VPP: IPv4-lite

A 100Mpps+ BGP/OSPF router with a single IPv4 address

Pim van Pelt • 2024–10-25 • ESNOG • Barcelona, Catalonia

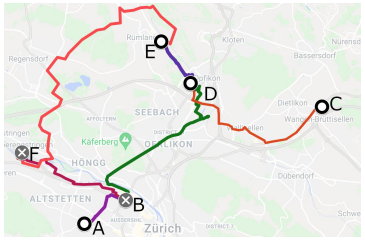
go.ipng.ch/esnog32

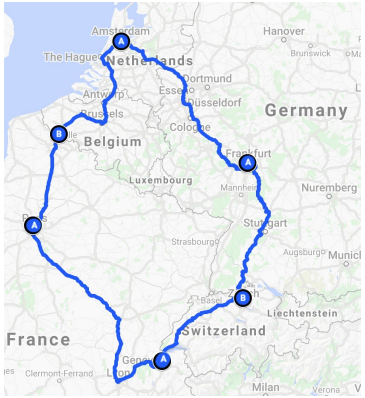
Pim van Pelt

Intro: Pim van Pelt (PBVP1-RIPE)

- Member of the RIPE community since 1999 (RIPE #34)
  - Has used [pim@ipng.nl] for 25 years
  - And also [pim@ipng.ch] for 18 years
  - Incorporated [ipng.ch] in Switzerland in 2021

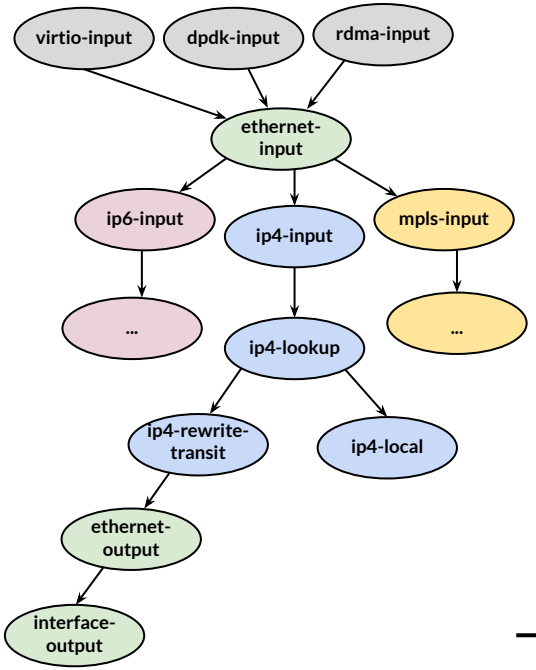
Intro: IPng Networks - AS8298





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

- Fourteen VPP/Bird2 routers [ref] (UN/LOCODE names)
- European ring: *peering on the FLAP** [ref] ~2'150 adjacencies
- Acquired AS8298 from SixXS [ref]



Intro: Vector Packet Processing

FD.io VPP [ref] is an open source dataplane that:

- runs in userspace,
- provides *very* fast networking,
- using DPDK, RDMA, VirtIO, VMXNet3, AVF, ...
- easily exceeds 100Mpps+ and 100Gbps+
- on commodity x86_64 / amd64 hardware!

See FOSDEM'22 [video] or GRNOG #16 [video]

- Contributed* to **Linux Control Plane** plugin [GitHub]
- LinuxCP adds BGP/OSPF/VRRP/etc to VPP

This talk: **ARP/ND/Unnumbered in VPP** and **OSPFv3+ in Bird2.**

*) *Thanks to Pierre Pfister, Neale Ranns, Matt Smith and Jon Loeliger for the [collaboration]!*

Intro: VPP LinuxCP


```
pim@hippo:~$ vppctl lcp create HundredGigabitEthernet4/0/0 host-if ice0
```

Intro: VPP LinuxCP


```
pim@hippo:~$ vppctl lcp create HundredGigabitEthernet4/0/0 host-if ice0
pim@hippo:~$ sudo ip link set ice0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev ice0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev ice0
```

Intro: VPP LinuxCP


```
pim@hippo:~$ vppctl lcp create HundredGigabitEthernet4/0/0 host-if ice0
pim@hippo:~$ sudo ip link set ice0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev ice0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev ice0

pim@hippo:~$ sudo ip link add link ice0 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
```

Intro: VPP LinuxCP


```
pim@hippo:~$ vppctl lcp create HundredGigabitEthernet4/0/0 host-if ice0
pim@hippo:~$ sudo ip link set ice0 up mtu 9000
pim@hippo:~$ sudo ip address add 2001:db8:0:1::2/64 dev ice0
pim@hippo:~$ sudo ip address add 192.0.2.2/24 dev ice0

pim@hippo:~$ sudo ip link add link ice0 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:~$ ping6 esnog.net
PING esnog.net (2001:7f8:f:112::114): 56 data bytes
64 bytes from 2001:7f8:f:112::114: icmp_seq=0 hlim=53 time=29.080 ms
64 bytes from 2001:7f8:f:112::114: icmp_seq=1 hlim=53 time=29.060 ms
...
```

Act 1: Babel with VPP

Babel: IPv4 routes with IPv6 next hop

Status

- ✓ Babel: Adjacency
- Babel: Learning
- VPP: FIB
- VPP: Forwarding
- VPP: ICMPv4

```
pim@vpp0-0:~$ cat /etc/bird/bird.conf
protocol babel {
  interface "e*" {
    type wired;
    extended next hop on;
  };
  ipv6 { import all; export all; };
  ipv4 { import all; export all; };
}
pim@vpp0-0:~$ birdc show babel interfaces
BIRD 2.14 ready.
babel1:
Interface State Auth RX cost  Nbrs Timer Next hop (v4) Next hop (v6)
e1        Up    No        96     1 0.958 ::            fe80::5054:ff:fef0:1101

pim@vpp0-0:~$ birdc show babel neighbors
BIRD 2.14 ready.
babel1:
IP address               Interface Metric Routes Hellos Expires Auth  RTT (ms)
fe80::5054:ff:fef0:1110  e1            96      8     16   5.003 No       4.831
```

Babel: IPv4 routes with IPv6 next hop

Status

- ✓ Babel: Adjacency
- ✓ Babel: Learning
- VPP: FIB
- VPP: Forwarding
- VPP: ICMPv4

```
pim@vpp0-0:~$ birdc show babel entries
BIRD 2.14 ready.
babel1:
Prefix                  Router ID               Metric Seqno  Routes Sources
192.168.10.0/32         00:00:00:00:c0:a8:0a:00      0     1       0       0
192.168.10.0/24         00:00:00:00:c0:a8:0a:00      0     1       1       0
192.168.10.1/32         00:00:00:00:c0:a8:0a:01     96     7       1       0
2001:678:d78:200::/128  00:00:00:00:c0:a8:0a:00      0     1       0       0
2001:678:d78:200::/60   00:00:00:00:c0:a8:0a:00      0     1       1       0
2001:678:d78:200::1/128 00:00:00:00:c0:a8:0a:01     96     7       1       0


pim@vpp0-0:~$ ip -6 ro | grep e1
2001:678:d78:200::1/128 via fe80::5054:ff:fef0:1110 dev e1 proto bird metric...
fe80::/64 dev e1 proto kernel metric 256 pref medium


pim@vpp0-0:~$ ip -4 ro | grep e1
192.168.10.1 via inet6 fe80::5054:ff:fef0:1110 dev e1 proto bird metric 32
```

VPP: IPv4 routes with IPv6 next hop

VPP already allows cross-address family nexthops:

- Added by vifino@ in Gerrit [38633]
- Uses netlink `rtnl_route_nh_get_via()` from libnl 3.4+

Status

- ✓ Babel: Adjacency
- ✓ Babel: Learning
- ✓ VPP: FIB
- VPP: Forwarding
- VPP: ICMPv4

```
pim@vpp0-0:~$ vppctl show ip fib 192.168.10.1
ipv4-VRF:0, fib_index:0, flow hash:[src dst sport dport proto flowlabel ] epoch:0 flags:none
locks:[default-route:1, lcp-rt:1, ]
192.168.10.1/32 fib:0 index:31 locks:2
  lcp-rt-dynamic refs:1 src-flags:added,contributing,active,
        path-list:[51] locks:4 flags:shared, uPRF-list:42 len:1 itfs:[2, ]
        path:[72] pl-index:51 ip6 weight=1 pref=32 attached-nexthop:  oper-flags:resolved,
              fe80::5054:ff:fef0:1110 GigabitEthernet10/0/1
        [@0]: ipv6 via fe80::5054:ff:fef0:1110 GigabitEthernet10/0/1: mtu:9000 next:7 flags:[]
              525400f01110525400f0110186dd

 forwarding:   unicast-ip4-chain
  [@0]: dpo-load-balance: [proto:ip4 index:34 buckets:1 uRPF:42 to:[0:0]]
        [0] [@5]: ipv4 via fe80::5054:ff:fef0:1110 GigabitEthernet10/0/1: mtu:9000 next:7
                  flags:[] 525400f01110525400f011010800
```

VPP: Forwarding IPv4 on non-ip4 interface

Status

- ✓ Babel: Adjacency
- ✓ Babel: Learning
- ✓ VPP: FIB
- ✗ VPP: Forwarding
- VPP: ICMPv4

```
pim@vpp0-1:~$ vppctl show trace
...
07:42:53:178765: ethernet-input
  frame: flags 0x1, hw-if-index 1, sw-if-index 1
  IP4: 52:54:00:f0:11:01 -> 52:54:00:f0:11:10

07:42:53:178791: ip4-input
  ICMP: 192.168.10.0 -> 192.168.10.1
       tos 0x00, ttl 64, length 84, checksum 0xb02b dscp CS0 ecn NON_ECN
       fragment id 0xf52b, flags DONT_FRAGMENT
  ICMP echo_request checksum 0x43b7 id 26166

07:42:53:178810: ip4-not-enabled
       ICMP: 192.168.10.0 -> 192.168.10.1
       tos 0x00, ttl 64, length 84, checksum 0xb02b dscp CS0 ecn NON_ECN
       fragment id 0xf52b, flags DONT_FRAGMENT
       ICMP echo_request checksum 0x43b7 id 26166

07:42:53:178833: error-drop
  rx:GigabitEthernet10/0/0

07:42:53:178835: drop
  dpdk-input: no error
```

VPP: Forwarding IPv4 on non-ip4 interface

Attempt 1: `ip4_sw_interface_enable_disable()` in Linux CP

- Forwarding works …
- … but breaks ICMPv4 (eg. Path MTU)

Status

✓ Babel: Adjacency
✓ Babel: Learning
✓ VPP: FIB
✓ VPP: Forwarding
❌ VPP: ICMPv4

```
pim@vpp0-0:~$ ping -c3 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=3.92 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=3.81 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=64 time=3.75 ms
--- 192.168.10.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 2.673/3.477/3.921/0.467 ms

pim@vpp0-0:~$ traceroute -n 192.168.10.3
traceroute to 192.168.10.3 (192.168.10.3), 30 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  192.168.10.3 (192.168.10.3)  10.418 ms  10.343 ms  11.362 ms
```

VPP: ARP for *on-link* IPv4 nexthop

Attempt 2: `set interface unnumbered` in VPP

- ICMPv4 works …
- … but forwarding breaks, VPP drops *on-link* ARP req

Status

- ✓ Babel: Adjacency
- ✓ Babel: Learning
- ✓ VPP: FIB
- ❌ VPP: Forwarding
- ✓ VPP: ICMPv4

```
vpp0-2# set interface unnumbered GigabitEthernet10/0/0 use loop0
vpp0-2# set interface unnumbered GigabitEthernet10/0/1 use loop0

pim@vpp0-2:~$ ip -br a
lo               UNKNOWN    127.0.0.1/8 ::1/128
loop0            UNKNOWN    192.168.10.2/32 2001:678:d78:200::2/128 fe80::dcad:ff:fe00:0/64
e0               UP         192.168.10.2/32 2001:678:d78:200::2/128 fe80::5054:ff:fef0:1120/64
e1               UP         192.168.10.2/32 2001:678:d78:200::2/128 fe80::5054:ff:fef0:1121/64
pim@vpp0-2:~$ ip ro
192.168.10.0 via 192.168.10.1 dev e0 proto bird metric 32 onlink
unreachable 192.168.10.0/24 proto bird metric 32
192.168.10.1 via 192.168.10.1 dev e0 proto bird metric 32 onlink
192.168.10.3 via 192.168.10.3 dev e1 proto bird metric 32 onlink

vpp0-2# show err
   Count                  Node                                Reason               Severity
         5             arp-reply            IP4 source address not local to sub    error
```

VPP: Forwarding IPv4 on unnumbered

Attempt 3: `set interface unnumbered` in VPP, inhibit Linux CP

- ARP issue fixed by pim@ in Gerrit [40482]
- Linux CP: inhibit sync of *unnumbered* to Linux [GitHub]

Status

- ✓ Babel: Adjacency
- ✓ Babel: Learning
- ✓ VPP: FIB
- ✓ VPP: Forwarding
- ✓ VPP: ICMPv4

```
vpp0-2# set interface unnumbered GigabitEthernet10/0/0 use loop0
vpp0-2# set interface unnumbered GigabitEthernet10/0/1 use loop0
vpp0-2# lcp lcp-sync-unnumbered disable
pim@vpp0-2:~$ ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
loop0            UNKNOWN        192.168.10.2/32 2001:678:d78:200::2/128 fe80::dcad:ff:fe00:0/64
e0               UP             fe80::5054:ff:fef0:1120/64
e1               UP             fe80::5054:ff:fef0:1121/64
pim@vpp0-2:~$ ip ro
192.168.10.0 via inet6 fe80::5054:ff:fef0:1121 dev e0 proto bird metric 32 onlink
unreachable 192.168.10.0/24 proto bird metric 32
192.168.10.1 via inet6 fe80::5054:ff:fef0:1121 dev e0 proto bird metric 32 onlink
192.168.10.3 via inet6 fe80::5054:ff:fef0:1123 dev e1 proto bird metric 32 onlink
pim@vpp0-0:~$ traceroute -n 192.168.10.3
traceroute to 192.168.10.3 (192.168.10.3), 30 hops max, 60 byte packets
 1  192.168.10.1  1.882 ms  2.231 ms  1.472 ms
 2  192.168.10.2  4.243 ms  3.492 ms  2.797 ms
 3  192.168.10.3  6.689 ms  5.925 ms  5.157 ms
```

Act 2: IPv4 OSPFv3 with VPP

OSPFv3: Enter a gross hack

[RFC 5838]: support multiple address families in OSPFv3

Although IPv6 link local addresses could be used as next hops for IPv4 …

… it is desirable to have IPv4 next-hop addresses

... IPv4 will be advertised in the "link local address" field in Link-LSA

… address is placed in the first 32 bits of the "link local address" field

… and the remaining bits MUST be set to zero.

This approach fundamentally breaks IPv6 next-hops!

`s/could be used/`**`cannot ever be used`**`/`

OSPFv3: Unnumbered

Clever solution by santiago@ in [commit] to Bird2:

Add `update_loopback_addr()` to scan *all* IPv4 interfaces

1. prefer *host* (/32) addresses
2. else use *OSPF stub* addresses
3. else just any old IPv4 address

No interface address?

- Find one for the (RFC5838) Link-LSA
- Learn routes as `RNF_ONLINK` from /32 neighbors

VPP: Unnumbered with OSPFv3

Match made in heaven:

1. Bird2 [Commit] makes neighbors *on-link*
2. VPP [Gerrit] makes *on-link* ARP resolution work
3. Linux CP [GitHub] inhibits unnumbered interfaces

- Allows for *exactly one* IPv4 and IPv6 on `loop0`
- Bonus: OSPF for IPv4 and IPv6 can now share BFD!

VPP: Unnumbered with OSPFv3

Status

✓ VPP+BFD: Config
OSPFv3: Config
BFD+OSPFv3: Adjacency
OSPFv3: Learning
VPP: FIB, ICMP, ICMPv6

```
vpp0-2# lcp lcp-sync-unnumbered disable
vpp0-2# set interface ip address loop0 2001:678:d78:200::2/128
vpp0-2# set interface ip address loop0 192.168.10.2/32
vpp0-2# set interface unnumbered GigabitEthernet10/0/0 use loop0
vpp0-2# set interface unnumbered GigabitEthernet10/0/1 use loop0

pim@vpp0-2:~$ ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
loop0            UNKNOWN        192.168.10.2/32 2001:678:d78:200::2/128 fe80::dcad:...
e0               UP             fe80::5054:ff:fef0:1120/64
e1               UP             fe80::5054:ff:fef0:1121/64

pim@vpp0-2:~$ cat /etc/bird/core/bfd.conf
protocol bfd bfd1 {
  interface "e*" {
    interval 100 ms;
    multiplier 20;
  };
}
```

VPP: Unnumbered with OSPFv3

Status

- ✓ VPP+BFD: Config
- ✓ OSPFv3: Config
- BFD+OSPFv3: Adjacency
- OSPFv3: Learning
- VPP: FIB, ICMP, ICMPv6

```
vpp0-2# lcp lcp-sync-unnumbered disable
vpp0-2# set interface ip address loop0 2001:678:d78:200::2/128
vpp0-2# set interface ip address loop0 192.168.10.2/32
vpp0-2# set interface unnumbered GigabitEthernet10/0/0 use loop0
vpp0-2# set interface unnumbered GigabitEthernet10/0/1 use loop0
pim@vpp0-2:~$ cat /etc/bird/core/ospf.conf
protocol ospf v3 ospf4 {
  ipv4 { export all; import all; };
  area 0 {
    interface "loop0" { stub yes; };
    interface "e*" { type pointopoint; cost 5; bfd on; };
  };
}
protocol ospf v3 ospf6 {
  ipv6 { export all; import all; };
  area 0 {
      interface "loop0" { stub yes; };
      interface "e*" { type pointopoint; cost 5; bfd on; };
  };
}
```

VPP: Unnumbered with OSPFv3

Status

- ✓ VPP+BFD: Config
- ✓ OSPFv3: Config
- ✓ BFD+OSPFv3: Adjacency
- OSPFv3: Learning
- VPP: FIB, ICMP, ICMPv6

```
pim@vpp0-2:~$ birdc show bfd session
BIRD v2.15.1-4-g280daed5-x ready.
bfd1:
IP address                Interface State  Since         Interval Timeout
fe80::5054:ff:fef0:1111   e0        Up     16:52:45.453     0.100 2.000
fe80::5054:ff:fef0:1130   e1        Up     16:53:06.857     0.100 2.000

pim@vpp0-2:~$ birdc show ospf neighbors
BIRD v2.15.1-4-g280daed5-x ready.
ospf4:
Router ID          Pri      State    DTime   Interface  Router IP
192.168.10.1       1     Full/PtP    36.931  e0         fe80::5054:ff:fef0:1111
192.168.10.3       1     Full/PtP    35.982  e1         fe80::5054:ff:fef0:1130

ospf6:
Router ID          Pri      State    DTime   Interface  Router IP
192.168.10.1       1     Full/PtP    36.931  e0         fe80::5054:ff:fef0:1111
192.168.10.3       1     Full/PtP    35.982  e1         fe80::5054:ff:fef0:1130
```

VPP: Unnumbered with OSPFv3

Status

- ✓ VPP+BFD: Config
- ✓ OSPFv3: Config
- ✓ BFD+OSPFv3: Adjacency
- ✓ OSPFv3: Learning

VPP: FIB, ICMP, ICMPv6

```
pim@vpp0-2:~$ birdc show route all for 192.168.10.3
BIRD v2.15.1-4-g280daed5-x ready.
Table master4:
192.168.10.3/32   unicast [ospf4 16:53:12.259] * I (150/5) [192.168.10.3]
      via 192.168.10.3 on e1 onlink
      Type: OSPF univ
      OSPF.metric1: 5
      OSPF.router_id: 192.168.10.3

pim@vpp0-2:~$ ip ro
default via 192.168.10.1 dev e0 proto bird metric 32 onlink
192.168.10.0 via 192.168.10.1 dev e0 proto bird metric 32 onlink
unreachable 192.168.10.0/24 proto bird metric 32
192.168.10.1 via 192.168.10.1 dev e0 proto bird metric 32 onlink
192.168.10.3 via 192.168.10.3 dev e1 proto bird metric 32 onlink
192.168.10.4/31 via 192.168.10.1 dev e0 proto bird metric 32 onlink
```

VPP: Unnumbered with OSPFv3

Status

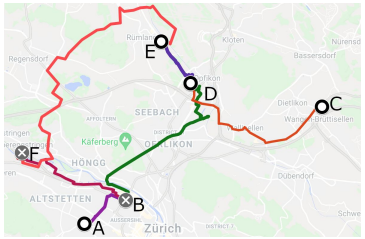
- ✓ VPP+BFD: Config
- ✓ OSPFv3: Config
- ✓ BFD+OSPFv3: Adjacency
- ✓ OSPFv3: Learning
- ✓ VPP: FIB, ICMP, ICMPv6

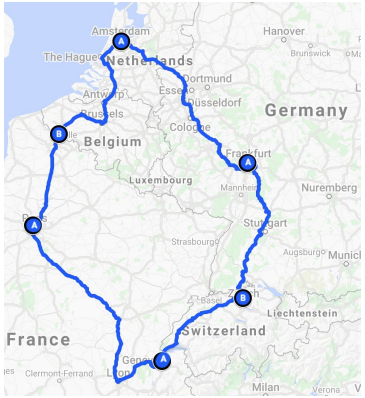
```
pim@lab:~$ traceroute -4 vpp0-3 9000
traceroute to vpp0-3.lab (192.168.10.3), 30 hops max, 9000 byte packets
 1  vpp0-0.lab.ipng.ch (192.168.10.0)  2.274 ms  0.621 ms  1.012 ms
 2  vpp0-1.lab.ipng.ch (192.168.10.1)  2.936 ms  3.515 ms  4.015 ms
 3  vpp0-2.lab.ipng.ch (192.168.10.2)  5.751 ms  6.218 ms  5.544 ms
 4  vpp0-3.lab.ipng.ch (192.168.10.3)  9.446 ms  9.531 ms  9.694 ms

pim@lab:~$ traceroute -6 vpp0-3 9000
traceroute to vpp0-3.lab (2001:678:d78:200::3), 30 hops max, 9000 byte packets
 1  vpp0-0.lab.ipng.ch (2001:678:d78:200::)  1.295 ms  1.702 ms  0.972 ms
 2  vpp0-1.lab.ipng.ch (2001:678:d78:200::1)  2.554 ms  2.881 ms  2.236 ms
 3  vpp0-2.lab.ipng.ch (2001:678:d78:200::2)  5.081 ms  4.364 ms  3.628 ms
 4  vpp0-3.lab.ipng.ch (2001:678:d78:200::3)  6.268 ms  6.812 ms  7.267 ms
```

Act 3: Rollout in AS8298

AS8298: Removing IP4/IP6 PtP addresses





Start situation:

- Each router has an IPv4 /32 and IPv6 /128 `loop0`
- IPv4 OSPF with /31 PtP, IPv6 OSPFv3 with /112 PtP

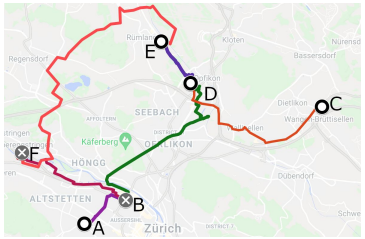
Game plan:

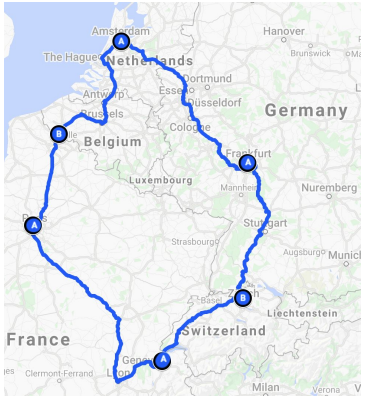
1. Upgrade bird2, upgrade VPP dataplane:
   - rename ‘ospf4’ to ‘ospf4_old’ (which is OSPFv2)
   - add an empty ‘ospf4’ (which is OSPFv3)
2. Reconfigure VPP interfaces to *unnumbered*
3. Move all interfaces from ‘ospf4_old’ to ‘ospf4’
4. Finally, delete ‘ospf4_old’

End situation:

- Each router has *only one* IPv4 /32 and IPv6 /128 `loop0`
- IPv4 OSPF with /31 PtP, IPv6 OSPFv3 with /112 PtP

Step 1: Upgrade software



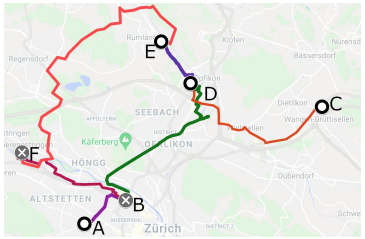


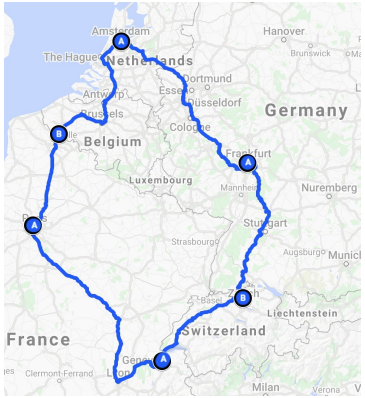

```
pim@ddln0:~$ sed -rn 's,cost (.+),cost 10\1,' /etc/bird/core/ospf.conf
pim@ddln0:~$ birdc configure

pim@ddln0:~$ sed -i 's,protocol.*ospf4,$1_old,' /etc/bird/core/ospf.conf
pim@ddln0:~$ scp bookworm-builder:bird2_2.15.1_amd64.deb .
pim@ddln0:~$ wget -m --no-parent \
             https://ipng.ch/media/vpp/bookworm/24.06-rc0~183-gb0d433978/

pim@ddln0:~$ sudo nsenter --net=/var/run/netns/dataplane
root@ddln0:~# pkill -9 vpp && systemctl stop vpp bird-dataplane
root@ddln0:~# dpkg -i ~pim/ipng.ch/media/vpp/bookworm/*/*.deb
root@ddln0:~# dpkg -i ~pim/bird2_2.15.1_amd64.deb
root@ddln0:~# systemctl start bird-dataplane
root@ddln0:~# systemctl restart vpp-snmp-agent-dataplane
root@ddln0:~# systemctl restart vpp-exporter-dataplane
```

Step 1: Upgrade software



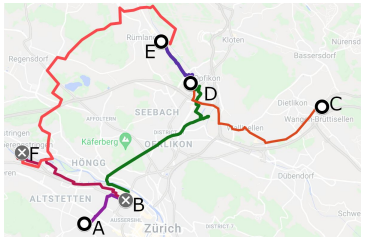


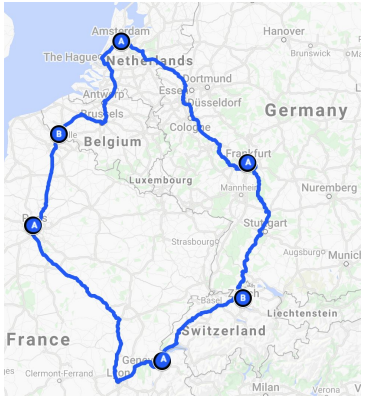

```
pim@summer:~$ ping ddln0.ipng.ch
PING ddln0.ipng.ch (194.1.163.5) 56(84) bytes of data.
64 bytes from ddln0.ipng.ch (194.1.163.5): icmp_seq=1 ttl=61 time=1.94 ms
64 bytes from ddln0.ipng.ch (194.1.163.5): icmp_seq=2 ttl=61 time=1.00 ms
...
64 bytes from ddln0.ipng.ch (194.1.163.5): icmp_seq=94 ttl=61 time=1001.83 ms
64 bytes from ddln0.ipng.ch (194.1.163.5): icmp_seq=95 ttl=61 time=1.03 ms

pim@ddln0:~$ birdc show ospf nei
BIRD v2.15.1-4-g280daed5-x ready.
ospf4_old:
Router ID   Pri        State        DTime Interface  Router IP
194.1.163.6   1   Full/PtP    32.113      xe1-1      194.1.163.27
194.1.163.0   1   Full/PtP    30.936      xe1-0.304  194.1.163.24

ospf6:
Router ID   Pri        State        DTime Interface  Router IP
194.1.163.6   1   Full/PtP    32.113      xe1-1      fe80::3eec:efff:fe46:68a8
194.1.163.0   1   Full/PtP    30.936      xe1-0.304  fe80::6a05:caff:fe32:4616
```

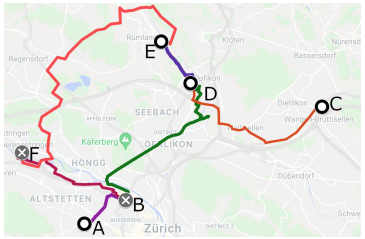
Step 2: Reconfigure VPP

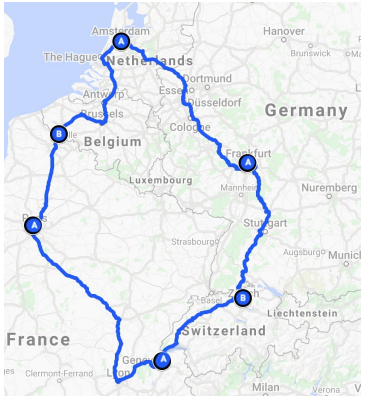





```
pim@ddln0:~$ vim /etc/vpp/vppcfg.yaml
...
loopbacks:
  loop0:
    description: 'Core: ddln0.ipng.ch'
    addresses: ['194.1.163.5/32', '2001:678:d78::5/128']
    lcp: loop0
    mtu: 9000
interfaces:
  TenGigabitEthernet6/0/1:
    device-type: dpdk
    description: 'Core: ddln1.ipng.ch'
    mtu: 9000
#    lcp: xe1-1
#    addresses: [ '194.1.163.20/31', '2001:678:d78::2:5:1/112' ]
    lcp: ddln1
    unnumbered: loop0
```

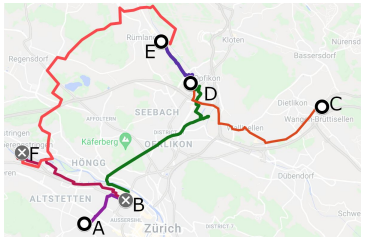
Step 3: Move interface to OSPFv3

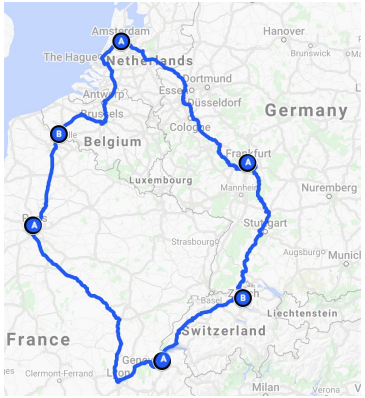





```
pim@ddln0:~$ vim /etc/bird/core/ospf.conf
protocol ospf v2 ospf4_old {
  ipv4 { export filter f_ospf; import filter f_ospf; };
  area 0 {
    interface "loop0" { stub yes; };
#    interface "xe1-1" { type pointopoint; cost 10; bfd on; };
    interface "xe1-0.304" { type pointopoint; cost 56; bfd on; };
  };
}
protocol ospf v3 ospf4 {
  ipv4 { export filter f_ospf; import filter f_ospf; };
  area 0 {
    interface "loop0","lo" { stub yes; };
    interface "ddln1" { type pointopoint; cost 10; bfd on; };
  };
}
```

Step 3: Move interface to OSPFv3



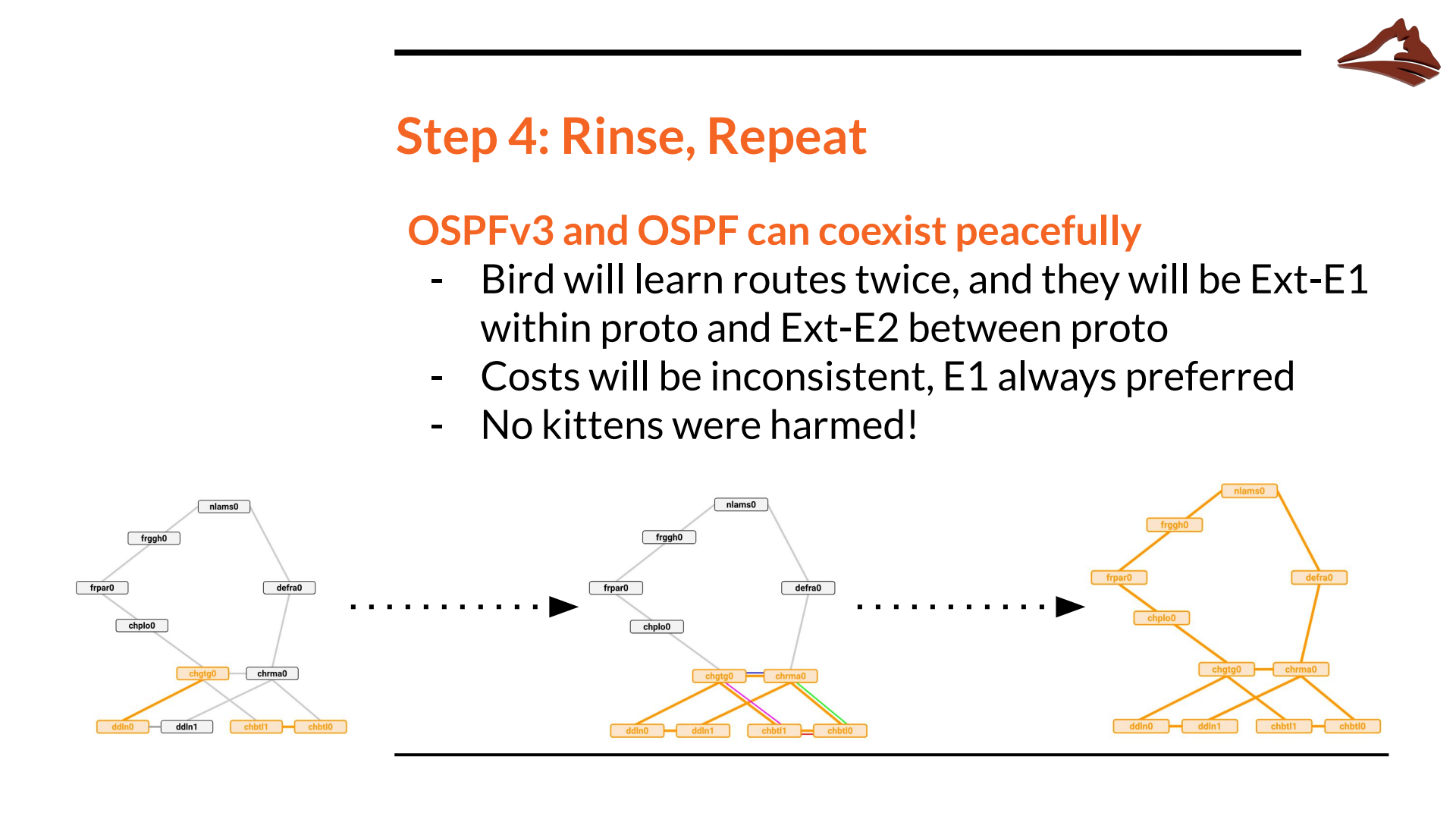



```
pim@ddln0:~$ birdc show ospf nei
BIRD v2.15.1-4-g280daed5-x ready.
ospf4_old:
Router ID   Pri         State        DTime Interface  Router IP
194.1.163.0   1   Full/PtP    30.936      xe1-0.304  194.1.163.24

ospf4:
Router ID   Pri         State        DTime Interface  Router IP
194.1.163.6   1   Full/PtP    32.113      ddln1      fe80::3eec:efff:fe46:68a8

ospf6:
Router ID   Pri         State        DTime Interface  Router IP
194.1.163.6   1   Full/PtP    32.113      ddln1      fe80::3eec:efff:fe46:68a8
194.1.163.0   1   Full/PtP    30.936      xe1-0.304  fe80::6a05:caff:fe32:4616

pim@ddln0:~$ $ birdc show route for 194.1.163.6
BIRD v2.15.1-4-g280daed5-x ready.
Table master4:
194.1.163.6/32    unicast [ospf4 2024-06-19 18:07:59] * I (150/5) [194.1.163.6]
      via 194.1.163.6 on ddln1 onlink
```

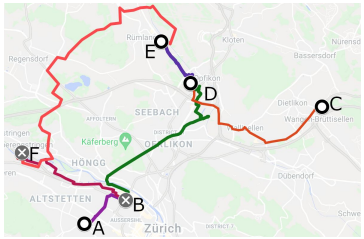


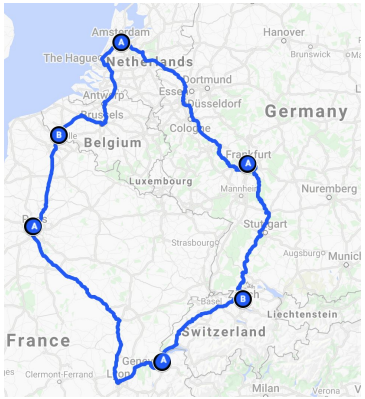
Step 4: Rinse, Repeat

OSPFv3 and OSPF can coexist peacefully

- Bird will learn routes twice, and they will be Ext-E1 within proto and Ext-E2 between proto
- Costs will be inconsistent, E1 always preferred
- No kittens were harmed!

Results






```
pim@squanchy:~$ traceroute bit.nl
traceroute to bit.nl (213.136.12.97), 64 hops max, 40 byte packets
 1  chbtl0 (194.1.163.66)  0.55 ms  2.051 ms  0.311 ms
 2  chrma0 (194.1.163.0)  1.369 ms  1.496 ms  1.281 ms
 3  defra0 (194.1.163.7)  6.933 ms  7.007 ms  7.049 ms
 4  nlams0 (194.1.163.8)  13.103 ms  12.93 ms  13.209 ms
 5  as12859.frys-ix.net (185.1.203.186)  17.774 ms  14.625 ms  21.249 ms
 6  http-bit.lb.network.bit.nl (213.136.12.97)  14.468 ms  14.677 ms  14.358 ms
pim@squanchy:~$ traceroute6 bit.nl
traceroute6 to bit.nl (2001:7b8:3:5::80:19), 64 hops max, 60 byte packets
 1  chbtl0.ipng.ch (2001:678:d78:3::1)  0.593 ms  2.858 ms  0.352 ms
 2  chrma0 (2001:678:d78::)  1.248 ms  1.446 ms  1.236 ms
 3  defra0 (2001:678:d78::7)  7.093 ms  7.083 ms  7.188 ms
 4  nlams0 (2001:678:d78::8)  13.201 ms  13.103 ms  13.17 ms
 5  as12859.frys-ix.net (2001:7f8:10f::323b:186)  14.488 ms  16.462 ms  17.489 ms
 6  http-bit.lb.network.bit.nl (2001:7b8:3:5::80:19)  14.027 ms  14.127 ms  14.118 ms
```

AS8298 returned:

- 27x IPv4 /31s and IPv6 /112s in total.

Dell R730
(2016)

Act 4: Performance of VPP

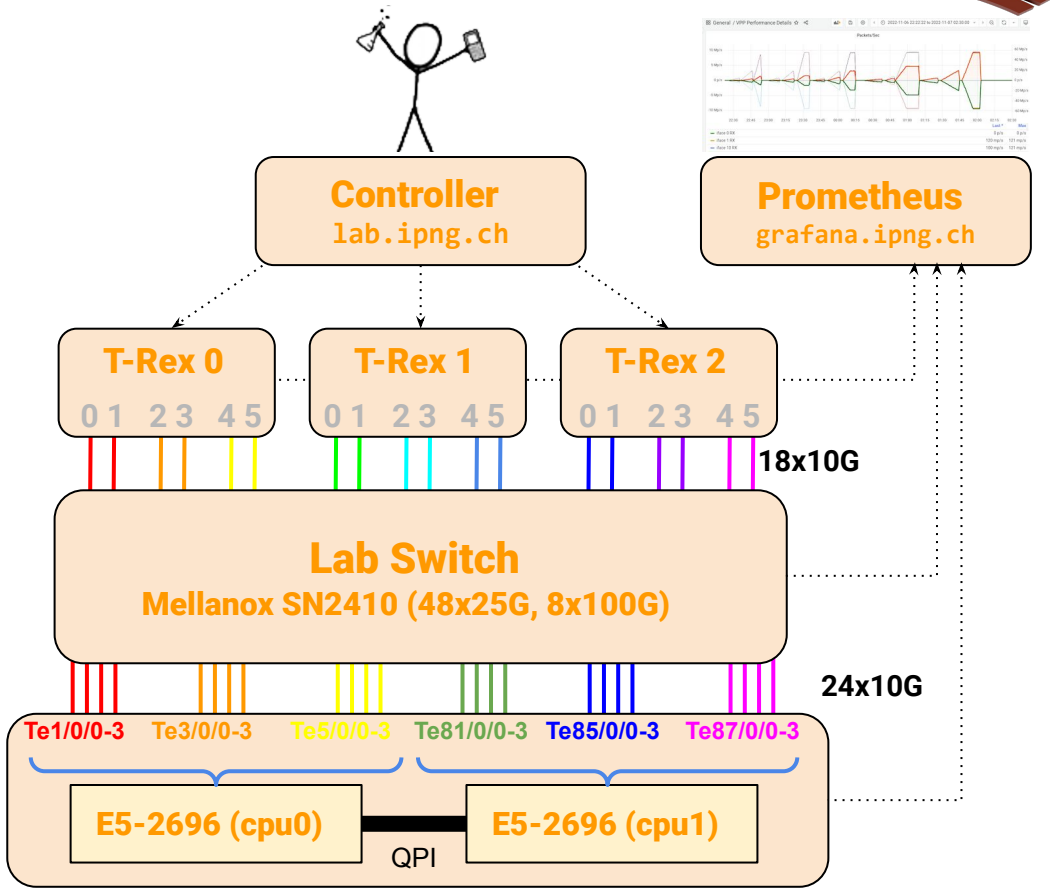
Lab Setup

Load Generator Machines

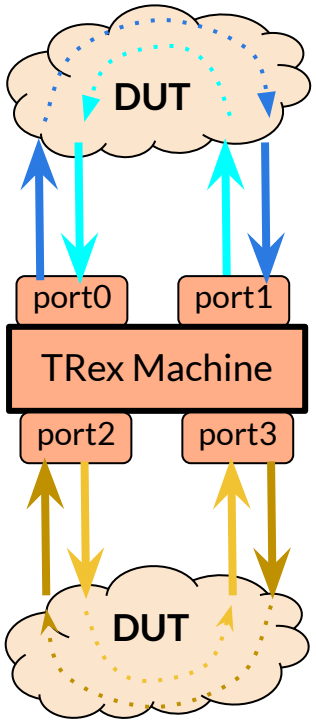
- 3x Dell R720 (E5-2620, 2.00GHz)
- 9x Dual Intel 82599ES (18x10G)
- Debian Bookworm, T-Rex v3.04
  - 4 CPUs per 10G interface pair

Device Under Test

- Dell R730 (E5-2696 v4 @ 2.20GHz)
  - 2x(22C/44T), 64GB DDR4 2.4GT/s
  - 2x40 PCIe v3.0 Lanes
- 6x Intel X710-DA4 (24x10G)
  - 12x10G on cpu0, 12x10G on cpu1
- Debian Bookworm (Linux 6.1.0-25)
- VPP v24.10-rc0~204-ge9bc33201



T-Rex: Config and Startup



Stateless configuration

```
- version: 2
  interfaces: ['04:00.0', '04:00.1', '06:00.0', '06:00.1', ...]
  port_info:
      - src_mac    : 9c:69:b4:61:ff:40 # T-Rex Nic0
        dest_mac   : 3c:ec:ef:c6:fb:26 # DUT MAC A
      - src_mac    : 9c:69:b4:61:ff:41 # T-Rex Nic1
        dest_mac   : 3c:ec:ef:6a:80:db # DUT MAC B
      - src_mac    : 9c:69:b4:89:22:86 # T-Rex Nic2
        dest_mac   : 3c:ec:ef:c6:fb:27 # DUT MAC C
      - src_mac    : 9c:69:b4:89:80:87 # T-Rex Nic3
        dest_mac   : 3c:ec:ef:6a:80:dc # DUT MAC D
```

Startup

```
$ sudo ./t-rex-64 -i -c 4
$ ./trex-console -s <trex-machine>
```

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.

Method 1 - Single Thread Saturation

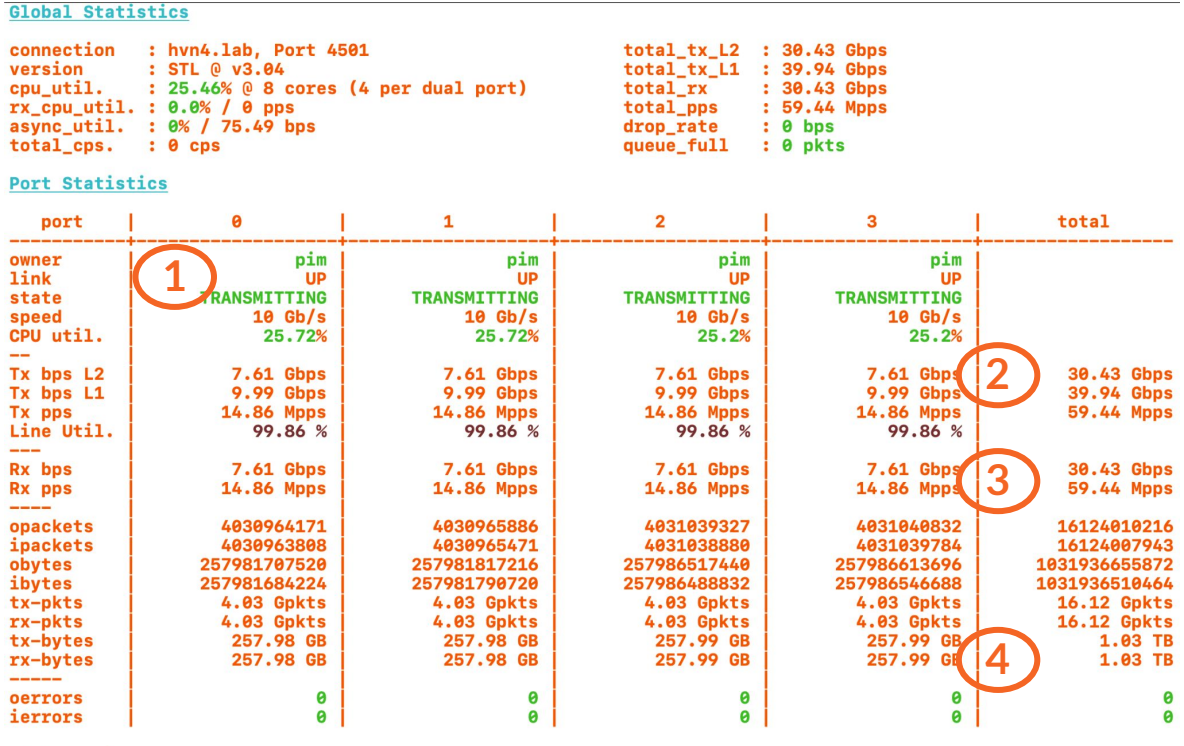
Legend

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: 4x10G @64b

- Tx: 59.44Mpps, 39.94Gbps
- Rx: 59.44Mpps, 39.94Gbps

⇒ L2 XC is (at least) 14.88Mpps per core!



Global Statistics

```
connection   : hvn4.lab, Port 4501          total_tx_L2  : 30.43 Gbps
version      : STL @ v3.04                  total_tx_L1  : 39.94 Gbps
cpu_util.    : 25.46% @ 8 cores (4 per dual port)  total_rx  : 30.43 Gbps
rx_cpu_util. : 0.0% / 0 pps                 total_pps    : 59.44 Mpps
async_util.  : 0% / 75.49 bps               drop_rate    : 0 bps
total_cps.   : 0 cps                        queue_full   : 0 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.	25.72%	25.72%	25.2%	25.2%	
--					
Tx bps L2	7.61 Gbps	7.61 Gbps	7.61 Gbps	7.61 Gbps	30.43 Gbps
Tx bps L1	9.99 Gbps	9.99 Gbps	9.99 Gbps	9.99 Gbps	39.94 Gbps
Tx pps	14.86 Mpps	14.86 Mpps	14.86 Mpps	14.86 Mpps	59.44 Mpps
Line Util.	99.86 %	99.86 %	99.86 %	99.86 %	

Rx bps	7.61 Gbps	7.61 Gbps	7.61 Gbps	7.61 Gbps	30.43 Gbps
Rx pps	14.86 Mpps	14.86 Mpps	14.86 Mpps	14.86 Mpps	59.44 Mpps
----					
opackets	4030964171	4030965886	4031039327	4031040832	16124010216
ipackets	4030963808	4030965471	4031038880	4031039784	16124007943
obytes	257981707520	257981817216	257986517440	257986613696	1031936655872
ibytes	257981684224	257981790720	257986488832	257986546688	1031936510464
tx-pkts	4.03 Gpkts	4.03 Gpkts	4.03 Gpkts	4.03 Gpkts	16.12 Gpkts
rx-pkts	4.03 Gpkts	4.03 Gpkts	4.03 Gpkts	4.03 Gpkts	16.12 Gpkts
tx-bytes	257.98 GB	257.98 GB	257.99 GB	257.99 GB	1.03 TB
rx-bytes	257.98 GB	257.98 GB	257.99 GB	257.99 GB	1.03 TB
-----					
oerrors	0	0	0	0	0
ierrors	0	0	0	0	0

Method 1: Results (E5-2696 v4)

	clocks/packet @ 1kpps	clocks/packet @ 1Mpps	clocks/packet @ 10Mpps	packets/sec per core
L2 xconnect	991	199	140	15.34Mpps
MPLS	1465	274	172	10.35Mpps
L3 IPv4	1596	299	156	11.08Mpps
L3 IPv6	1941	347	178	9.72Mpps

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

Claim 1: VPP can forward 100Gbit

Bidirectional (18 ports, 18 VPP threads):

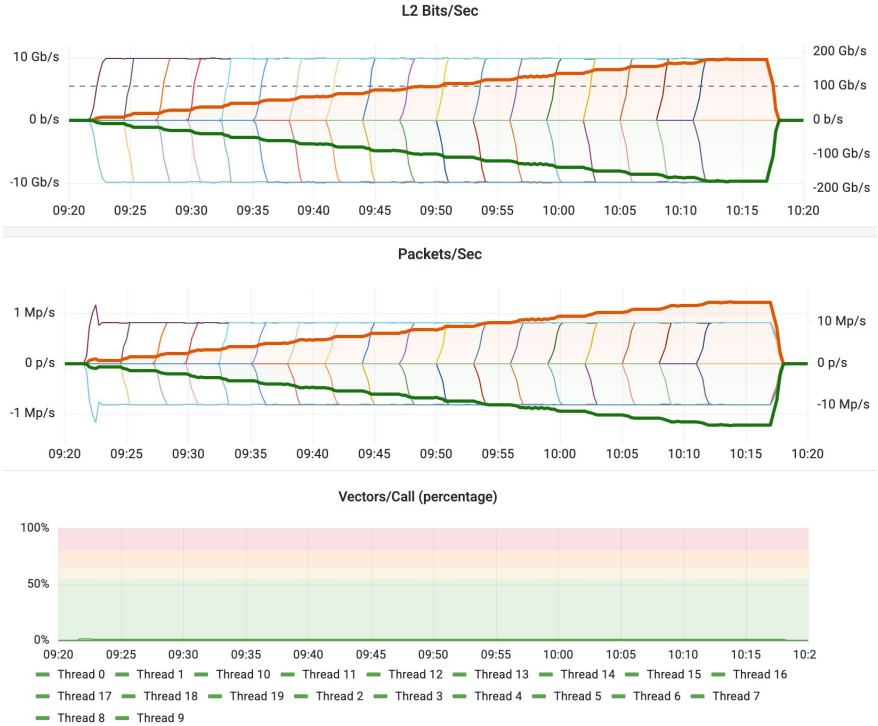
1) 09:22 - Incremental colored traces
T-Rex ports turned on one by one,
3 minutes apart, sending 1514b

2) 09:50 100Gbps achieved

3) 10:12180Gbps achieved
14.7Mpps @1514b

Note: 24 CPU threads unused; 6 NICs unused.

⇒ Proof that VPP (easily) forwards 100Gbps



Claim 2: VPP can forward 100Mpps

Bidirectional (18 ports, 18 VPP threads):

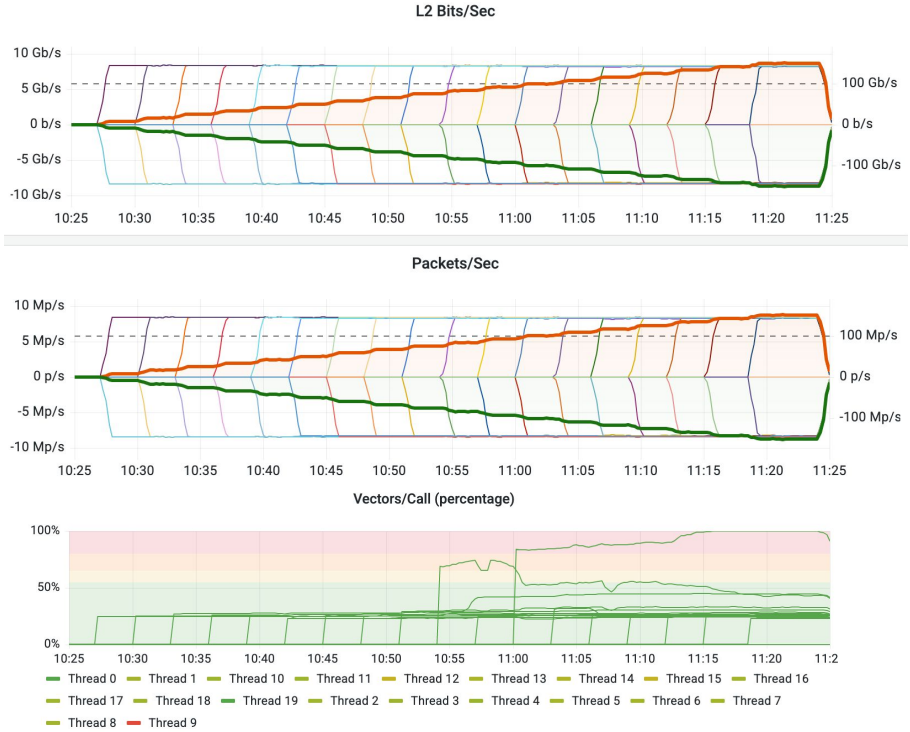
1) 10:26 - Incremental colored traces
T-Rex ports turned on one by one,
3 minutes apart, sending 128b.

2) 11:02 100Mpps achieved

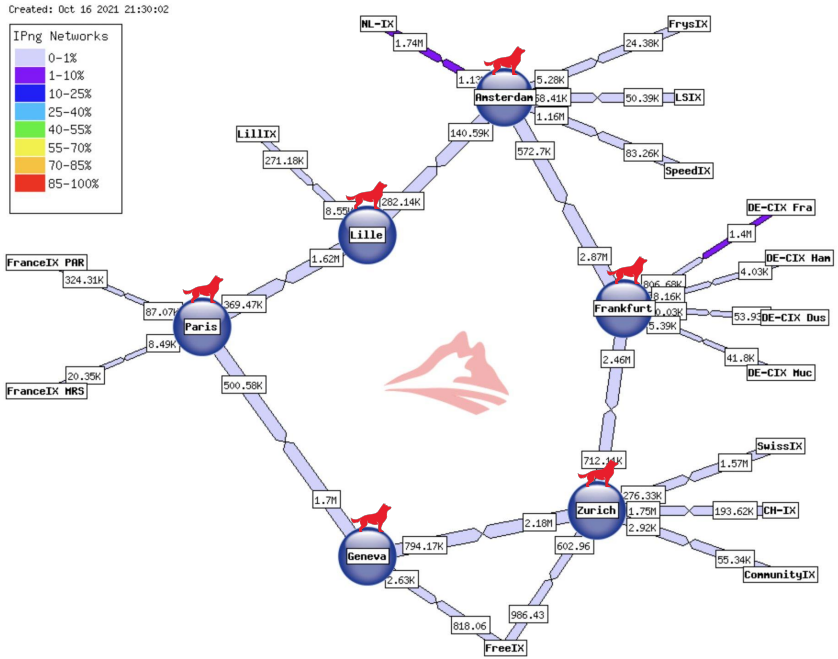
3) 11:19165Mpps achieved
149Gbps @128b

Note: 24 CPU threads unused; 6 NICs unused.

⇒ Proof that VPP (easily) forwards 100Mpps



Questions, Discussion



If you peer with IPng Networks, thanks!
If you don't: please peer with AS8298
<peering@ipng.ch>

Useful Resources

- VPP Mailinglist [vpp-dev@lists.fd.io]
- VPP Linux CP [GitHub]
- Articles [ipng.ch]
- Mastodon [@IPngNetworks]

Also: thanks for listening!