Virtualized PE for BGP/MPLS L3-VPN using Open-Source Software

NANOG 74 (October 2018)

Bilal Anwer, Robert Bays, Vijay Gopalakrishnan, Bo Han, Dewi Morgan,

Patrick Ruddy, Aman Shaikh, Susheela Vaidya, Chengwei Wang and George Wilkie

Introduction

Objective

- Demonstrate feasibility of creating a BGP/MPLS L3-VPN vPE using open-source software

Motivation

- Use-case for AT&T's DANOS (Disaggregated Network OS)
- Why L3-VPN vPE from open-source software?

– L3-VPN
  - Allows creation of multiple layer-3 virtual networks on top of a shared service-provider network
  - Widely used service by enterprises

– vPE
  - Enabler VNF which acts as the ingress and egress for L3-VPN traffic in the service-provider network

– Open source software
  - Allows increased agility in providing new features while reducing the cost

Challenges

- Required functional and integration-related extensions to open-source components

Software Components of Open Source vPE

Control-plane

- FRR (5.1-dev, snapshot e8f9540) for OSPF, LDP and Zebra
- GoBGP (version 1.31.1 = version 1.31 + our enhancements)

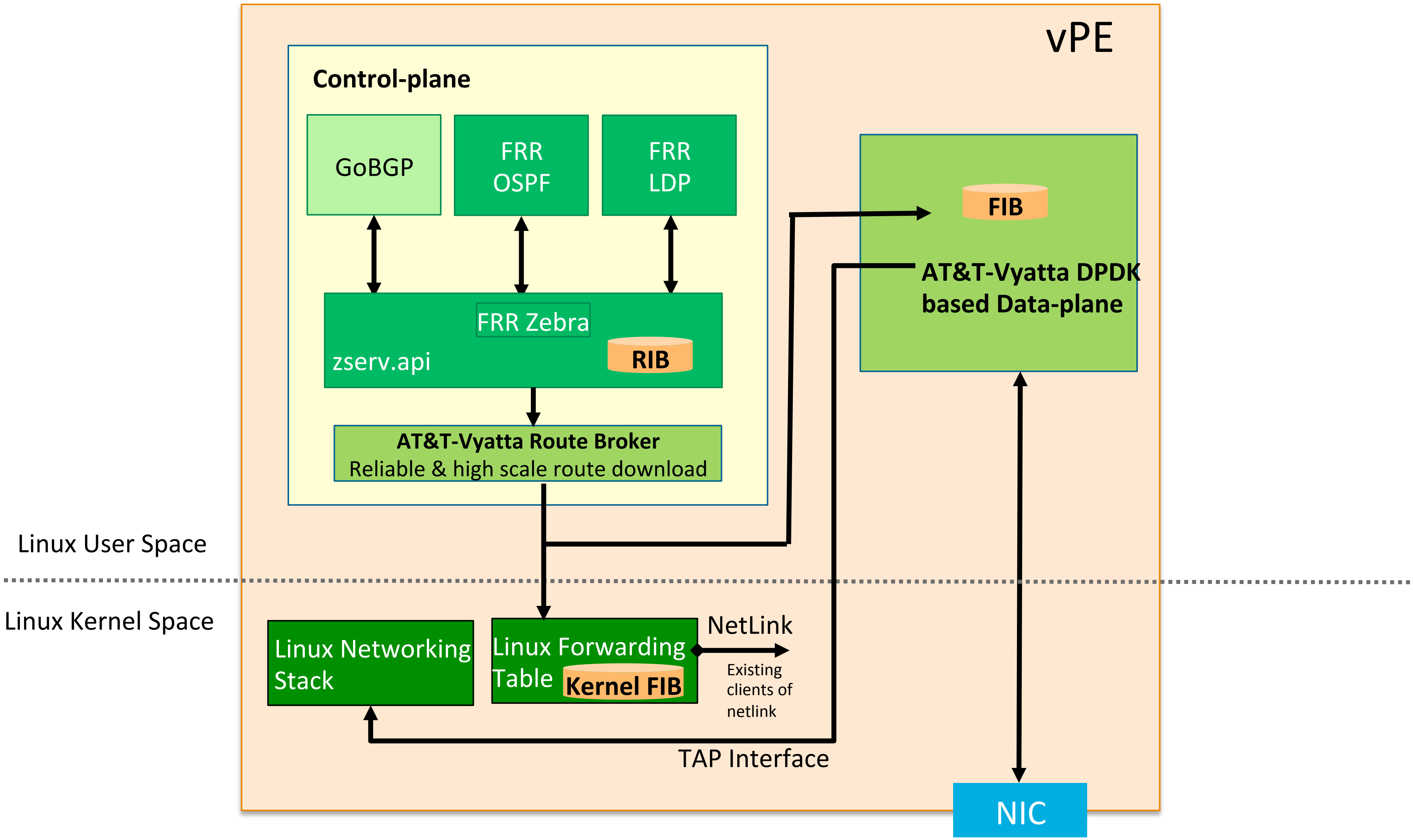
DANOS Use-Case

Data-plane

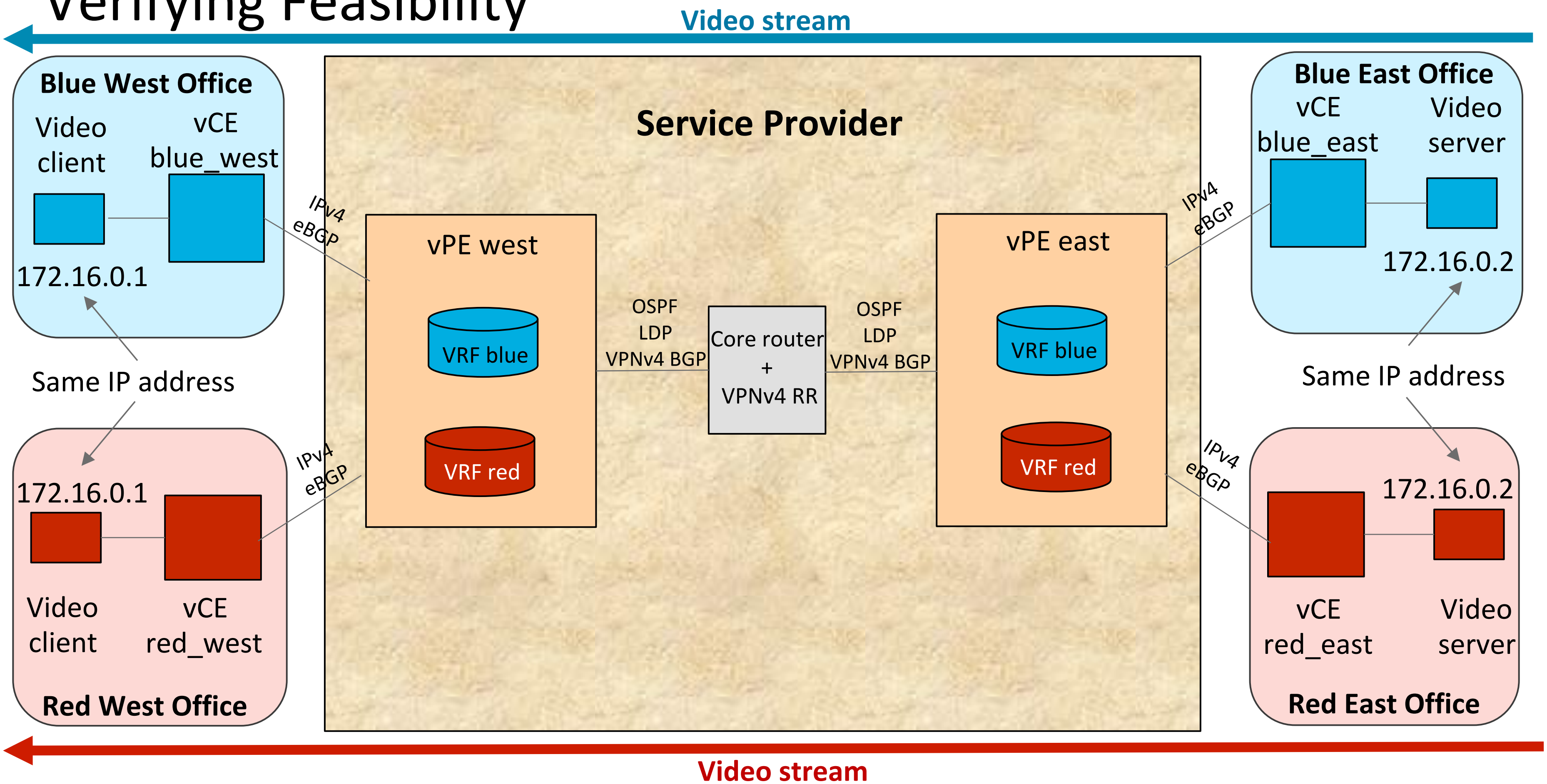
- AT&T-Vyatta's (DPDK-based) data-plane

- We also verified feasibility with …
  - Linux data-plane (kernel 4.14.4-mpls)
  - VPP data-plane (release 1801 + router plug-in with our enhancements which have been up-streamed)

Software Architecture of DANOS Open Source vPE

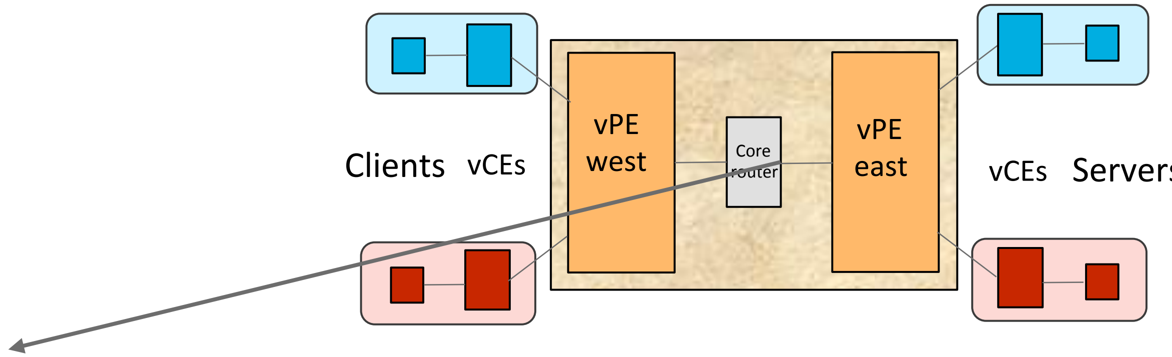


Verifying Feasibility



- Demonstrated feasibility by concurrently running two video streams
  - Keep video traffic separate despite same IP addresses being used by two customers
  - Each client/server has a static route pointing to its upstream CE
  - Each CE advertises appropriate prefix to the PE

Packet Capture at Core Router during Video Streaming




```
15:30:52.820824 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10559520:10560768, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820828 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10560768:10562016, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820919 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10562016:10563264, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820941 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10563264:10564512, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820956 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10564512:10565760, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820968 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10565760:10567008, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 1248: HTTP
15:30:52.820971 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], seq 10567008:10568000, ack 1, win 205, options [nop,nop,TS val 214625492 ec
r 214623791], length 992: HTTP
15:30:52.830446 MPLS (label 145, exp 0, [S], ttl 62) IP 10.31.3.10.47814 > 172.16.0.2.80: Flags [.], ack 10568000, win 0, options [nop,nop,TS val 214623794 ecr 214625491], length 0
15:30:53.041460 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], ack 1, win 205, options [nop,nop,TS val 214625548 ecr 214623794], length 0
15:30:53.044101 MPLS (label 145, exp 0, [S], ttl 62) IP 10.31.3.10.47814 > 172.16.0.2.80: Flags [.], ack 10568000, win 0, options [nop,nop,TS val 214623847 ecr 214625491], length 0
15:30:53.469700 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], ack 1, win 205, options [nop,nop,TS val 214625655 ecr 214623847], length 0
15:30:54.317493 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], ack 1, win 205, options [nop,nop,TS val 214625867 ecr 214623847], length 0
15:30:54.321034 MPLS (label 145, exp 0, [S], ttl 62) IP 10.31.3.10.47814 > 172.16.0.2.80: Flags [.], ack 10568000, win 0, options [nop,nop,TS val 214624166 ecr 214625491], length 0
15:30:55.172204 IP 10.31.11.10.646 > 224.0.0.2.646: LDP, Label-Space-ID: 192.168.0.2:0, pdu-length: 38
15:30:55.203298 IP 10.31.11.11 > 224.0.0.5: OSPFv2, Hello, length 48
15:30:55.246345 IP 10.31.11.11.646 > 224.0.0.2.646: LDP, Label-Space-ID: 192.168.0.3:0, pdu-length: 38
15:30:56.023629 MPLS (label 17, exp 0, ttl 64) (label 145, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.3.10.47814: Flags [.], ack 1, win 205, options [nop,nop,TS val 214626293 ecr 214624166], length 0
15:30:56.026492 MPLS (label 145, exp 0, [S], ttl 62) IP 10.31.3.10.47814 > 172.16.0.2.80: Flags [.], ack 10568000, win 0, options [nop,nop,TS val 214624593 ecr 214625491], length 0
15:30:56.372977 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], ack 1, win 204, options [nop,nop,TS val 214626372 ecr 214622990], length 0
15:30:56.376825 MPLS (label 144, exp 0, [S], ttl 62) IP 10.31.1.10.34498 > 172.16.0.2.80: Flags [.], ack 1, win 0, options [nop,nop,TS val 214624722 ecr 214622951], length 0
15:30:57.696543 MPLS (label 144, exp 0, [S], ttl 62) IP 10.31.1.10.34498 > 172.16.0.2.80: Flags [.], ack 1, win 123, options [nop,nop,TS val 214625052 ecr 214622951], length 0
15:30:57.698303 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], seq 1:1249, ack 1, win 204, options [nop,nop,TS val 214626703 ecr 214625505$
], length 1248: HTTP
15:30:57.698327 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], seq 1249:2497, ack 1, win 204, options [nop,nop,TS val 214626703 ecr 21462$
052], length 1248: HTTP
15:30:57.698339 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], seq 2497:3745, ack 1, win 204, options [nop,nop,TS val 214626703 ecr 21462$
052], length 1248: HTTP
15:30:57.698351 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], seq 3745:4993, ack 1, win 204, options [nop,nop,TS val 214626703 ecr 21462$
052], length 1248: HTTP
15:30:57.698356 MPLS (label 17, exp 0, ttl 64) (label 144, exp 0, [S], ttl 64) IP 172.16.0.2.80 > 10.31.1.10.34498: Flags [.], seq 4993:6241, ack 1, win 204, options [nop,nop,TS val 214626703 ecr 21462$
052], length 1248: HTTP
```

red server → red client

red client → red server

blue client → blue server

blue server → blue client

Configuration Details

Establish LSP (Label Switched Paths) between PEs

- Enable IP and MPLS forwarding
- Configure OSPF and LDP on service provider routers

Enable L3-VPN service

- Configure VRFs
- Configure eBGP sessions between PEs and CEs
- Configure iBGP sessions between PEs and route reflector

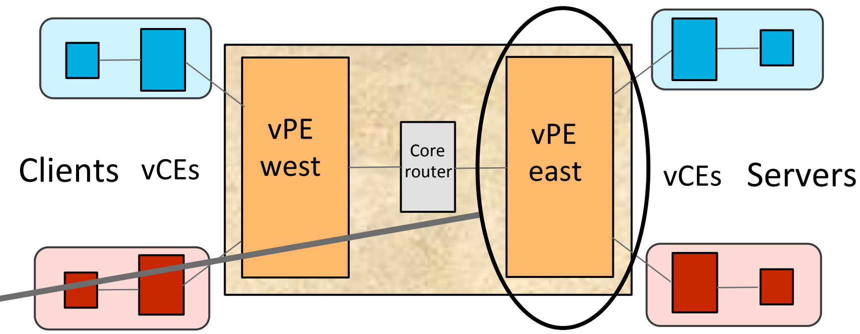
Note: We used AT&T/DANOS Yang Modules for configuring vPEs where possible, but show equivalent Linux, FRR and GoBGP commands in subsequent slides

Tale of Two Loopbacks

Configured two loopback addresses on vPEs and core router

- Loopback1
  - Used for IP traffic including control-plane traffic
- Loopback2
  - Used for MPLS traffic
    - Hence all traffic from VPN customers

vPE East: Configuring MPLS Forwarding



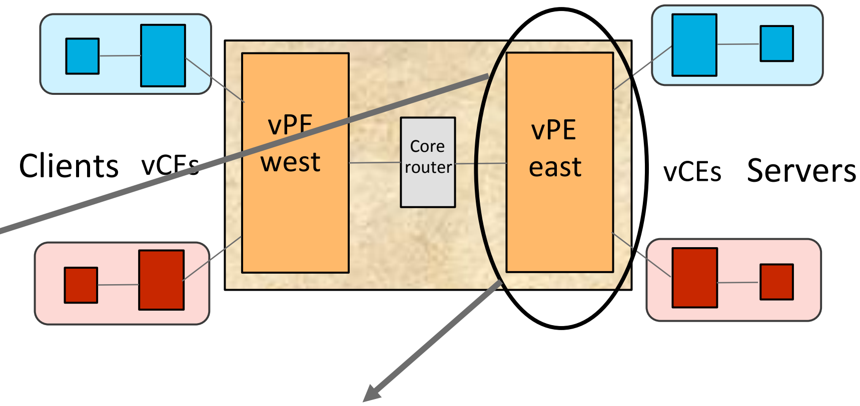

```
# Enable IPv4 forwarding
$ sudo sysctl -w net.ipv4.ip_forward=1

# Load MPLS modules
$ sudo modprobe mpls_router
$ sudo modprobe mpls_iptunnel

# Enable MPLS forwarding on the interface facing the core router
$ sudo sysctl -w net.mpls.conf.ens4.input=1

# Allocate entries in MPLS label table
$ sudo sysctl -w net.mpls.platform_labels=1048575
```

vPE East: FRR OSPF and LDP Configurations



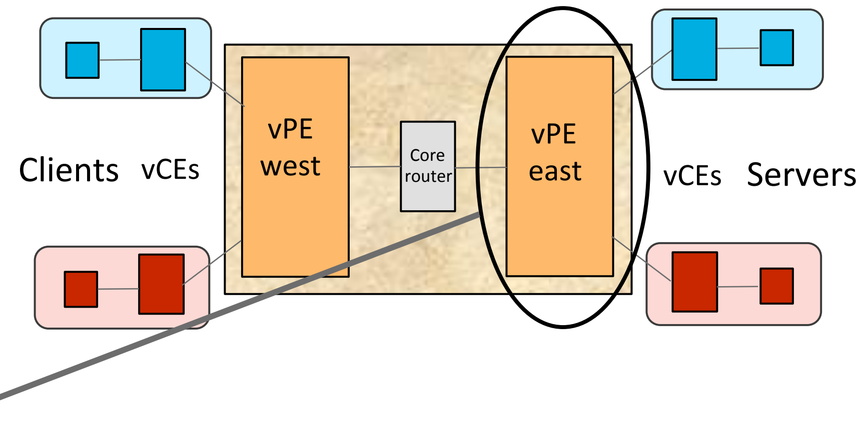
OSPF Configuration

```
router ospf
 !
 ! Loopback1 address as router-id
 ospf router-id 192.168.1.3
 !
 ! Add both loopbacks to OSPF
 network 192.168.1.3/32 area 0
 network 192.168.2.3/32 area 0
 !
 ! Adjacency to the core router
 network 10.31.11.0/24 area 0
!
```

LDP Configuration

```
mpls ldp
 !
 ! Use loopback1 as router-id
 router-id 192.168.1.3
 address-family ipv4
  !
  ! Discover the neighbor on
  ! specified interface
  discovery transport-address 192.168.1.3
  interface ens4
  !
 !
!
```

vPE East: VRF Configuration

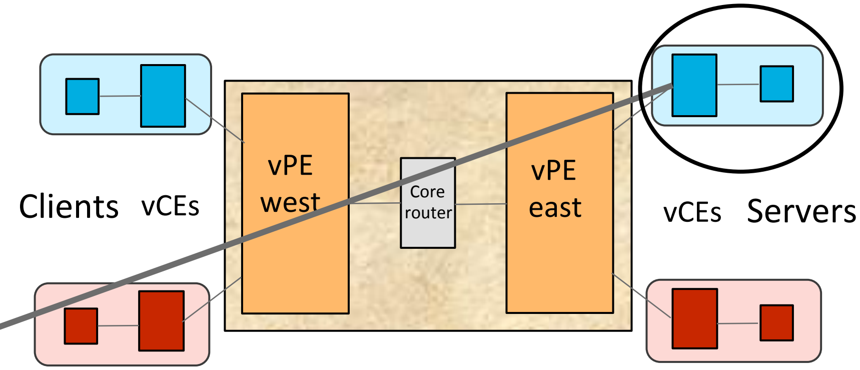



```
# Create VRF blue and bring it up
$ sudo ip link add blue type vrf table 1
$ sudo ip route add table 1 unreachable default metric 4278198272
$ sudo ip link set dev blue up

# Add interface to vCE blue-east to VRF blue
$ sudo ip link set dev ens6 master blue

# Allow BGP to listen on port 179 over the VRF-bound interface
$ sudo sysctl -w net.ipv4.tcp_l3mdev_accept=1
$ sudo sysctl -w net.ipv4.udp_l3mdev_accept=1
```

vCE Blue East: FRR BGP Configuration




```
router bgp 65101
 bgp disable-ebgp-connected-route-check
!
! Use IP address of the interface to
! vPE-east as router-id
bgp router-id 10.31.4.10
 !
 ! Network with the video server
 network 10.31.0.0/24
 ! Loopback address of the video server
 network 172.16.0.2/32
 !
 ! Peering session with vPE-east
 neighbor 192.168.1.3 remote-as 65001
 neighbor 192.168.1.3 ebgp-multihop
```

Allows configuration of eBGP session to vPE East loopback

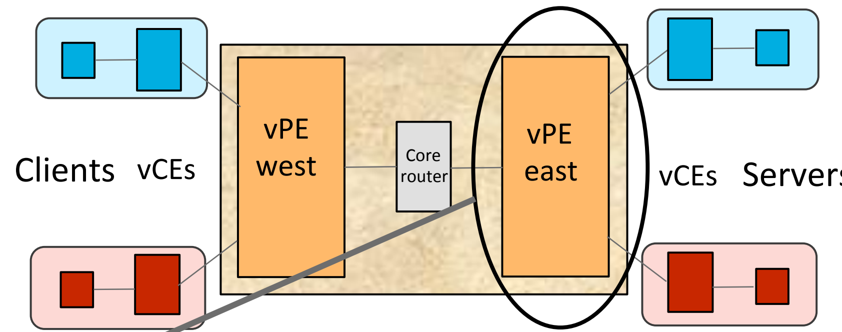
vPE East: GoBGP Configuration

```
[global.config]
    as = 65001
    router-id = "192.168.1.3"

[global.apply-policy.config]
    export-policy-list = [ "set-next-hop-self" ]

[zebra]
    [zebra.config]
        enabled = true
        url = "unix:/var/run/frr/zserv.api"
        version = 5
        mpls-label-range-size = 100

# iBGP client-session to core router.
# AFI/SAFI supported: VPNv4 unicast.
[[neighbors]]
    [neighbors.config]
        neighbor-address = "192.168.1.2"
    [neighbors.transport.config]
        local-address = "192.168.1.3"
    [[neighbors.afi-safis]]
        [neighbors.afi-safis.config]
            afi-safi-name = "l3vpn-ipv4-unicast"
```

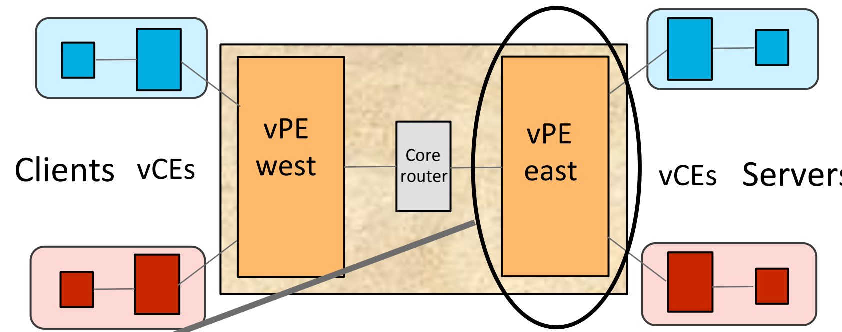


Use of policy for setting next-hop-self requires policy

Communicate to Linux kernel and Vyatta data-plane via FRR Zebra

iBGP session with Route-Reflector

vPE East: Defining “Set Next-Hop Self” policy



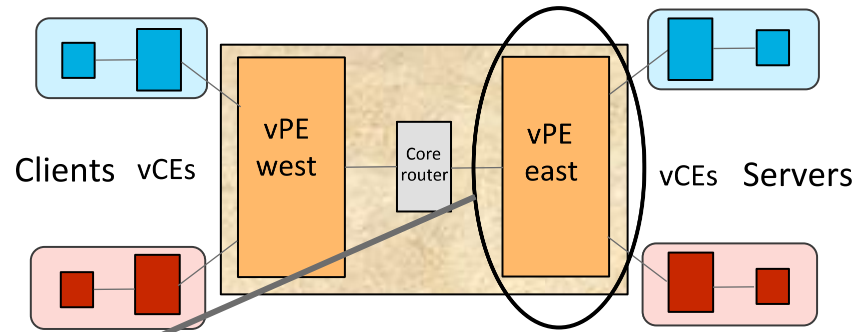

```
# Neighbor set.
[[defined-sets.neighbor-sets]]
    neighbor-set-name = "core-route-reflectors"
    neighbor-info-list = [ "192.168.1.2" ]

# Policy definition for next-hop-self
[[policy-definitions]]
    name = "set-next-hop-self"
    [[policy-definitions.statements]]
        [policy-definitions.statements.conditions.match-neighbor-set]
            neighbor-set = "core-route-reflectors"
        [policy-definitions.statements.actions.bgp-actions]
            set-next-hop = "192.168.2.3"
```

Match on the route reflector as neighbor

Set loopback2 of this PE as the next-hop

vPE East: Adding VRF and eBGP Neighbor via GoBGP CLI




```
$ ip link show blue
11: blue: <NOARP,MASTER,UP,LOWER_UP> mtu 65536 qdisc noqueue state UP
    mode DEFAULT group default qlen 1000
     link/ether ae:07:ef:a3:f3:f7 brd ff:ff:ff:ff:ff:ff
```

Commands for adding blue VRF and eBGP session to vCE east-blue

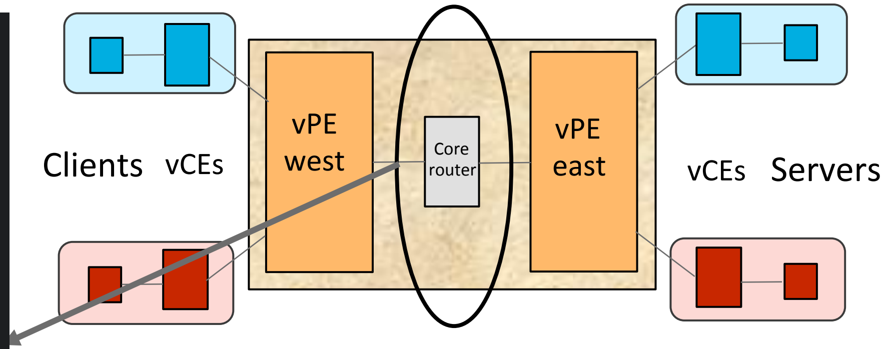
```
$ gobgp add vrf blue id 11 rd 100:1 rt both 100:1
$ gobgp nei add 10.31.4.10 as 65101 vrf blue
```

Use of ifIndex value assigned by Linux as VRF id

GoBGP Configuration for Router Reflector

```
[global.config]
    as = 65001
    router-id = "192.168.1.2"

# iBGP session to vpe-east router.
# AFI/SAFI supported: VPNv4 unicast.
[[neighbors]]
    [neighbors.config]
        neighbor-address = "192.168.1.1"
    [neighbors.transport.config]
        local-address = "192.168.1.2"
    [neighbors.route-reflector.config]
        route-reflector-client = true
        route-reflector-cluster-id = "192.168.1.2"
    [[neighbors.afi-safis]]
        [neighbors.afi-safis.config]
            afi-safi-name = "l3vpn-ipv4-unicast"
```



No need to communicate with FRR Zebra since VPNv4 routes are not installed in forwarding table

Implementation: L3-VPN Support in GoBGP

Key building blocks

Existing support was adequate

- Internet routing with BGP
  - Message handling, route computation, and policies
- Partition of routing table into global and VRF
  - Assign BGP sessions to appropriate partition
- VPNv(4|6) BGP address family
  - IP prefix, Route Distinguisher (RD) and MPLS label
- Route targets (RTs)
  - To associate routes with VRF(s)

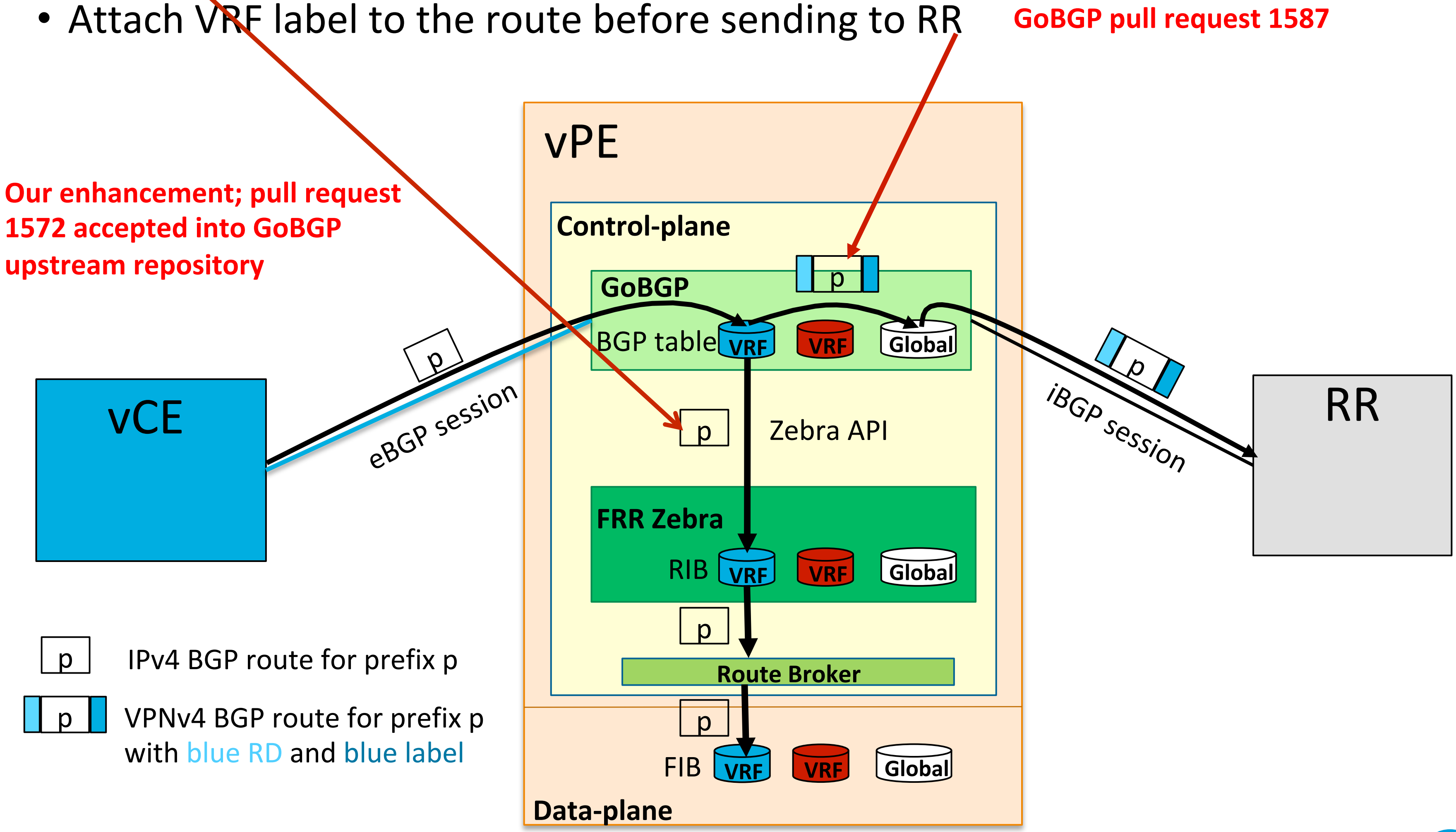
Interaction with “outside world”

Needed some enhancements

- Allow configuration of VRF(s)
  - Associate an eBGP session with CE to a VRF
- Handle a route received from a CE
- Handle a route received from RR (or remote PEs)
- Communicate with Zebra

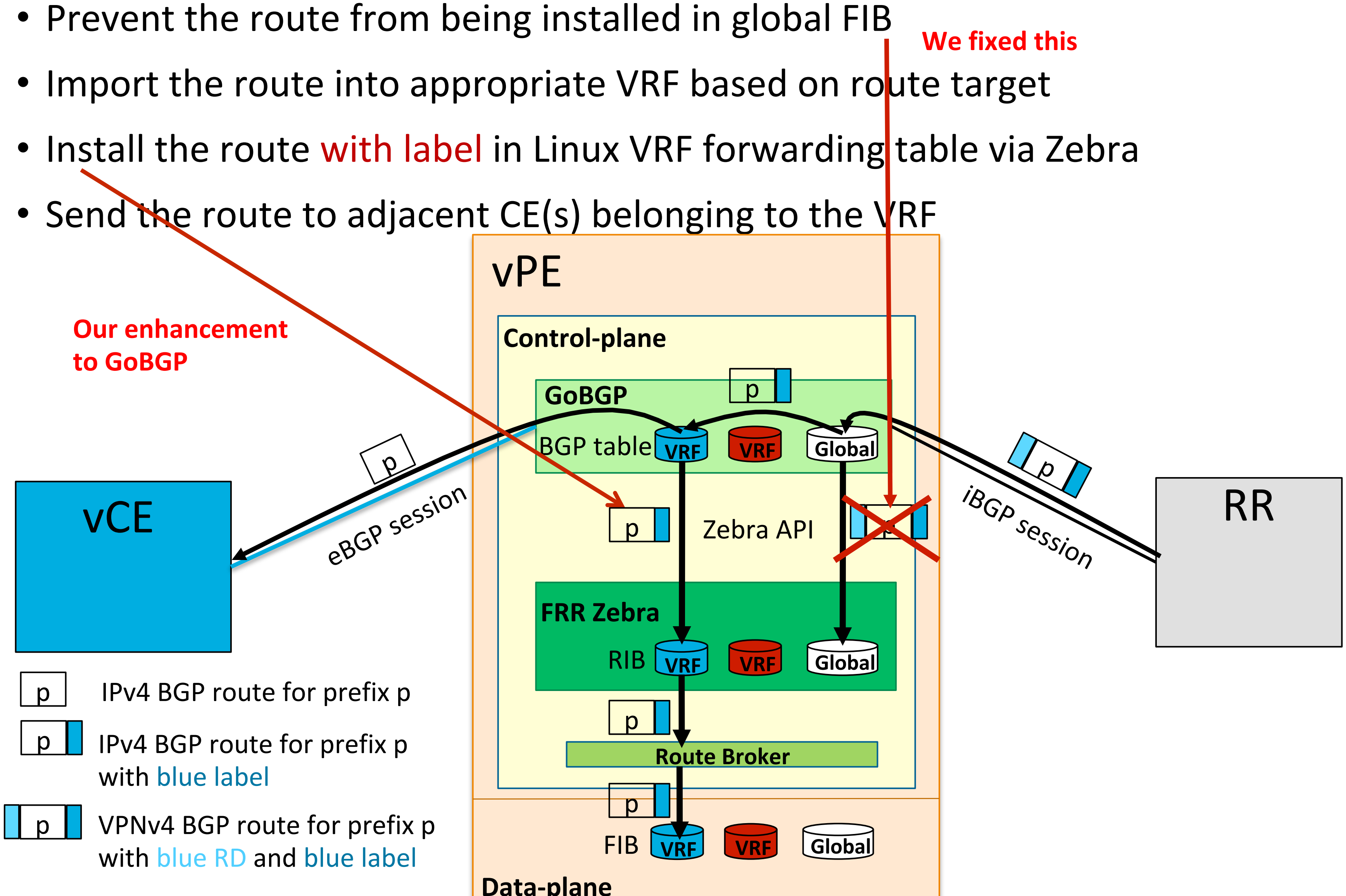
GoBGP: Handling a Route Received from an Adjacent CE

- Install the route in VRF forwarding table via Zebra
  - Allows vPE to send traffic coming from other PEs to the CE
- Attach VRF label to the route before sending to RR



GoBGP: Handling a Route Received from an RR or a PE

- Prevent the route from being installed in global FIB
- Import the route into appropriate VRF based on route target
- Install the route with label in Linux VRF forwarding table via Zebra
- Send the route to adjacent CE(s) belonging to the VRF



Summary

Demonstrated feasibility of creating an L3-VPN vPE using Open Source Software

- Control-plane:
  - GoBGP, FRR (OSPF, LDP and Zebra)
- Data-plane:
  - AT&T-Vyatta DPDK based data-plane
  - Also verified feasibility with VPP and Linux data-planes

Required us to make some enhancements to GoBGP 1.31

- Proper installation of routes into FIB
- Assign MPLS labels to VPNv4 routes
- Modifications available on Github at: https://github.com/amanshaikh75/gobgp/tree/zapi_version_5

DANOS URL: https://www.danosproject.org/

Acknowledgements

AT&T

- Bill Benson, Ramana Chinnapa, Kenneth Duell, Jennifer Yates

Cumulus Networks

- David Ahern (for explaining how Linux VRFs work)

FRR

- Donald Sharp, Renato Westphal, Russ White, https://github.com/paulzlabn

GoBGP

- Iwase Yusuke

VPP

- Michael Borokhovich, Pierre Pfister, Jeff Shaw

Backup

Open-Source Software across the Feasibility Test-bed

Network Function	VNF OS	Control-plane	Data-plane
vCE	Ubuntu 16.04.2 LTS Linux Kernel 4.4.0-64 generic	FRR 5.1-dev BGP and Zebra	Linux
vPE	Debian 4.14.62-0 Vyatta1+9.1 Linux Kernel 4.14.0-trunk-vyatta-amd64… (DANOS)	GoBGP 1.31.1 FRR 5.1-dev OSPF, LDP and Zebra (snapshot e8f9540)	AT&T-Vyatta DPDK
Core router	Ubuntu 16.04.3 LTS Linux kernel 4.14.4-mpls (custom configuration)	GoBGP 1.31.1 FRR 5.1-dev OSPF, LDP and Zebra (snapshot e8f9540)	Linux

Control-plane

- GoBGP 1.31.1 = version 1.31 + our enhancements
- FRR 5.1-dev = snapshot e8f9540

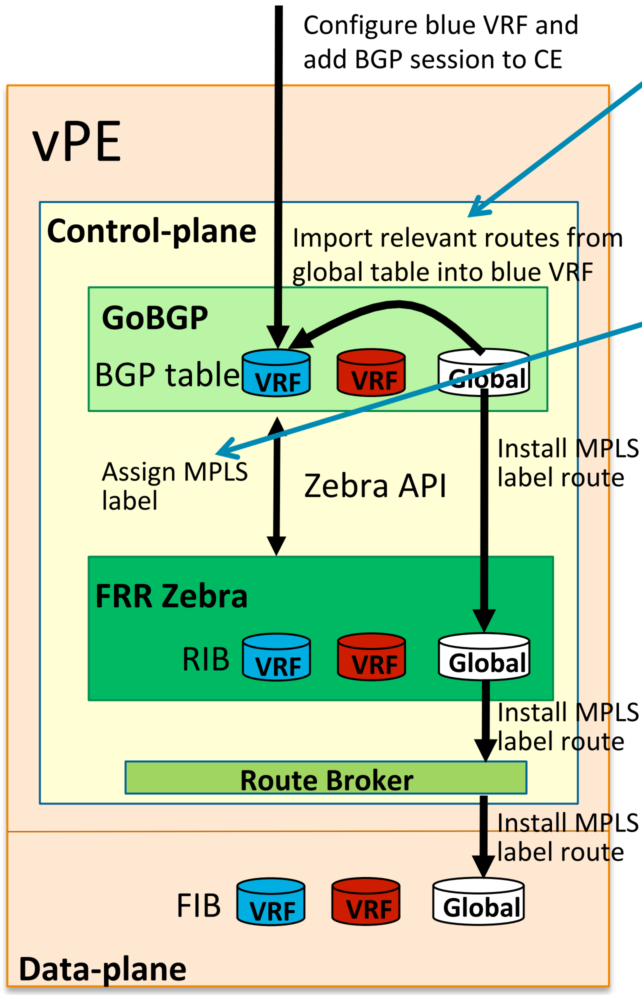
When Linux is used as data-plane on vPE

- vPE OS: Ubuntu 16.04.3 LTS, Linux kernel 4.14.4-mpls

When VPP is used as data-plane on vPE

- vPE data-plane: VPP release 1801 + our enhancements to router plug-in
- OS: same as when Linux is used as data-plane

GoBGP: Configuration of VRF and Associated CE Sessions




```
# Example of MPLS route installed
# in Linux kernel by GoBGP
$ ip -f mpls route
144 dev blue proto bgp
```

Import matching VPNv4 routes into VRF

- Routes received from RRs and/or other PEs
- **Outstanding issue; we use a workaround**

Obtain unique label for VRF from Zebra

- Zebra acts as a central agent for label assignment
  - Prevents label collision between different protocols like BGP and LDP
- **Pull request 1587 to GoBGP repository**

Install an MPLS route for the label in Linux default forwarding table

- Allows vPE to handle traffic from other PEs
- **We enhanced GoBGP code**

GoBGP: Interacting with Zebra

GoBGP by default uses API version 4 for interaction with Zebra

- API version 4 does not have all features to support L3-VPN
  - Example: lack of support for multi-level recursive next-hop lookup
- Required us to upgrade to Zebra API version 5

Added partial support for API version 5 in GoBGP

- Support for parts required for L3-VPN, not everything