 %	

Rx bps	6.85 Gbps	6.92 Gbps	6.84 Gbps	6.85 Gbps	27.46 Gbps
Rx pps	13.38 Mpps	13.51 Mpps	13.35 Mpps	13.39 Mpps	53.63 Mpps
----					
opackets	12593800641	11197749872	11197738324	11197763846	46187052683
ipackets	10060226827	11424580128	10009980733	10036657669	41531445357
obytes	806003241216	716655992256	716655253248	716656886656	2955971373376
ibytes	643854517184	731173128384	640638767360	642346091008	2658012503936
tx-pkts	12.59 Gpkts	11.2 Gpkts	11.2 Gpkts	11.2 Gpkts	46.19 Gpkts
rx-pkts	10.06 Gpkts	11.42 Gpkts	10.01 Gpkts	10.04 Gpkts	41.53 Gpkts
tx-bytes	806 GB	716.66 GB	716.66 GB	716.66 GB	2.96 TB
rx-bytes	643.85 GB	731.17 GB	640.64 GB	642.35 GB	2.66 TB
-----					
oerrors	0	0	0	0	0
ierrors	0	0	0	0	0

Method 1 - Single Thread Saturation

Legend

1. NIC Info, T-Rex CPU utilization
2. Sent traffic (L1, L2, packets/sec)
3. Received traffic (L2, packets/sec)
4. Detailed packet/byte counters

Shown here

- Tx: 59.65Mpps, 40.00Gbps
- Rx: 53.63Mpps, 35.97Gbps

⇒ L2 XC is 13.4Mpps per core!

Global Statistics

```
connection   : hvn0, Port 4501                  total_tx_L2  : 30.54 Gbps
version      : STL @ v3.00                      total_tx_L1  : 40.08 Gbps
cpu_util.    : 37.52% @ 18 cores (6 per dual port)  total_rx  : 27.46 Gbps
rx_cpu_util. : 0.0% / 0 pps                     total_pps    : 59.65 Mpps
async_util.  : 0% / 115.04 bps                  drop_rate    : 3.08 Gbps
total_cps.   : 0 cps                            queue_full   : 1,552,701,642 pkts
```

Port Statistics

port	0	1	2	3	total
owner	pim	pim	pim	pim	
link	UP	UP	UP	UP	
state	TRANSMITTING	TRANSMITTING	TRANSMITTING	TRANSMITTING	
speed	10 Gb/s	10 Gb/s	10 Gb/s	10 Gb/s	
CPU util.	34.46%	34.46%	78.11%	78.11%	
--					
Tx bps L2	7.63 Gbps	7.63 Gbps	7.64 Gbps	7.64 Gbps	30.54 Gbps
Tx bps L1	10.01 Gbps	10.01 Gbps	10.03 Gbps	10.03 Gbps	40.08 Gbps
Tx pps	14.89 Mpps	14.89 Mpps	14.93 Mpps	14.93 Mpps	59.65 Mpps
Line Util.	100.08 %	100.08 %	100.33 %	100.33 %	

Rx bps	6.85 Gbps	6.92 Gbps	6.84 Gbps	6.85 Gbps	27.46 Gbps
Rx pps	13.38 Mpps	13.51 Mpps	13.35 Mpps	13.39 Mpps	53.63 Mpps
----					
opackets	12593800641	11197749872	11197738324	11197763846	46187052683
ipackets	10060226827	11424580128	10009980733	10036657669	41531445357
obytes	806003241216	716655992256	716655253248	716656886656	2955971373376
ibytes	643854517184	731173128384	640638767360	642346091008	2658012503936
tx-pkts	12.59 Gpkts	11.2 Gpkts	11.2 Gpkts	11.2 Gpkts	46.19 Gpkts
rx-pkts	10.06 Gpkts	11.42 Gpkts	10.01 Gpkts	10.04 Gpkts	41.53 Gpkts
tx-bytes	806 GB	716.66 GB	716.66 GB	716.66 GB	2.96 TB
rx-bytes	643.85 GB	731.17 GB	640.64 GB	642.35 GB	2.66 TB
-----					
oerrors	0	0	0	0	0
ierrors	0	0	0	0	0

Method 1: Results (E5 2660 v2)

	clocks/vector @ 1kpps	clocks/vector @ 1Mpps	clocks/vector @ 10Mpps	vectors/sec per core
L2 xconnect	1765	293	173	13.4Mpps
L3 IPv4	2146	366	201	9.31Mpps
L3 IPv6	2744	381	221	8.48Mpps

- CPU cycles/packet: lower is better
- Max PPS per core: higher is better

Method 2 - T-Rex Python API

```
usage: trex-loadtest.py [-h] [-s SERVER] [-p PROFILE_FILE] [-o OUTPUT_FILE] [-wm WARMUP_MULT]
                        [-wd WARMUP_DURATION] [-rt RAMPUP_TARGE] [-rd RAMPUP_DURATION] [-hd HOLD_DURATION]
T-Rex Stateless Loadtester -- pim@ipng.nl
optional arguments:
  -h, --help            show this help message and exit
  -s SERVER, --server SERVER
                        Remote trex address (default: 127.0.0.1)
  -p PROFILE_FILE, --profile PROFILE_FILE
                        STL profile file to replay (default: imix.py)
  -o OUTPUT_FILE, --output OUTPUT_FILE
                        File to write results into, use "-" for stdout (default: -)
  -wm WARMUP_MULT, --warmup_mult WARMUP_MULT
                        During warmup, send this "mult" (default: 1kpps)
  -wd WARMUP_DURATION, --warmup_duration WARMUP_DURATION
                        Duration of warmup, in seconds (default: 30)
  -rt RAMPUP_TARGET, --rampup_target RAMPUP_TARGET
                        Target percentage of line rate to ramp up to (default: 100)
  -rd RAMPUP_DURATION, --rampup_duration RAMPUP_DURATION
                        Time to take to ramp up to target percentage of line rate, in seconds (default: 600)
  -hd HOLD_DURATION, --hold_duration HOLD_DURATION
                        Time to hold the loadtest at target percentage, in seconds (default: 30)
```

Method 2 - Interesting profiles

From easier to more challenging (*):

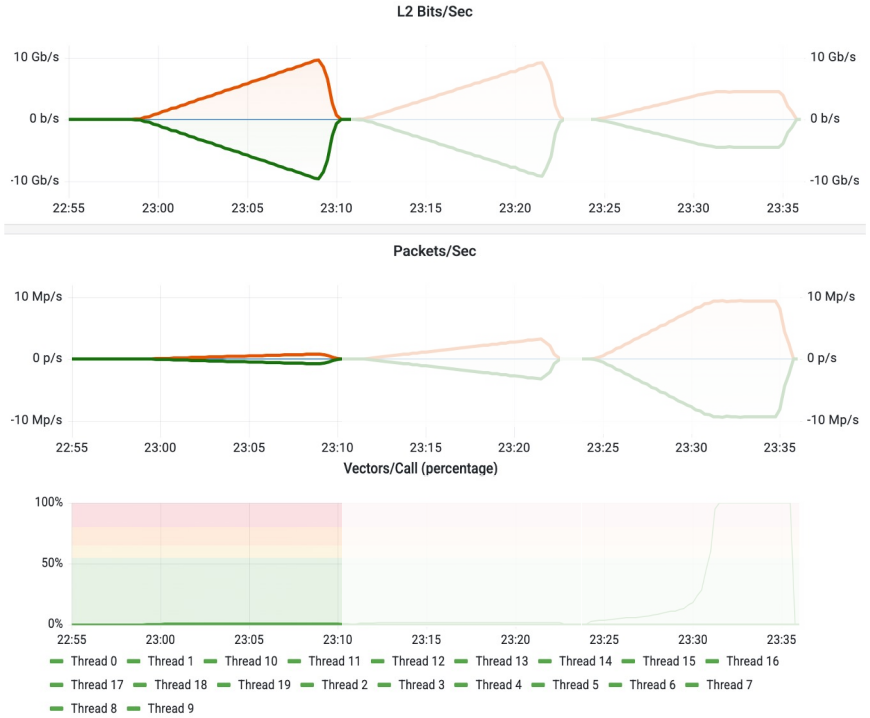
1. `bench-var2-1514b`: 1514b UDP, multiple flows (random src/dst IP)
   ⇒ 810Kpps at 10Gbps
2. `bench-var2-imix`: Mix of 60, 590, 1514 byte UDP, multiple flows
   ⇒ 3.2Mpps at 10Gbps
3. `bench-var2-64b`: 64 byte UDP, multiple flows
   ⇒ 14.88Mpps at 10Gbps, RSS with multiple Rx queues
4. `bench`: 64 byte UDP, single flow (constant src/dst IP:port)
   ⇒ 14.88Mpps at 10Gbps, only one Rx queue (= one *lcore*)

**) numbers are unidirectional*

Method 2: Results 1x 10G

Unidirectional (one port, one VPP thread):

1) 22:57 - 23:08 – 1514b
810kpps, 9.7GbpsL2, 100% line rate

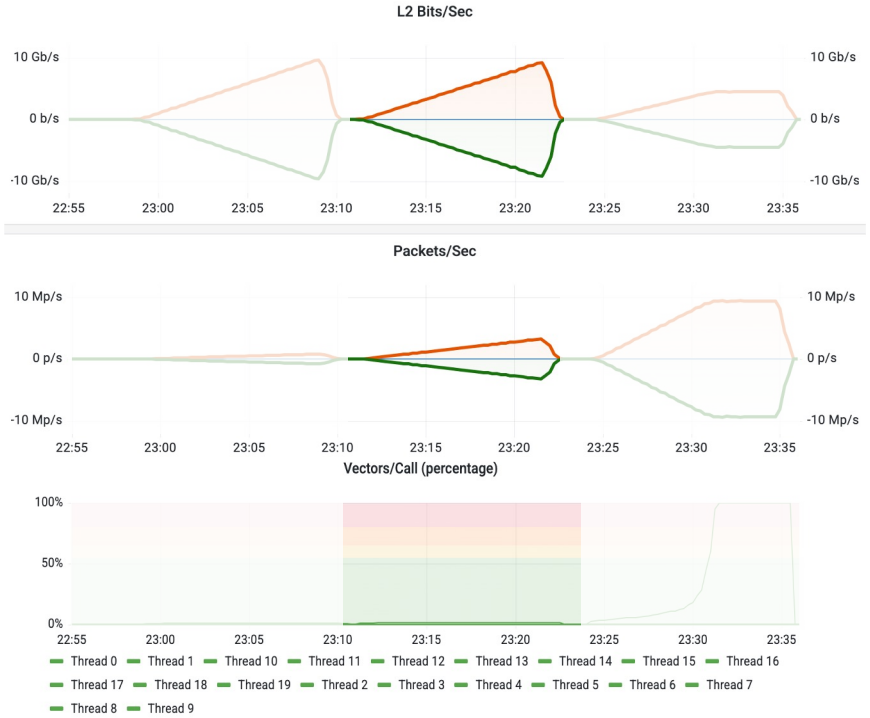


Method 2: Results 1x 10G

Unidirectional (one port, one VPP thread):

1) 22:57 - 23:08 – 1514b
810kpps, 9.7GbpsL2, 100% line rate

2) 23:10 - 23:22 – imix
3.2Mpps, 9.2GbpsL2, 100% line rate



TRex
Realistic Traffic Generator

Method 2: Results 1x 10G

Unidirectional (one port, one VPP thread):

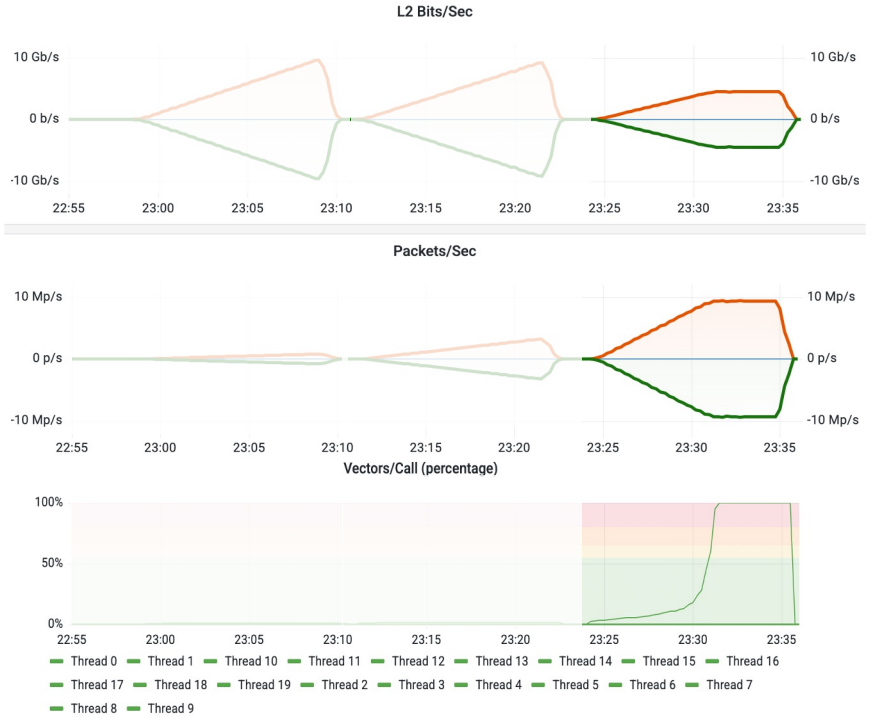
1) 22:57 - 23:08 – 1514b
810kpps, 9.7GbpsL2, 100% line rate

2) 23:10 - 23:22 – imix
3.2Mpps, 9.2GbpsL2, 100% line rate

3) 23:24 - 23:36 – 64b
9.3Mpps, 4.48GbpsL2, 62.5% line rate

At ~8.9Mpps ⇒ overload
vectors/call shoots through the roof

Note: L1 vs L2 bandwidth matters in small packets!



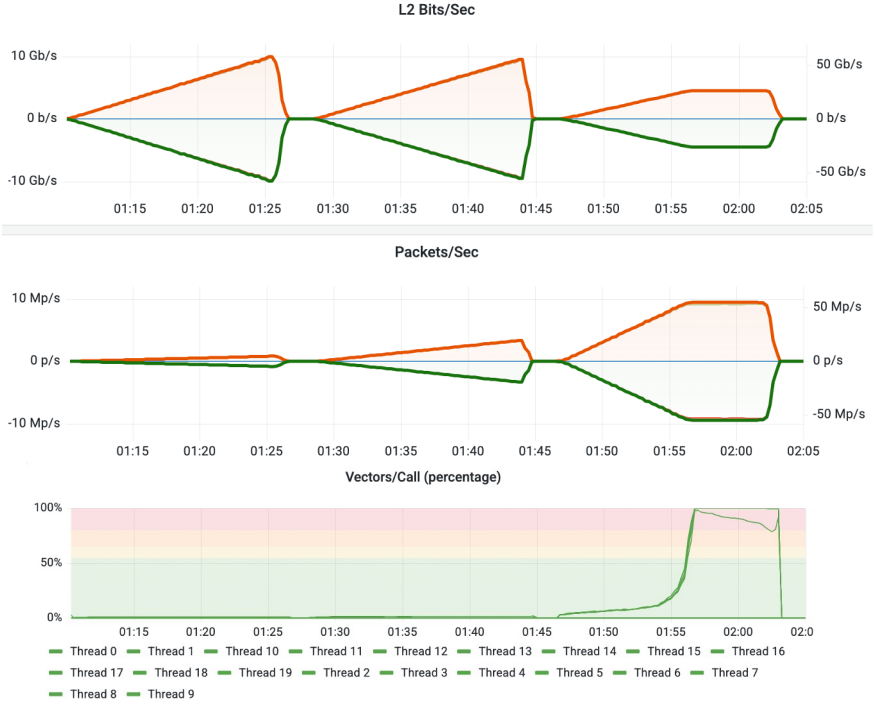
Method 2: Results 6x 10G bidirectional

Bidirectional (six ports, six VPP threads):

1) 01:11 - 01:26 – 1514b

2) 01:29 - 01:44 – imix

3) 01:47 - 02:03 – 64b



- 54.9Mpps total ~ 9.15Mpps per thread
- 58.1GbpsL2 (1514b) 100% line rate
- 55.8GbpsL2 (imix) 100% line rate
- 26.4GbpsL2 (64b) 61.5% line rate

```
vpp v23.02-rc0~22-g6161bba1e built by pim on hippo at 2022-10-04T12:45:06
```

⇒ **Proof that VPP scales linearly**

Claim 1: VPP can forward 100Gbit

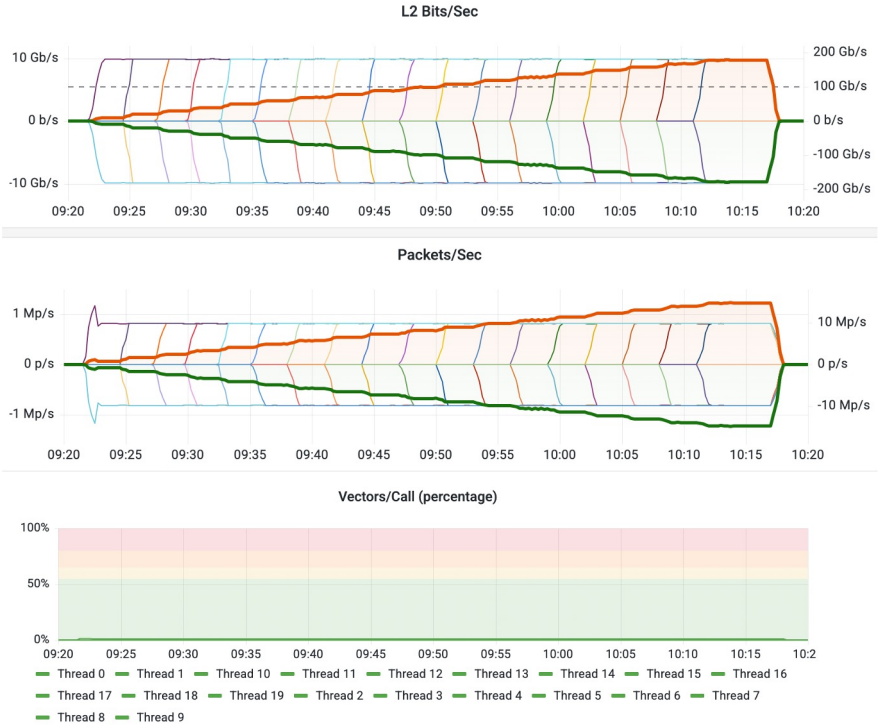
Bidirectional (18 ports, 18 VPP threads):

1) 09:22 - Incremental colored traces
T-Rex ports turned on one by one,
3 minutes apart, sending 1514b

2) 09:50 100Gbit achieved

3) 10:12 180Gbit achieved
14.7Mpps
@1514b

⇒ Proof that VPP (easily) forwards 100Gbit



Claim 2: VPP can forward 100Mpps

Bidirectional (18 ports, 18 VPP threads):

1) 10:26 - Incremental colored traces
T-Rex ports turned on one by one,
3 minutes apart, sending 128b.

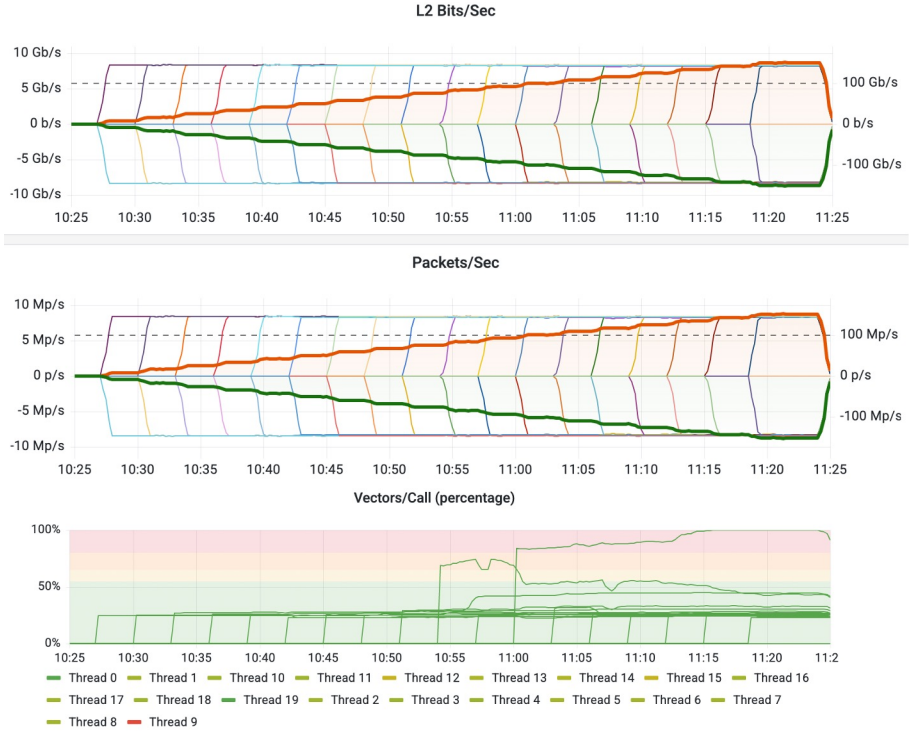
2) 11:02 100Mpps achieved

3) 11:19 151Mpps achieved
149Gbps

@128b

Note: definitely elevated vectors/call, one thread (11:00am) was unhappy due to cross-numa traffic!

⇒ **Proof that VPP (easily) forwards 100Mpps**

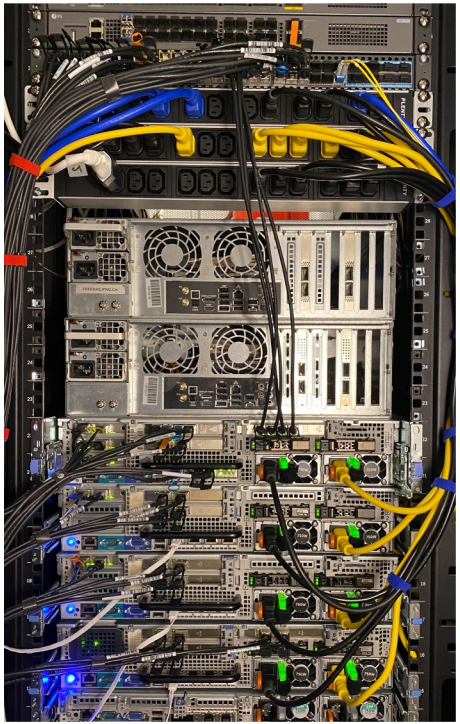


What's next?

Maybe a new claim for 2023:

- PCIE v4.0 and 100G Interfaces
  - Mellanox Cx5/Cx6
  - Intel E810
- 1.0 Tbps and 1.0 Gpps
  "should be possible"
- Looking for sponsors !

What's next?



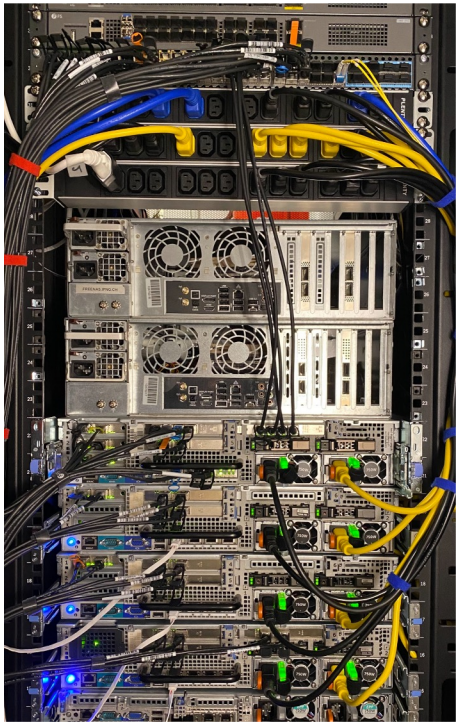
Maybe a new claim for 2023:

- PCIE v4.0 and 100G Interfaces
  - Mellanox Cx5/Cx6
  - Intel E810
- 1.0 Tbps and 1.0 Gpps
  "should be possible"
- Looking for sponsors !

Until then, take a look at these:

- **Our rest rack at AS8298**

What's next?



Maybe a new claim for 2023:

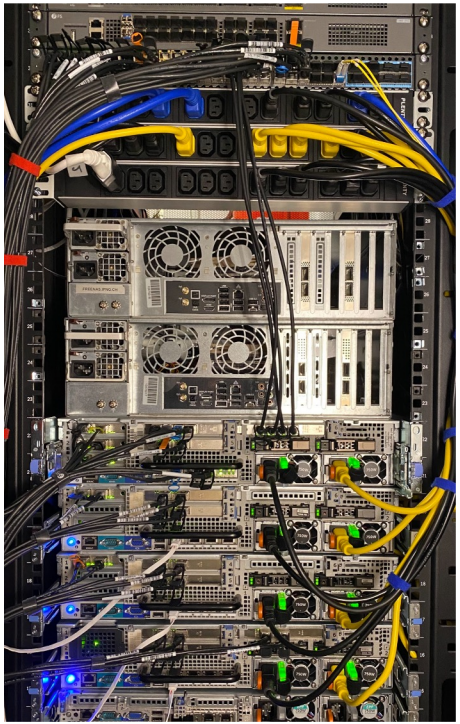
- PCIE v4.0 and 100G Interfaces
  - Mellanox Cx5/Cx6
  - Intel E810
- 1.0 Tbps and 1.0 Gpps
  "should be possible"
- Looking for sponsors !

Until then, take a look at these:

- **Our rest rack at AS8298**
- **FS switch at 36x10G 99.9%**

```
fsw0-lab#show int usage up
Interface                                Bandwidth   Average Usage    Output Usage     Input Usage
---------------------------------------- ----------- ---------------- ---------------- ----------------
TenGigabitEthernet 0/1                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/2                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/3                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/4                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/5                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/6                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/7                   10000  Mbit 99.2147483647%   99.2147483647%   99.2147483647%
TenGigabitEthernet 0/8                   10000  Mbit 99.2133766599%   99.2133827699%   99.2133705499%
TenGigabitEthernet 0/9                   10000  Mbit 99.2114371650%   99.2112639900%   99.2116103399%
TenGigabitEthernet 0/10                  10000  Mbit 99.1090921450%   99.1086479800%   99.1095363099%
TenGigabitEthernet 0/11                  10000  Mbit 99.1129029390%   99.1128538190%   99.1129520599%
TenGigabitEthernet 0/12                  10000  Mbit 99.1125258649%   99.1122409799%   99.1128107499%
TenGigabitEthernet 0/13                  10000  Mbit 99.1099570800%   99.1099414100%   99.1099727499%
TenGigabitEthernet 0/14                  10000  Mbit 99.1093942150%   99.1094080200%   99.1093804099%
TenGigabitEthernet 0/15                  10000  Mbit 99.1098459350%   99.1098057300%   99.1098861400%
TenGigabitEthernet 0/16                  10000  Mbit 99.1145067200%   99.1144910200%   99.1145224200%
TenGigabitEthernet 0/17                  10000  Mbit 99.1152121800%   99.1151935900%   99.1152307699%
TenGigabitEthernet 0/18                  10000  Mbit 99.1142002100%   99.1138415100%   99.1145589099%
TenGigabitEthernet 0/19                  1000   Mbit 0.000005500%     0.000019000%     0.000000000%
TenGigabitEthernet 0/25                  10000  Mbit 99.0941304899%   99.0938793299%   99.0943816500%
TenGigabitEthernet 0/26                  10000  Mbit 99.0930120649%   99.0926209499%   99.0934031799%
TenGigabitEthernet 0/27                  10000  Mbit 99.0989754800%   99.0987229800%   99.0992279799%
TenGigabitEthernet 0/28                  10000  Mbit 99.0480031499%   99.0474431899%   99.0485631100%
TenGigabitEthernet 0/29                  10000  Mbit 99.0471915800%   99.0468672199%   99.0475159399%
TenGigabitEthernet 0/30                  10000  Mbit 99.0474744150%   99.0472400700%   99.0477087600%
TenGigabitEthernet 0/31                  10000  Mbit 99.0468730440%   99.0465942999%   99.0471517800%
TenGigabitEthernet 0/32                  10000  Mbit 99.0467655850%   99.0467423799%   99.0467887899%
TenGigabitEthernet 0/33                  10000  Mbit 99.0380024399%   99.0377253999%   99.0382794799%
TenGigabitEthernet 0/34                  10000  Mbit 99.0422202750%   99.0419492000%   99.0424913500%
TenGigabitEthernet 0/35                  10000  Mbit 99.0419552200%   99.0417516600%   99.0421587799%
TenGigabitEthernet 0/36                  10000  Mbit 99.0417785749%   99.0415835900%   99.0419735600%
TenGigabitEthernet 0/37                  10000  Mbit 99.0402811950%   99.0400948900%   99.0404675000%
TenGigabitEthernet 0/38                  10000  Mbit 99.0397952749%   99.0392615199%   99.0403290299%
TenGigabitEthernet 0/39                  10000  Mbit 99.0136100499%   98.2147483647%   99.0393164799%
TenGigabitEthernet 0/40                  10000  Mbit 98.2147483647%   98.2147483647%   98.2147483647%
TenGigabitEthernet 0/41                  10000  Mbit 98.2147483647%   98.2147483647%   98.2147483647%
TenGigabitEthernet 0/42                  10000  Mbit 98.2147483647%   98.2147483647%   98.2147483647%
TenGigabitEthernet 0/48                  10000  Mbit 0.000003088500%  0.000003000000%  0.000005077000%
fsw0-lab#
```

What's next?



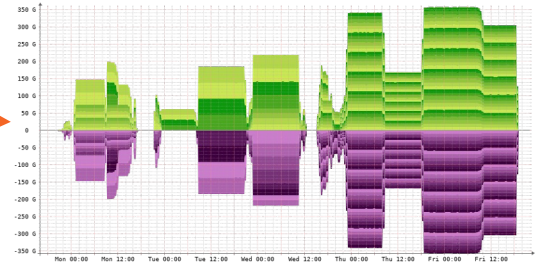
Maybe a new claim for 2023:

- PCIE v4.0 and 100G Interfaces
  - Mellanox Cx5/Cx6
  - Intel E810
- 1.0 Tbps and 1.0 Gpps
  "should be possible"
- Looking for sponsors !

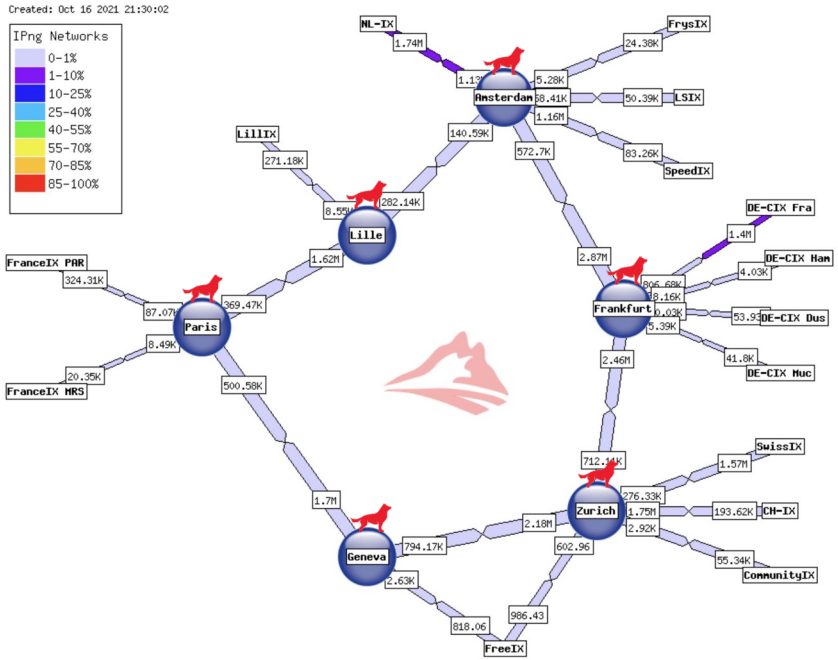
Until then, take a look at these:

- **Our rest rack at AS8298**
- **FS switch at 36x10G 99.9%**
- **LibreNMS confirming 360Gbps**

```
fsw0-lab#show int usage up
Interface                                Bandwidth   Average Usage    Output Usage     Input Usage
---------------------------------------- ----------- ---------------- ---------------- ----------------
TenGigabitEthernet 0/1                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/2                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/3                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/4                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/5                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/6                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/7                   10000  Mbit 99.21474836473%  99.21474836473%  99.21474836473%
TenGigabitEthernet 0/8                   10000  Mbit 99.21337665999%  99.21338276599%  99.21337054999%
TenGigabitEthernet 0/9                   10000  Mbit 99.21143716500%  99.21126399000%  99.21161033999%
TenGigabitEthernet 0/10                  10000  Mbit 99.10909214500%  99.10864798000%  99.10953630999%
TenGigabitEthernet 0/11                  10000  Mbit 99.11290293900%  99.11285381900%  99.11295205900%
TenGigabitEthernet 0/12                  10000  Mbit 99.11252586499%  99.11224007999%  99.11281074999%
TenGigabitEthernet 0/13                  10000  Mbit 99.10955700000%  99.10994141000%  99.10997274999%
TenGigabitEthernet 0/14                  10000  Mbit 99.10939421500%  99.10940802000%  99.10938044099%
TenGigabitEthernet 0/15                  10000  Mbit 99.10984593500%  99.10980573000%  99.10988614000%
TenGigabitEthernet 0/16                  10000  Mbit 99.11450672000%  99.11449102000%  99.11452242000%
TenGigabitEthernet 0/17                  10000  Mbit 99.11521218000%  99.11519359000%  99.11523076999%
TenGigabitEthernet 0/18                  10000  Mbit 99.11420021000%  99.11384151000%  99.11455890999%
TenGigabitEthernet 0/19                  1000   Mbit 0.000005500%     0.000019000%     0.000000000%
TenGigabitEthernet 0/25                  10000  Mbit 99.09413044099%  99.09387932999%  99.09438165000%
TenGigabitEthernet 0/26                  10000  Mbit 99.09301206499%  99.09262094999%  99.09340317999%
TenGigabitEthernet 0/27                  10000  Mbit 99.09897548000%  99.09872298000%  99.09922797999%
TenGigabitEthernet 0/28                  10000  Mbit 99.04800314999%  99.04744318999%  99.04856311000%
TenGigabitEthernet 0/29                  10000  Mbit 99.04719158000%  99.04686721999%  99.04751593999%
TenGigabitEthernet 0/30                  10000  Mbit 99.04747441500%  99.04724007000%  99.04770876000%
TenGigabitEthernet 0/31                  10000  Mbit 99.04687304400%  99.04659429999%  99.04715178000%
TenGigabitEthernet 0/32                  10000  Mbit 99.04676558500%  99.04674237999%  99.04678879999%
TenGigabitEthernet 0/33                  10000  Mbit 99.03800243999%  99.03772539999%  99.03827947999%
TenGigabitEthernet 0/34                  10000  Mbit 99.04222027500%  99.04194920000%  99.04249135000%
TenGigabitEthernet 0/35                  10000  Mbit 99.04195522000%  99.04175166000%  99.04215877999%
TenGigabitEthernet 0/36                  10000  Mbit 99.04177857499%  99.04158359000%  99.04197356000%
TenGigabitEthernet 0/37                  10000  Mbit 99.04028119500%  99.04009489000%  99.04046750000%
TenGigabitEthernet 0/38                  10000  Mbit 99.03979527499%  99.03926151999%  99.04032902999%
TenGigabitEthernet 0/39                  10000  Mbit 99.01361004999%  98.21474836473%  99.03931647999%
TenGigabitEthernet 0/40                  10000  Mbit 98.21474836473%  98.21474836473%  98.21474836473%
TenGigabitEthernet 0/41                  10000  Mbit 98.21474836473%  98.21474836473%  98.21474836473%
TenGigabitEthernet 0/42                  10000  Mbit 98.21474836473%  98.21474836473%  98.21474836473%
TenGigabitEthernet 0/48                  10000  Mbit 0.000003088500%  0.000003000000%  0.000005077000%
fsw0-lab#
```



Questions, Discussion



If you peer with IPng Networks, thanks! If you don't: <peering@ipng.ch>

Support our research: <info@ipng.ch>

Useful Resources

- VPP: fd.io
- VPP Linux CP: Github
- Articles: ipng.ch
- Twitter: @IPngNetworks

Also: thanks for listening!