VPP: eVPN

eVPN Management Plane for VPP

Pim van Pelt • 2026–09-04 • NLNOG • Hilversum, Netherlands
go.ipng.ch/nlnog26-vpp

Intro: IPng Networks

Pim van Pelt (PBVP1-RIPE)

Member of the RIPE community since 1999 (RIPE #34)

- Has used [pim@ipng.nl] since 2000
- And also [pim@ipng.ch] since 2006
- Incorporated [ipng.ch] in Switzerland in 2021

Turned 5yo in Aug'26, happy birth date! 🥳

IPng Networks GmbH

- Developer of DPDK and VPP solutions [ref]
- Twenty-two VPP/Bird2 routers [ref] (UN/LOCODE names)
- Operating AS8298 *on the FLAP** [ref] ~2'200 adjacencies

Act 1: Moving Parts

Intro: eVPN

Transport ethernet or IPv4/IPv6 across an underlay

- Controlplane: BGP for Ethernet services [RFC 7432]
- Dataplane: MPLS, SRv6 or VxLAN encapsulation
- Routers advertise the MAC/IP/Prefixes NLRI

L2 Services

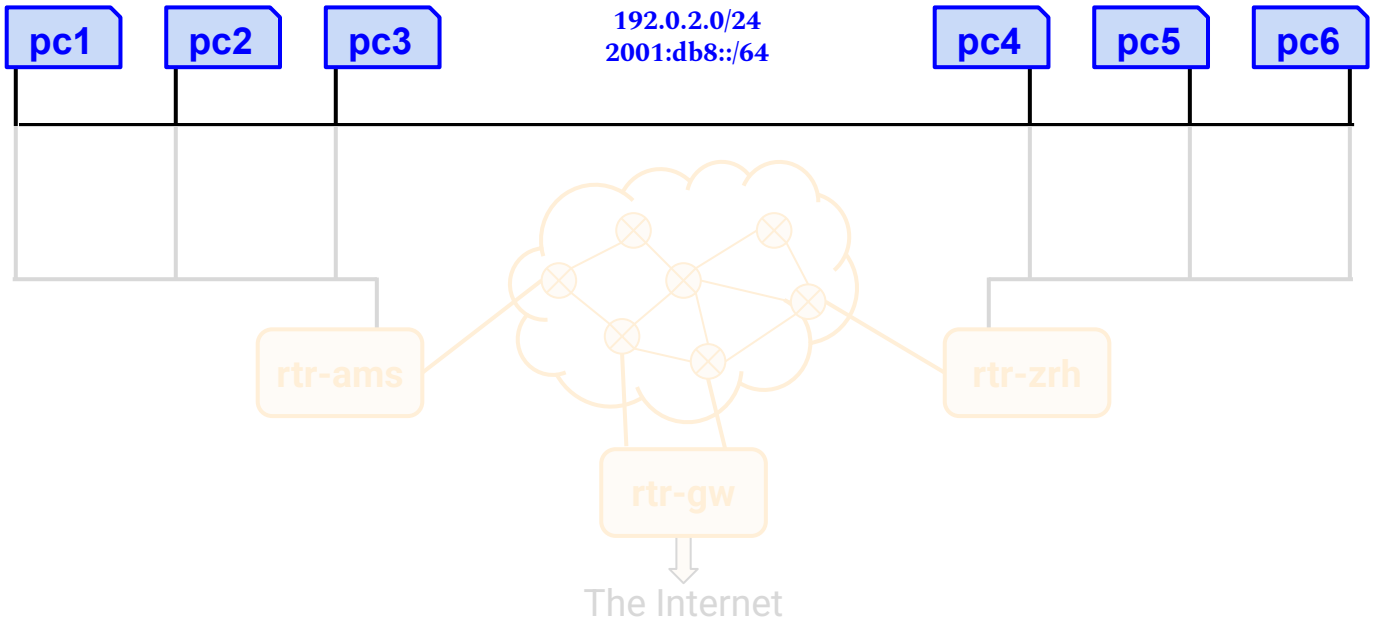
- MAC-VRF [RFC8214, RFC8317, RFC7623]
- Point-to-Point: VPWS or E-LINE
- Point-to-Multipoint: VPLS or E-LAN

L3 Services

- IP-VRF [RFC 9135, RFC9136]
- Asymmetric IRB (ingress router does route lookup)
- Symmetric IRB (all routers share a common L3VNI)

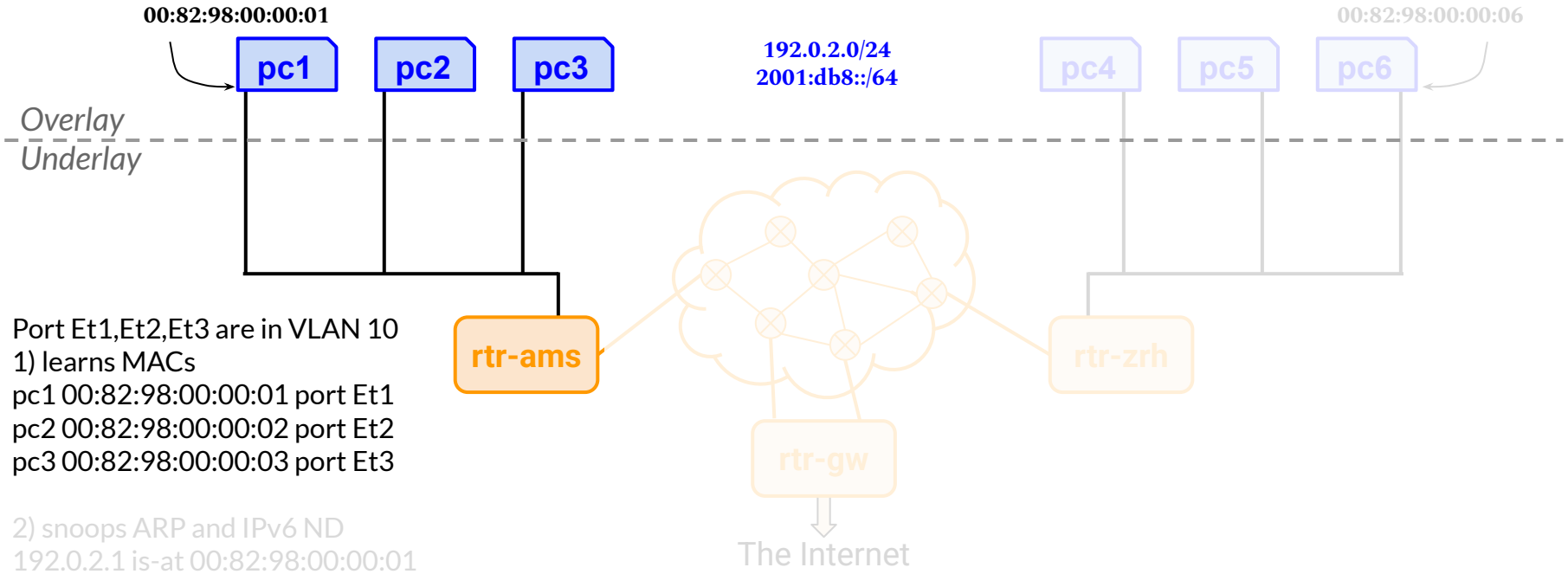
Intro: eVPN

Consider this broadcast domain



Intro: eVPN

Step 1 - just learning

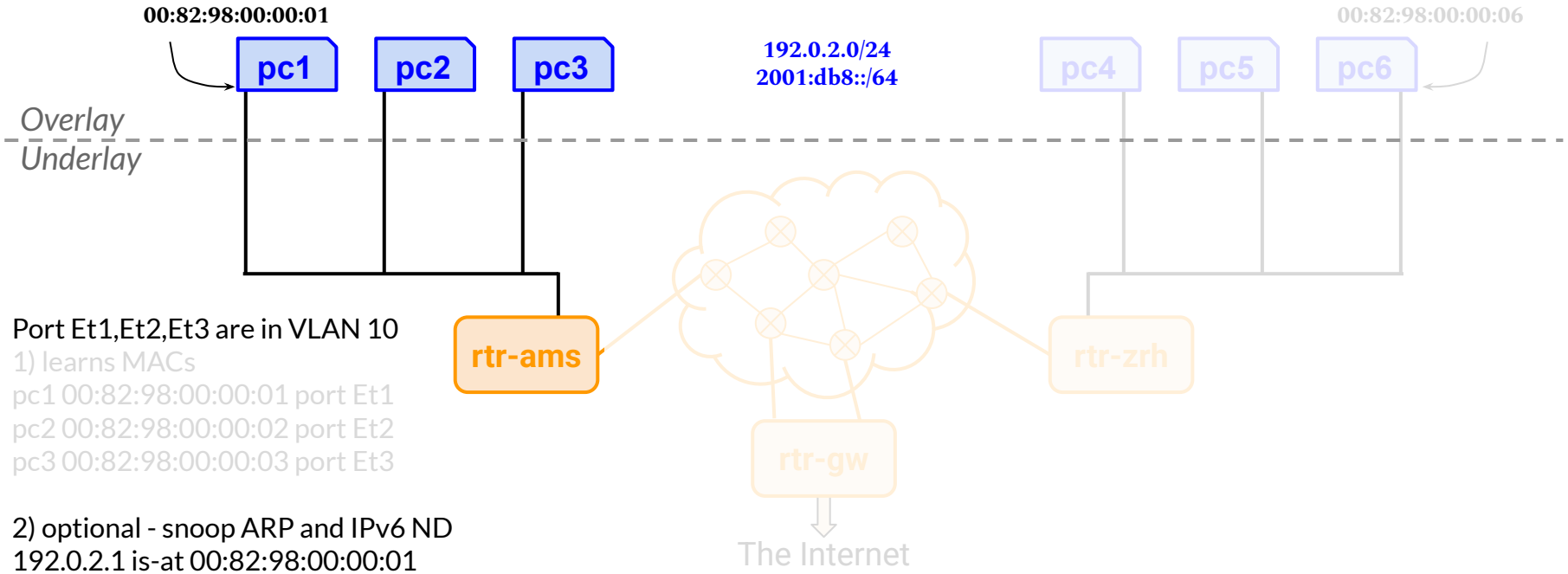


Port Et1,Et2,Et3 are in VLAN 10
1) learns MACs
pc1 00:82:98:00:00:01 port Et1
pc2 00:82:98:00:00:02 port Et2
pc3 00:82:98:00:00:03 port Et3

2) snoops ARP and IPv6 ND
192.0.2.1 is-at 00:82:98:00:00:01
2001:db8::2 is-at 00:82:98:00:00:02

Intro: eVPN

Step 1 - just learning



Port Et1,Et2,Et3 are in VLAN 10

1) learns MACs

pc1 00:82:98:00:00:01 port Et1

pc2 00:82:98:00:00:02 port Et2

pc3 00:82:98:00:00:03 port Et3

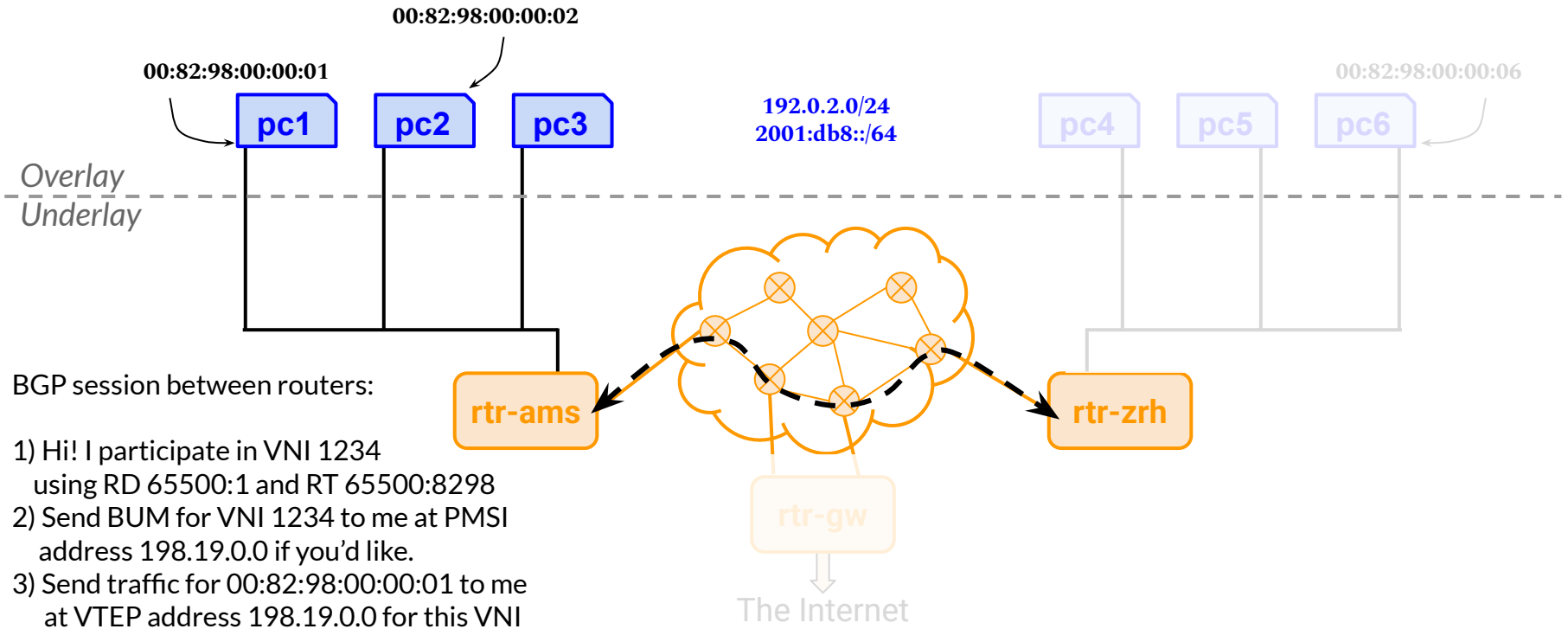
2) optional - snoop ARP and IPv6 ND

192.0.2.1 is-at 00:82:98:00:00:01

2001:db8::2 is-at 00:82:98:00:00:02

Intro: eVPN

Step 2 - BGP session

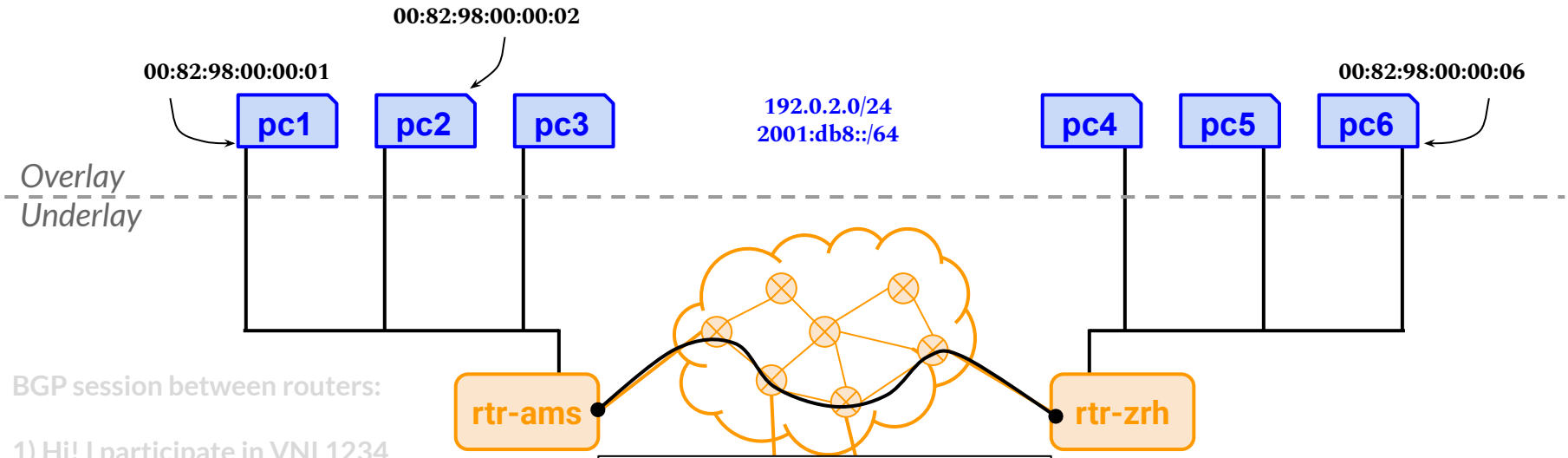


BGP session between routers:

1) Hi! I participate in VNI 1234
   using RD 65500:1 and RT 65500:8298
2) Send BUM for VNI 1234 to me at PMSI
   address 198.19.0.0 if you'd like.
3) Send traffic for 00:82:98:00:00:01 to me
   at VTEP address 198.19.0.0 for this VNI
4) By the way MAC 00:82:98:00:00:02
   belongs to IPv6 2001:db8::2

Intro: eVPN

Step 3 - BGP eVPN Table



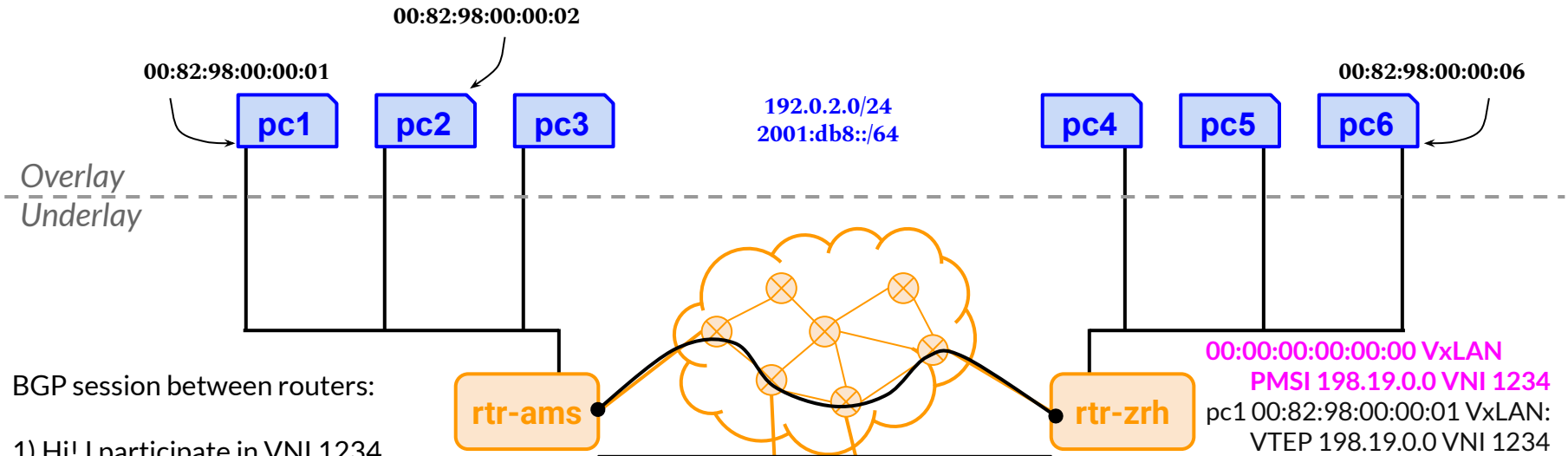
BGP session between routers:

1) Hi! I participate in VNI 1234 using RD 65500:1 and RT 65500:8298
2) Send BUM for VNI 1234 to me at PMSI address 198.19.0.0 if you'd like.
3) Send traffic for 00:82:98:00:00:01 to me at VTEP address 198.19.0.0 for this VNI
4) By the way MAC 00:82:98:00:00:02 belongs to IPv6 2001:db8::2

BGP Session (RD 65500:1)
00:00:00:00:00:00
via PMSI 198.19.0.0 VNI 1234
via PMSI 198.19.0.1 VNI 1234
00:82:98:00:00:01
via VTEP 198.19.0.0 VNI 1234
...
00:82:98:00:00:06
via VTEP 198.19.0.1 VNI 1234

Intro: eVPN

Step 4 - VTEP and PMSI



BGP session between routers:

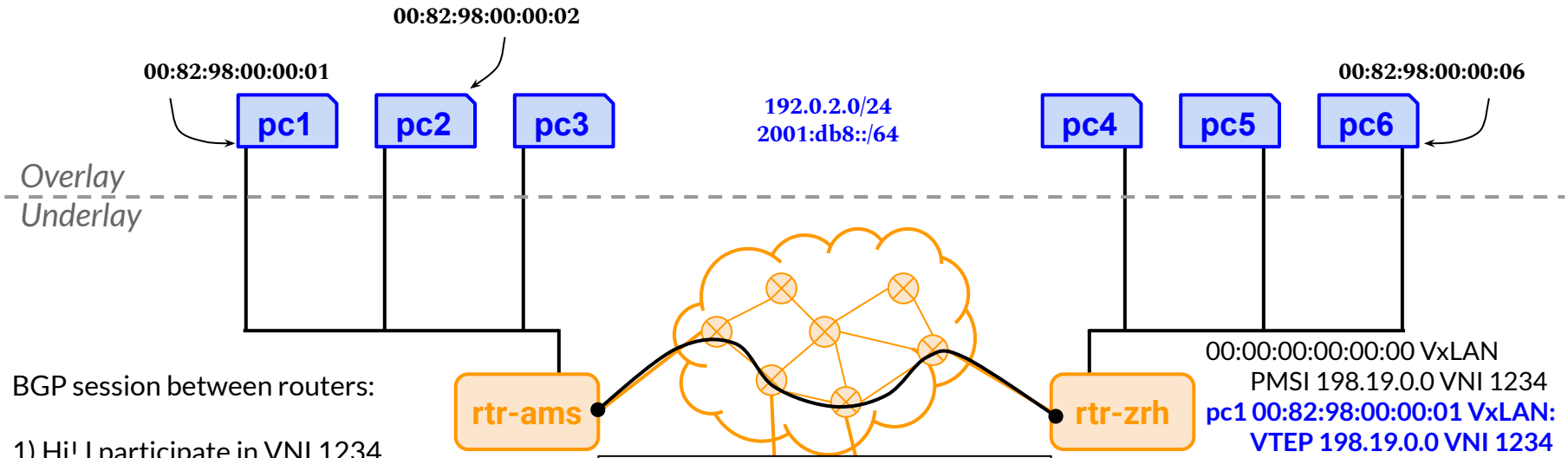
1) Hi! I participate in VNI 1234
   using RD 65500:1 and RT 65500:8298
2) Send BUM for VNI 1234 to me at PMSI
   address 198.19.0.0 using VxLAN.
3) Send traffic for 00:82:98:00:00:01 to me
   at VTEP 198.19.0.0 for this VNI
4) By the way MAC 00:82:98:00:00:02
   belongs to IPv6 2001:db8::2

BGP Session (RD 65500:1)
00:00:00:00:00:00
via PMSI 198.19.0.0 VNI 1234
via PMSI 198.19.0.1 VNI 1234
00:82:98:00:00:01
via VTEP 198.19.0.0 VNI 1234
...
00:82:98:00:00:06
via VTEP 198.19.0.1 VNI 1234

00:00:00:00:00:00 VxLAN
PMSI 198.19.0.0 VNI 1234
pc1 00:82:98:00:00:01 VxLAN:
VTEP 198.19.0.0 VNI 1234
pc2 00:82:98:00:00:02 VxLAN:
VTEP 198.19.0.0 VNI 1234
pc3 00:82:98:00:00:03 VxLAN:
VTEP 198.19.0.0 VNI 1234
pc4 00:82:98:00:00:04 port Et1
pc5 00:82:98:00:00:05 port Et2
pc6 00:82:98:00:00:06 port Et3

Intro: eVPN

Step 4 - VTEP and PMSI



BGP session between routers:

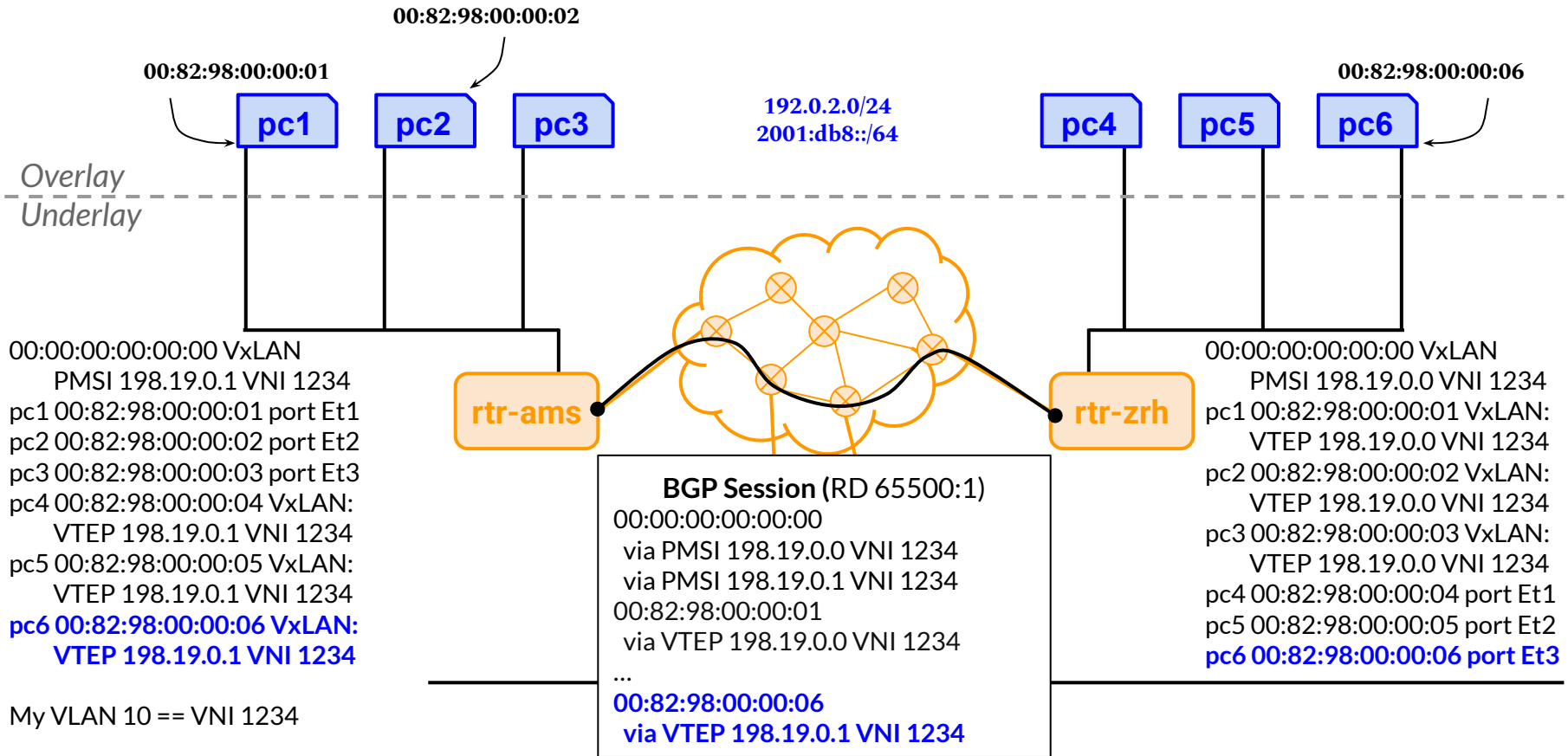
1) Hi! I participate in VNI 1234 using RD 65500:1 and RT 65500:8298
2) Send BUM for VNI 1234 to me at PMSI address 198.19.0.0 using VxLAN.
3) Send traffic for 00:82:98:00:00:01 to me at VTEP 198.19.0.0 for this VNI
4) By the way MAC 00:82:98:00:00:02 belongs to IPv6 2001:db8::2

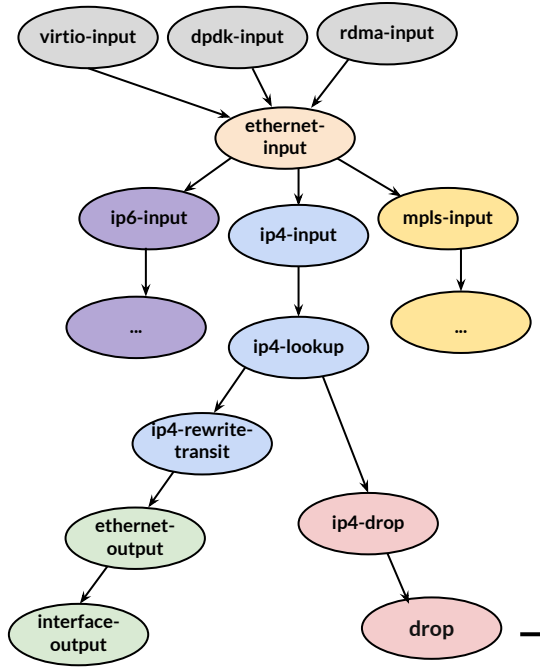
BGP Session (RD 65500:1)
00:00:00:00:00:00
via PMSI 198.19.0.0 VNI 1234
via PMSI 198.19.0.1 VNI 1234
00:82:98:00:00:01
via VTEP 198.19.0.0 VNI 1234
...
00:82:98:00:00:06
via VTEP 198.19.0.1 VNI 1234

00:00:00:00:00:00 VxLAN PMSI 198.19.0.0 VNI 1234
pc1 00:82:98:00:00:01 VxLAN: VTEP 198.19.0.0 VNI 1234
pc2 00:82:98:00:00:02 VxLAN: VTEP 198.19.0.0 VNI 1234
pc3 00:82:98:00:00:03 VxLAN: VTEP 198.19.0.0 VNI 1234
pc4 00:82:98:00:00:04 port Et1
pc5 00:82:98:00:00:05 port Et2
pc6 00:82:98:00:00:06 port Et3

Intro: eVPN

Step 5 - End to End





Intro: Vector Packet Processing

VPP [ref] is an open source dataplane that can:

- provide *very* fast networking
- using DPDK, RDMA, VirtIO, VMXNet3, AVF, …
- easily exceeds 100Mpps+ and 100Gbps+
- on commodity x86_64 / amd64 hardware!

Quick Overview:

- [ZANOG'25] 1Tbps with VPP
- [NLNOG'24] 100Mpps+ BGP/OSPF with one IPv4 address
- [GRNOG'24] VPP with MPLS
- [NetBCN'23] AS8298: Deep Dive

VPP: eVPN building blocks

1) Bridge Domain

2) Bridge Virtual Interface

3) VxLAN Tunnels

4) Split Horizon Group

5) Disable Learn & Flood

```
vpp# create bridge-domain 1000 learn 1 uu-flood 0
vpp# set interface l2 bridge HundredGigabitEthernet3/0/0 1000
vpp# set interface l2 bridge HundredGigabitEthernet3/0/1 1000
```

IPng Article #1

VPP: eVPN building blocks

1) Bridge Domain

2) Bridge Virtual Interface

3) VxLAN Tunnels

4) Split Horizon Group

5) Disable Learn & Flood

```
vpp# create bridge-domain 1000 learn 1 uu-flood 0
vpp# set interface l2 bridge HundredGigabitEthernet3/0/0 1000
vpp# set interface l2 bridge HundredGigabitEthernet3/0/1 1000

vpp# bvi create instance 1000 mac b8:ce:f6:82:98:00
vpp# set interface l2 bridge bvi1000 1000 bvi
vpp# set interface ip address bvi1000 192.0.2.1/24
vpp# set interface ip address bvi1000 2001:db8::1/64
```

IPng Article #1

VPP: eVPN building blocks

1) Bridge Domain

2) Bridge Virtual Interface

3) VxLAN Tunnels

4) Split Horizon Group

5) Disable Learn & Flood

```
vpp# create bridge-domain 1000 learn 1 uu-flood 0
vpp# set interface l2 bridge HundredGigabitEthernet3/0/0 1000
vpp# set interface l2 bridge HundredGigabitEthernet3/0/1 1000

vpp# bvi create instance 1000 mac b8:ce:f6:82:98:00
vpp# set interface l2 bridge bvi1000 1000 bvi
vpp# set interface ip address bvi1000 192.0.2.1/24
vpp# set interface ip address bvi1000 2001:db8::1/64

vpp# create vxlan tunnel instance 0 src 198.19.0.64 dst 198.19.0.13 vni 1000
vpp# create vxlan tunnel instance 1 src 198.19.0.64 dst 198.19.1.16 vni 1000
```

IPng Article #1

VPP: eVPN building blocks

1) Bridge Domain

2) Bridge Virtual Interface

3) VxLAN Tunnels

4) Split Horizon Group

5) Disable Learn & Flood

```
vpp# create bridge-domain 1000 learn 1 uu-flood 0
vpp# set interface l2 bridge HundredGigabitEthernet3/0/0 1000
vpp# set interface l2 bridge HundredGigabitEthernet3/0/1 1000

vpp# bvi create instance 1000 mac b8:ce:f6:82:98:00
vpp# set interface l2 bridge bvi1000 1000 bvi
vpp# set interface ip address bvi1000 192.0.2.1/24
vpp# set interface ip address bvi1000 2001:db8::1/64

vpp# create vxlan tunnel instance 0 src 198.19.0.64 dst 198.19.0.13 vni 1000
vpp# create vxlan tunnel instance 1 src 198.19.0.64 dst 198.19.1.16 vni 1000


vpp# set interface l2 bridge vxlan_tunnel0 1000 1
vpp# set interface l2 bridge vxlan_tunnel1 1000 1
```

IPng Article #1

VPP: eVPN building blocks

1) Bridge Domain

2) Bridge Virtual Interface

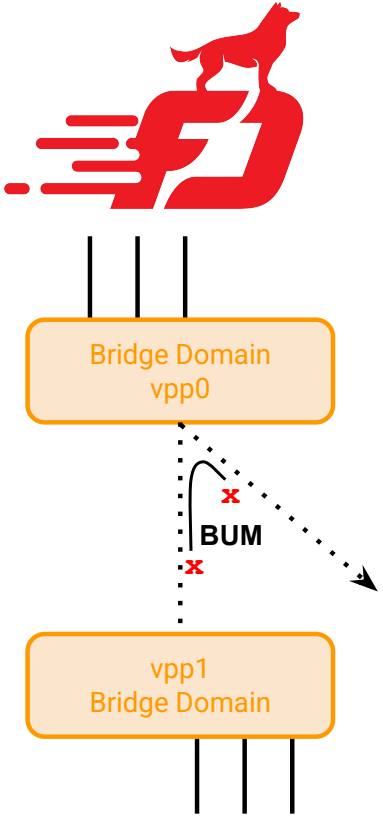
3) VxLAN Tunnels

4) Split Horizon Group

5) Disable Learn & Flood

```
vpp# create bridge-domain 1000 learn 1 uu-flood 0
vpp# set interface l2 bridge HundredGigabitEthernet3/0/0 1000
vpp# set interface l2 bridge HundredGigabitEthernet3/0/1 1000

vpp# bvi create instance 1000 mac b8:ce:f6:82:98:00
vpp# set interface l2 bridge bvi1000 1000 bvi
vpp# set interface ip address bvi1000 192.0.2.1/24
vpp# set interface ip address bvi1000 2001:db8::1/64

vpp# create vxlan tunnel instance 0 src 198.19.0.64 dst 198.19.0.13 vni 1000
vpp# create vxlan tunnel instance 1 src 198.19.0.64 dst 198.19.1.16 vni 1000


vpp# set interface l2 bridge vxlan_tunnel0 1000 1
vpp# set interface l2 bridge vxlan_tunnel1 1000 1

vpp# set interface l2 learn vxlan_tunnel0 disable
vpp# set interface l2 flood vxlan_tunnel0 disable
vpp# set interface l2 learn vxlan_tunnel1 disable
vpp# set interface l2 flood vxlan_tunnel1 disable
```

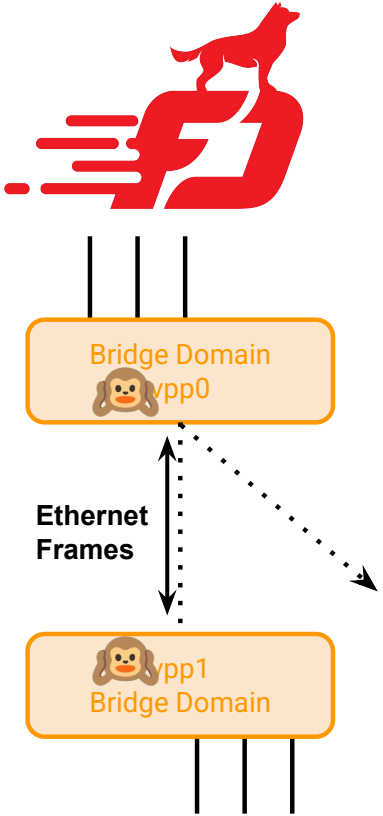
IPng Article #1



VPP: Details on SHG, Learn, Flood

Split Horizon Group

- BUM packet received on an interface with SHG of N?
  - x won't flood to any other interface also in SHG N



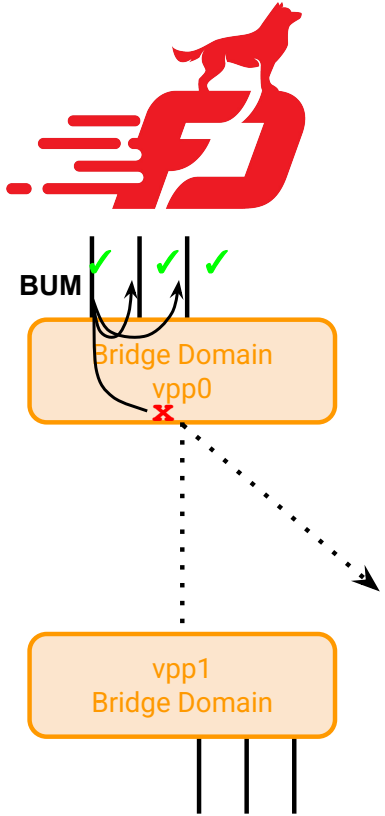
VPP: Details on SHG, Learn, Flood

Split Horizon Group

- BUM packet received on an interface with SHG of N?
  - x won't flood to any other interface also in SHG N

Disable Learning

- Any packet received on a *tunnel* interface?
  - 🙊 do not automatically learn and store in L2FIB



VPP: Details on SHG, Learn, Flood

Split Horizon Group

- BUM packet received on an interface with SHG of N?
  - x won't flood to any other interface also in SHG N

Disable Learning

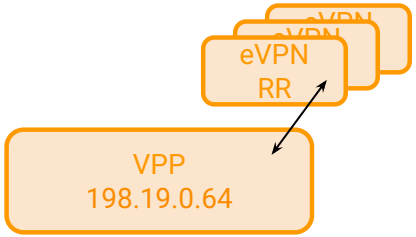
- Any packet received on a *tunnel* interface?
  - 🙊 do not automatically learn and store in L2FIB

Disable Flooding

- BUM packet received on a local ethernet interface?
  - ✓ flood to other local interface (normal)
  - x do not flood to *tunnel* interfaces

Act 2

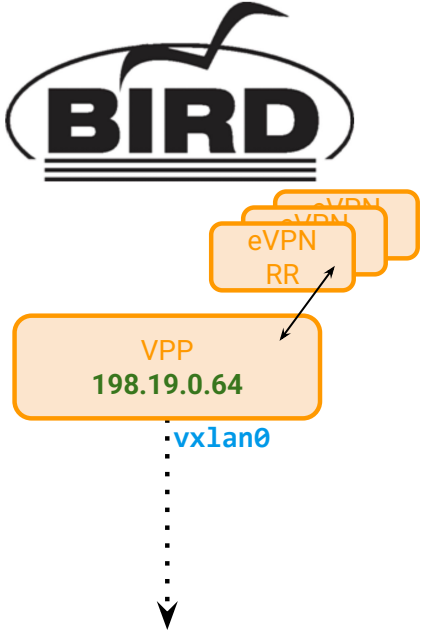
git.ipng.ch/ipng/bird-vppevpn



Bird: using eVPN

Step 1 - iBGP route-reflector client to exchange eVPN NLRI

```
evpn table evpntab;

template bgp T_EVPN {
  local as 65500; neighbor as 65500; source address 198.19.0.64;
  evpn { table evpntab; import all; export all; };
};

protocol bgp evpn0_nlams0 from T_EVPN { neighbor 198.19.4.101; }
protocol bgp evpn0_chplo0 from T_EVPN { neighbor 198.19.4.174; }
protocol bgp evpn0_chbtl0 from T_EVPN { neighbor 198.19.6.232; }

pim@dpu0-ddln0:~$ birdc show route count table evpntab
BIRD 2.19.1+branch.vppevpn.ef91372512dc ready.
4803 of 4803 routes for 1601 networks in table evpntab
```



Bird: using eVPN

Step 2 - Create an eVPN instance

```
eth table etab_test;

protocol evpn evpn_test {
  eth  { table etab_test; };
  evpn { table evpntab; import all; export all; };
  rd 65500:1000;
  route target (rt, 65500, 1000);
  encapsulation vxlan { tunnel device "vxlan0"; router address 198.19.0.64; };
  vni 1000;
};
```

Results:

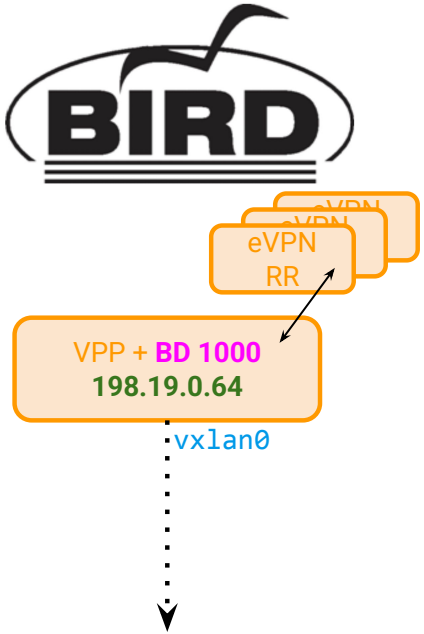
1) eVPN instance `evpn_test` uses RT (65500,1000) and RD (65500:1000)
2) MAC information from table `etab_test` syncs with `evpntab`, namespaced by RD
2) PMSI uses interface `vxlan0` and announces address 198.19.0.64
3) VxLAN uses VNI 1000 source 198.19.0.64 to reach other VTEPs in `evpn_test`

My offer: Bird vppevpn *protocol*

Step 3 - Implement custom vppevpn bird protocol:

1) VPP: create Bridge Domain
2) VPP: create BVI + LinuxCP pair
3) VPP: receive IMETs from Bird and create VxLAN tunnels
4) VPP: receive MACs from Bird and program L2FIB
5) Bird: announce IMET for this VPP router into BGP
6) Bird: announce MACs from VPP into BGP
7) Bird: scan VPP and etab and reconcile every N secs

(ask me for my source code, patch on 2.19.1)



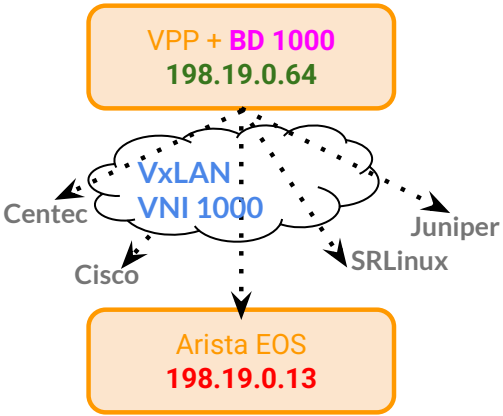
My offer: Bird vppevpn *protocol*

Step 4 - Create a VPPEVPN instance

```
eth table etab_test;

protocol vppevpn vppevpn_test {
  eth { table etab_test; import all; export all; };
  vxlan ipv4 src 198.19.0.64;
  vxlan src port 4789;
  vxlan dst port 4789;
  bridge domain 1000;
  scan time 300;
};
```

→ Bird will start syncing the eVPN to/from this bridge-domain

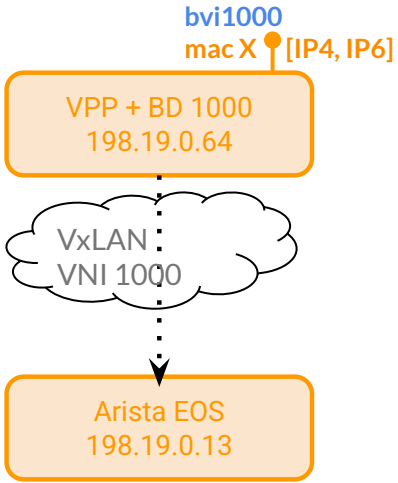
My offer: Bird vppevpn *protocol*




```
pim@dpu0-ddln0:~$ birdc show proto all vppevpn_test
BIRD 2.19.1+branch.vppevpn.ef91372512dc ready.
Name         Proto      Table      State  Since         Info
vppevpn_test VPPEVPN    etab_test  up     2026-04-14 13:04:06
  Channel eth
    State:          UP
    Table:          etab_test
    Preference:     0
    Input filter:   ACCEPT
    Output filter:  ACCEPT
    Routes:         1 imported, 32 exported, 1 preferred
    Route change stats:     received   rejected   filtered    ignored   accepted
      Import updates:              2          0          0          0          2
      Import withdraws:            1          0        ---          0          1
      Export updates:             34          2          0        ---         32
      Export withdraws:            1        ---        ---        ---          0
  Bridge domain:    1000
  VxLAN:
    IPv4 src:       [198.19.0.64]:4789
    Tunnel:         [198.19.1.16]:4789 vni=1000 sw_if_index=22 imet=1 refcount=2
    Tunnel:         [198.19.0.10]:4789 vni=1000 sw_if_index=24 imet=1 refcount=1
    Tunnel:         [198.19.0.13]:4789 vni=1000 sw_if_index=18 imet=1 refcount=14
    ...
```

IPng Article #2

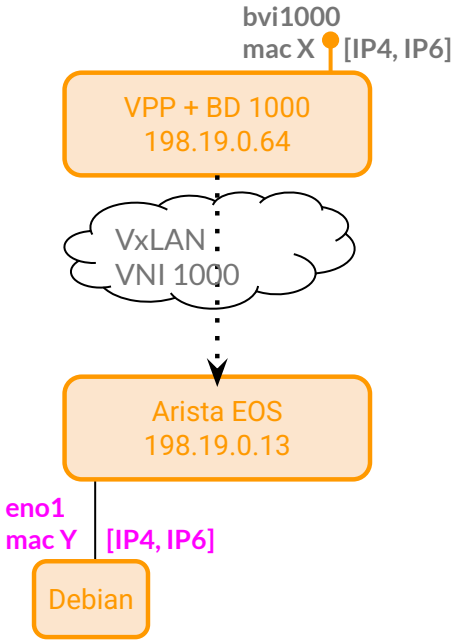
My offer: Bird vppevpn *protocol*




```
pim@dpu0-ddln0:~$ vppctl show lcp phy bvi1000
itf-pair: [7] bvi1000 tap4100 bvi-test 9 type tap netns dataplane

pim@dpu0-ddln0:~$ ip link show bvi-test
9: bvi-test: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state ...
    link/ether b8:ce:f6:82:98:00 brd ff:ff:ff:ff:ff:ff

pim@dpu0-ddln0:~$ vppctl show bridge-domain 1000 detail
           Interface           If-idx ISN  SHG  BVI  TxFlood        VLAN-Tag-Rewrite
            bvi1000              20    1    0    *      *                none
          vxlan_tunnel3          18    1    1    -      *                none
          vxlan_tunnel4          19    1    1    -      *                none
          vxlan_tunnel5          22    1    1    -      *                none
          vxlan_tunnel6          23    1    1    -      *                none
          vxlan_tunnel7          24    1    1    -      *                none
pim@dpu0-ddln0:~$ vppctl show l2fib bd_id 1000
    Mac-Address     BD-Idx If-Idx BSN-ISN Age(min) static filter bvi  Interface-Name
 b8:ce:f6:82:98:00    2      20     0/0       no      *      -     *       bvi1000
 ee:31:c8:09:e6:27    2      18     0/0       no      *      -     -    vxlan_tunnel3
 3c:ec:ef:9a:b9:f9    2      18     0/0       no      *      -     -    vxlan_tunnel3
 b8:cb:29:9c:40:44    2      18     0/0       no      *      -     -    vxlan_tunnel3
 92:90:e6:a7:d9:a3    2      19     0/0       no      *      -     -    vxlan_tunnel4
 ...
```

[IPng Article #2]

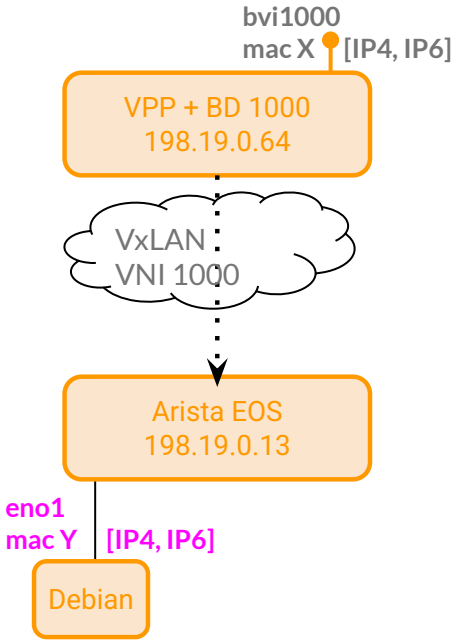
My offer: Bird vppevpn *protocol*




```
pim@dpu0-ddln0:~$ sudo ip addr add 192.0.2.254/24 dev bvi-test
pim@dpu0-ddln0:~$ sudo ip addr add 2001:db8::fe/64 dev bvi-test
pim@dpu0-ddln0:~$ ip nei | grep 192.0.2.1
192.0.2.1 dev bvi-test lladdr ee:31:c8:09:e6:27 REACHABLE
pim@dpu0-ddln0:~$ birdc show route all table evpntab where net.mac=ee:31:c8:09:e6:27
...
evpn mac 65500:1000 0 ee:31:c8:09:e6:27 * unicast [evpn0_nlams0 2026-04-12 13:45:43
from 198.19.4.101] * (100/42) [i]
        via 198.19.2.52 on nsw0-ddln0 mpls 1000
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path:
        BGP.next_hop: 198.19.0.13
        BGP.local_pref: 100
        BGP.originator_id: 198.19.0.13
        BGP.cluster_list: 198.19.4.101
        BGP.ext_community: (rt, 65500, 1000) (generic, 0x30c0000, 0x8)
        BGP.mpls_label_stack: 1000

pim@dpu0-ddln0:~$ ping 192.0.2.1
PING 192.0.2.1 (192.0.2.1) 56(84) bytes of data.
64 bytes from 192.0.2.1: icmp_seq=1 ttl=64 time=0.159 ms
...
```

IPng Article #2

My offer: Bird vppevpn *protocol*



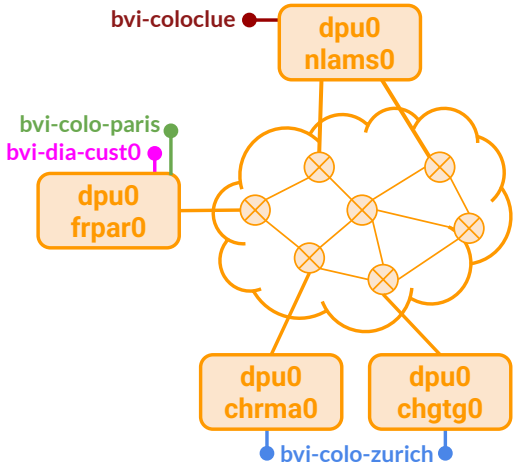

```
pim@dpu0-ddln0:~$ sudo ip addr add 192.0.2.254/24 dev bvi-test
pim@dpu0-ddln0:~$ sudo ip addr add 2001:db8::fe/64 dev bvi-test
pim@dpu0-ddln0:~$ ip nei | grep 192.0.2.1
192.0.2.1 dev bvi-test lladdr ee:31:c8:09:e6:27 REACHABLE
pim@dpu0-ddln0:~$ birdc show route all table evpntab where net.mac=ee:31:c8:09:e6:27
...
evpn mac 65500:1000 0 ee:31:c8:09:e6:27 * unicast [evpn0_nlams0 2026-04-12 13:45:43
from 198.19.4.101] * (100/42) [i]
        via 198.19.2.52 on nsw0-ddln0 mpls 1000
        Type: BGP univ
        BGP.origin: IGP
        BGP.as_path:
        BGP.next_hop: 198.19.0.13
        BGP.local_pref: 100
        BGP.originator_id: 198.19.0.13
        BGP.cluster_list: 198.19.4.101
        BGP.ext_community: (rt, 65500, 1000) (generic, 0x3 0c 0000, 0x0008)
        BGP.mpls_label_stack: 1000

pim@dpu0-ddln0:~$ ping 192.0.2.1
PING 192.0.2.1 (192.0.2.1) 56(84) bytes of data.
64 bytes from 192.0.2.1: icmp_seq=1 ttl=64 time=0.159 ms
...
```

```
03      type    = opaque extended community
0c      sub     = Encapsulation (RFC 5512)
00 00 00 00     = reserved
00 08           = tunnel type (VXLAN)
```

IPng Article #2

Act 3

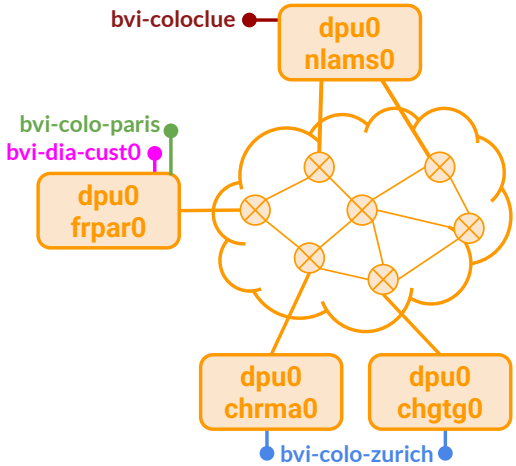
git.ipng.ch/ipng/vpp-evpn



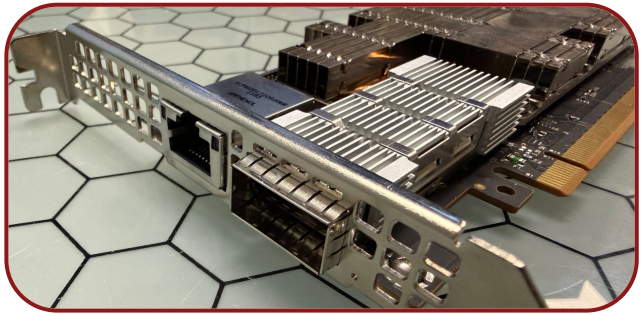
Imagine a growing eVPN network

Requirements and Scale

- O(100) routers
- O(1000) L3 subnets (IPv4/IPv6 networks)
- O(10000) P2MP L2 segments
- Ability to move segments and subnets around for maintenance
- Use BGP signalling for eVPN, OSPF/BGP for L3
- Redundancy, automated failover
- Linear capacity scaling: 40Mpps/100Gbps per node
- Fully Open Source
- Reusing VPP's features:
  - Maglev loadbalancing
  - IPSec, Wireguard, MACSEC, ...



My offer: vpp-evpn



Hardware Solution

- Use Mellanox Bluefield DPUs
- Run Ubuntu / Debian on them
- Run VPP on them! 🥰

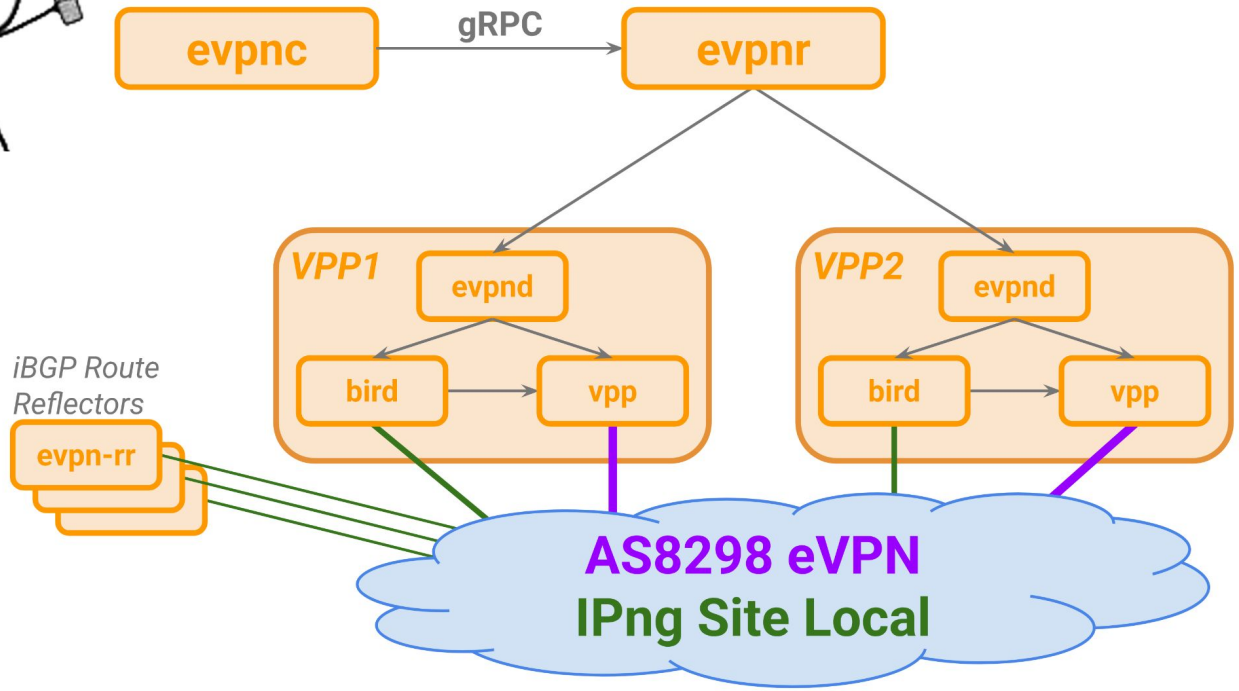
Controlplane for VPP+Bird

- Logically add / remove DPUs in an eVPN
- Create BVIs with vMAC and IPv4/IPv6
- Move BVIs between DPUs

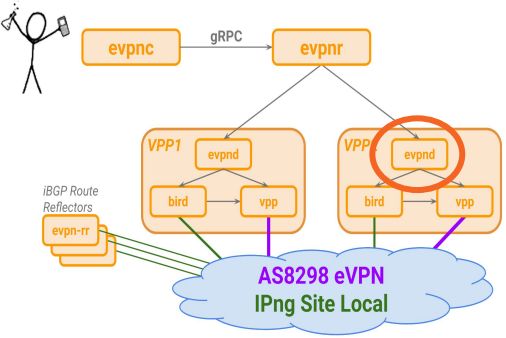
Managementplane for a global operations

- RPC endpoint, CLI, WebUI
- Automated healthchecking + failover
- Observability with Grafana/Prometheus

vpp-evpn: components



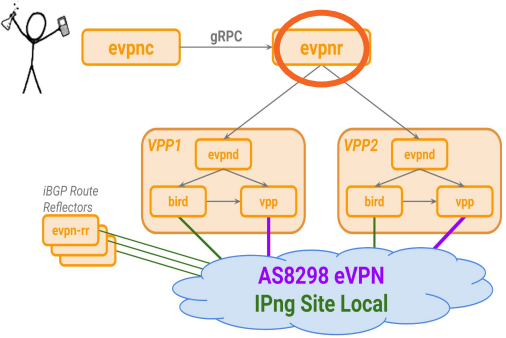
1) Daemon on DPU
   - Bird 2.19+
   - VPP 26.06+
2) Central Registry
3) gRPC Client
4) SolidJS WebUI



vpp-evpn: epvnd

eVPN Daemon, runs inside of DPU

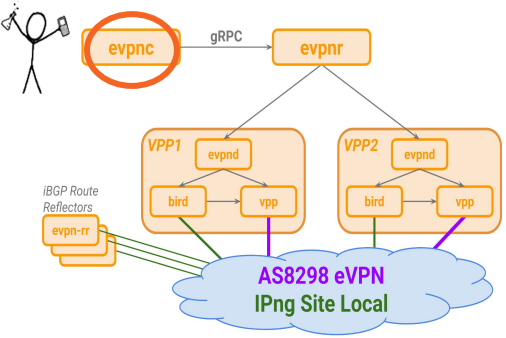
- Connects to a central registry, streams events, accepts commands
- Maintains `/etc/vpp-evpn/bird.d/*.conf` with vppevpn protocols
- Creates BVI in VPP w/ ephemeral MAC or vMAC
- Sets/Clears IPv4/IPv6 addresses on BVI
- Healthchecker (think CARP/HSRP/VRRP)
  - Sends multicast IPv6 probes on eVPN segment
  - Listens to mcast from other DPUs
  - Sends unicast to internet
- Is restart-safe: can tolerate loss of registry (and loss of self)
- Is reentrant: reconfigures Bird or VPP if they crash
- Provides gRPC endpoint and Prometheus `/metrics`



vpp-evpn: epvnr

eVPN Registry, runs on IPng Site Local in docker

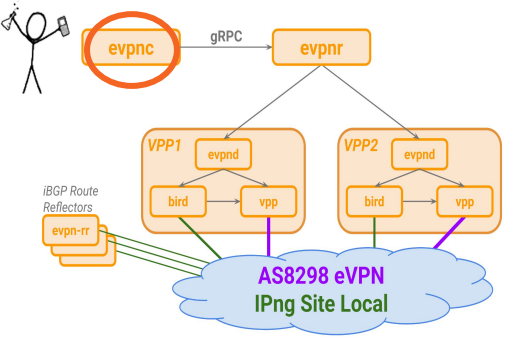
- Maintains connections from all DPUs
- Holds configuration of eVPNs
  - Configures GroupBVIs, MACs, MTU
  - Adds/Removes IPv4/IPv6 to GroupBVIs
- Adds/Removes DPUs eVPNs
- Promotes/Demotes DPUs to ‘Primary’ or ‘Standby’
  - Sequences failover/migration of GroupBVIs
- Is restart-safe: can tolerate loss of registry (and loss of self)
- Is reentrant: reconfigures `evpnd` if they crash
- Provides gRPC endpoint and Prometheus `/metrics`



vpp-evpn: epvnc

```
pim@dpu0-ddln0:~$ evpnc -server evpn-registry.net.ipng.ch:9095
evpn> create group test 65500 1000
```

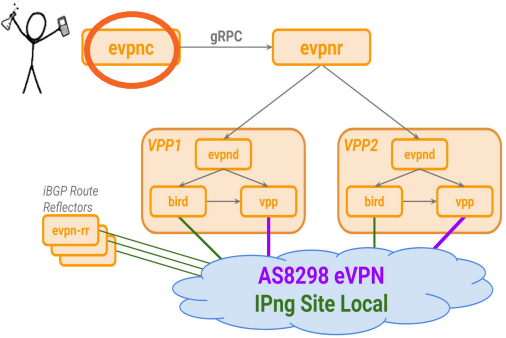
IPng Article #3



vpp-evpn: epvnc

```
pim@dpu0-ddln0:~$ evpnc -server evpn-registry.net.ipng.ch:9095
evpn> create group test 65500 1000
evpn> group test add dpu0-ddln0
evpn> group test add dpu0-chplo0
```

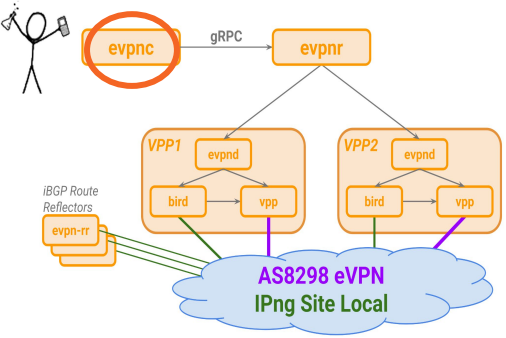
IPng Article #3



vpp-evpn: epvnc

```
pim@dpu0-ddln0:~$ evpnc -server evpn-registry.net.ipng.ch:9095
evpn> create group test 65500 1000
evpn> group test add dpu0-ddln0
evpn> group test add dpu0-chplo0
evpn> group test bvi create mac a4:fc:14:82:98:01 mtu 1500
evpn> group test bvi add address 172.16.0.32/24 address fec0::32/64
```

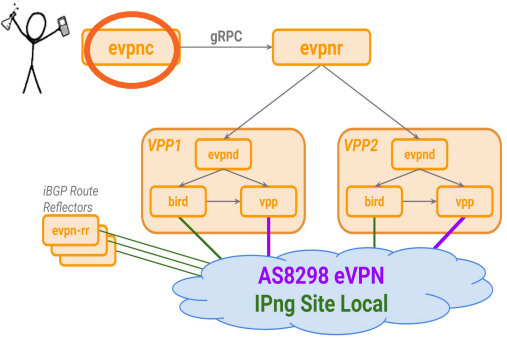
IPng Article #3



vpp-evpn: epvnc

```
pim@dpu0-ddln0:~$ evpnc -server evpn-registry.net.ipng.ch:9095
evpn> create group test 65500 1000
evpn> group test add dpu0-ddln0
evpn> group test add dpu0-chplo0
evpn> group test bvi create mac a4:fc:14:82:98:01 mtu 1500
evpn> group test bvi add address 172.16.0.32/24 address fec0::32/64
evpn> group test bvi set primary dpu0-ddln0
evpn> group test health psk add IPngLovesVPP
evpn> group test health enable failover enable
```

IPng Article #3



vpp-evpn: epvnc

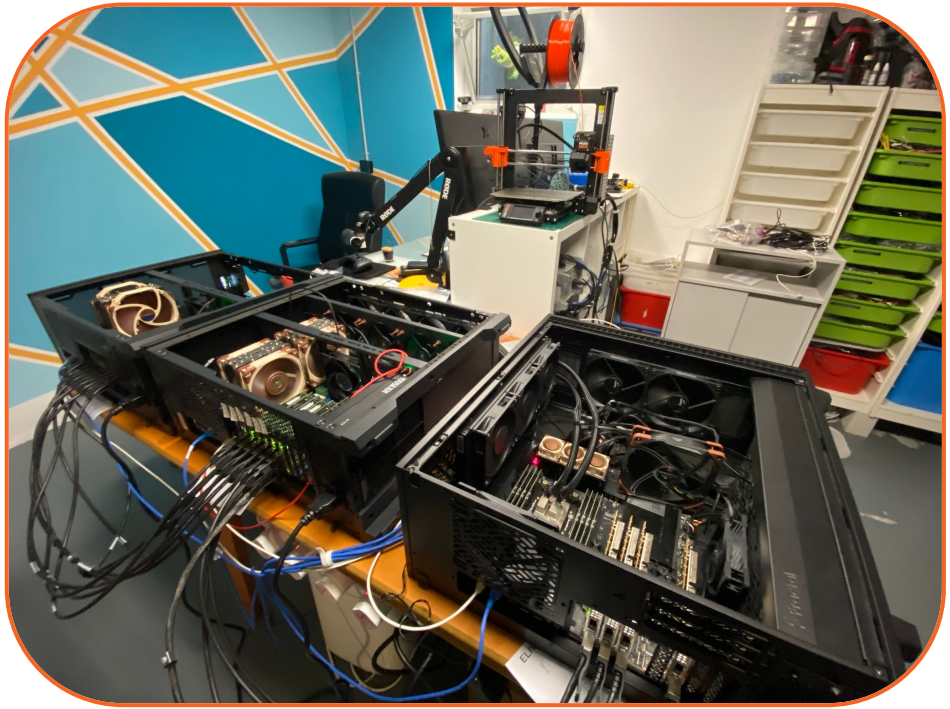
```
pim@dpu0-ddln0:~$ evpnc -server evpn-registry.net.ipng.ch:9095
evpn> create group test 65500 1000
evpn> group test add dpu0-ddln0
evpn> group test add dpu0-chplo0
evpn> group test bvi create mac a4:fc:14:82:98:01 mtu 1500
evpn> group test bvi add address 172.16.0.32/24 address fec0::32/64
evpn> group test bvi set primary dpu0-ddln0
evpn> group test health psk add IPngLovesVPP
evpn> group test health enable failover enable

evpn> show group test
group test  vni=1000 bdid=1000 mtu=1500 rd=65500:1000
            rt-import=(rt, 65500, 1000) rt-export=(rt, 65500, 1000)
  created=2026-08-31 13:03:53Z (7h 15m 59s)
  modified=2026-08-31 20:19:50Z (2s)
  group-bvi  ifname=bvi-test mac=b8:ce:f6:82:98:00
    addresses=172.16.0.254/24,fec0::32/64
    created=2026-08-31 13:04:43Z (7h 15m 9s)
    modified=2026-08-31 13:06:02Z (7h 13m 50s)
  health  enabled  healthy
    psk psk_15cf01d9  active  created=2026-08-31 20:19:40Z (12s)
  member dpu0-ddln0  connected primary since=2026-08-31 13:06:02Z (7h 13m 50s)
  member dpu0-chplo0  connected
```

IPng Article #3

Act 4: Demo time!


0:bash*

IPng Lab: Converting 1.8kW into 1.25Tbps VxLAN packets and then into heat =)

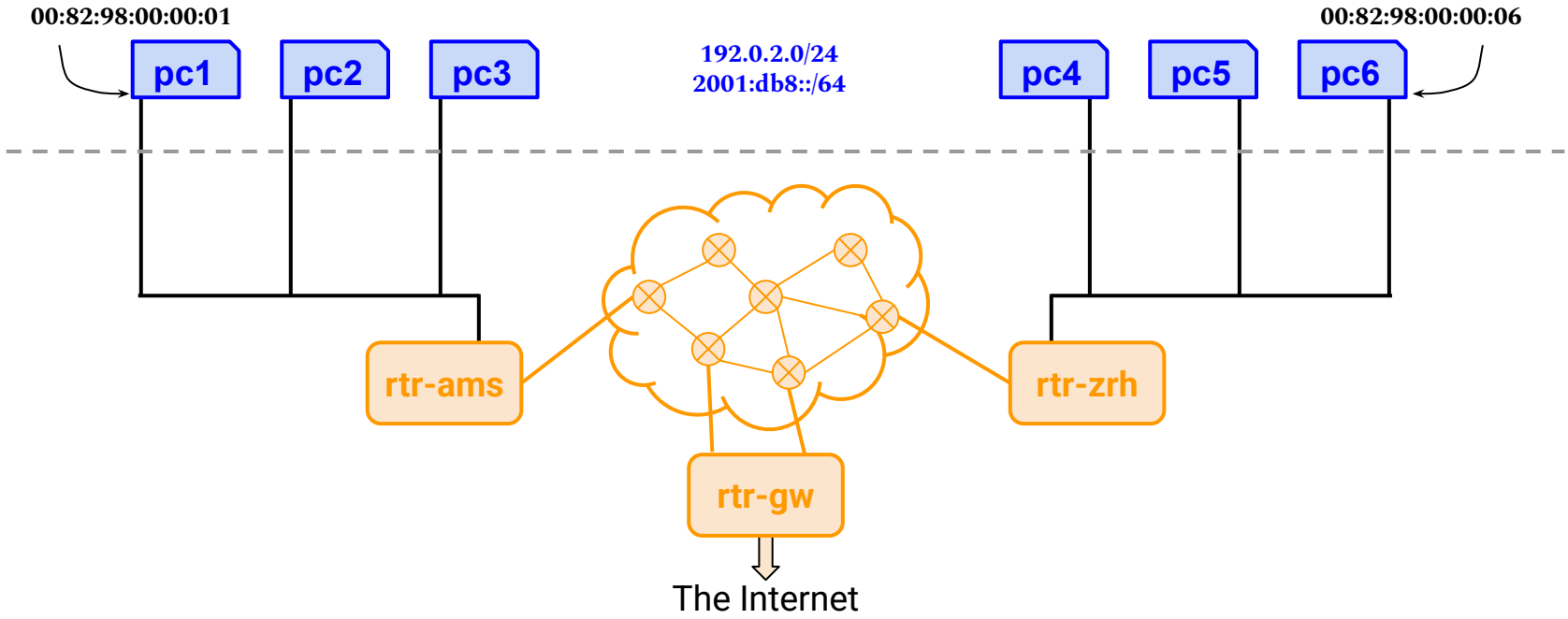


If you peer with IPng Networks, thanks!
If you don't: please peer with AS8298
<peering@ipng.ch>

Useful Resources

- VPP Mailinglist [vpp-dev@lists.fd.io]
- IPng Git Server [git.ipng.ch]
- Articles [ipng.ch]
- Mastodon [@IPngNetworks]

Intro: eVPN


00:82:98:00:00:01
00:82:98:00:00:06
pc1
pc2
pc3
192.0.2.0/24
2001:db8::/64
pc4
pc5
pc6
rtr-ams
rtr-zrh
rtr-gw
The Internet