Arquitetura de Comunicações

Docker Setup

The PE routers must be implemented as a **Docker** container with the FRRouting protocol suite installed. With the provided Dockerfile, create a custom docker image:

```
$ docker build -t local/frrouting .
```

Add the Docker container to GNS3 as template: (i) choosing an existing image, (ii) define it as local/frrouting container image, (iii) activate 4 network interfaces (adapters), and (iv) add the following directories as permanent in the advanced tab:

```
/etc/frr/
/etc/init.d/
```

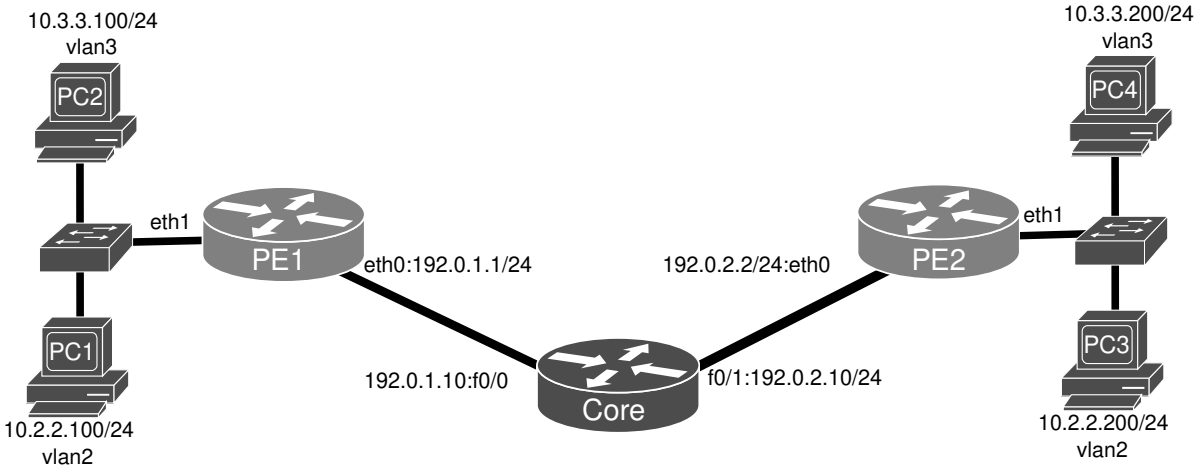
Note: FRRouting image is a linux container with a set of services/protocols installed that enable advance network features. The command vtysh provides a Cisco-like CLI.

Note2: Running this docker outside GNS3 requires additional privileges enabled (NET_ADMIN and SYS_ADMIN):

```
$ docker run --cap-add NET_ADMIN --cap-add SYS_ADMIN -it local/frrouting
```

VXLAN

The following network setup depicts two private VLAN, with devices in two remote locations. Both VLAN devices must be seamless connected at Layer 2 over an IP (Layer 3) network. PC end-devices must not have a default gateway configured (to ensure Layer2-only connectivity).



1. Assemble the previous network setup in GNS3. The Core router should be a Cisco device and Routers PE1 and PE2 should be a local/frrouting docker device. Configure all IPv4 addresses and OSPF protocol at the core connections. For PE1:

```
bash-5.1# vtysh
PE1# configure terminal
PE1(config)# int eth0
PE1(config-if)# ip address 192.0.1.1/24
PE1(config-if)# ip ospf 1 area 0
PE1(config-if)# no shutdow
PE1(config-if)# router ospf 1
PE1(config-if)# end
PE1# write
```

Make a similar configuration in PE2.

Verify the IPv4 routing table at the PE routers using the commands:

```
#show ip route
#show ip fib
```

>> Verify the connectivity between PE1 and PE2.

Note: to exit the Cisco-like CLI vtysh, type exit.

2. Configure the PC IPv4 addresses (without gateway) and the Layer 2 switches. PC1 and PC3 belong to VLAN2 and PC2 and PC4 belong to VLAN3, **configure the respective VLAN ports at the switches**. **The connection between the Layer 2 switches and the PE routers should be a 802.1Q trunk.** In the switches, configure a trunk port (dot1q) to connect the switches to the PE routers.

Note: FRRouting vtysh does not yet support bridge and vxlan commands. Commands must be done in Linux bash.

At the PE routers configure sub-interfaces for VLAN 2 and VLAN 3. For PE1 (**in Linux bash**):

```
bash-5.1#
 ip link add link eth1 name eth1.2 type vlan id 2
 ip link add link eth1 name eth1.3 type vlan id 3
 ip link set up dev eth1.2
 ip link set up dev eth1.3
```

Make an equal configuration in PE2.

Check the status of the sub-interfaces with the commands:

```
 ip -d link show eth1.2
 ip -d link show eth1.3
```

3. Create two VXLAN connections between the PE. For PE1:

```
bash-5.1#
 ip link add vxlan102 type vxlan id 102 remote 192.0.2.2 dstport 4789
 ip link add vxlan103 type vxlan id 103 remote 192.0.2.2 dstport 4789
 ip link set up dev vxlan102
 ip link set up dev vxlan103
```

Make an similar configuration in PE2, change only the remote VXLAN address.

Check the status of the VXLAN interfaces with the commands:

```
 ip -d link show vxlan102
 ip -d link show vxlan103
```

4. Create two virtual bridges with the VXLAN virtual interfaces, and the respective client VLAN sub-interface. For PE1:

```
bash-5.1#
 brctl addbr br102
 brctl addif br102 vxlan102
 brctl addif br102 eth1.2
 brctl stp br102 off

 brctl addbr br103
 brctl addif br103 vxlan103
 brctl addif br103 eth1.3
 brctl stp br103 off

 ip link set up dev br102
 ip link set up dev br103
```

Make an equal configuration in PE2.

Check the status of each bridge with the commands:

```
 brctl show br102
 brctl show br103
```

Note: The virtual bridge interfaces can have an IPv4 address and act as gateways of the VLAN.

5. Start a packet capture in one of the core links. Test the connectivity between the PCs of the same VLAN.

>> Analyze capture packets.

>> Explain how the packets are being sent from PE1 to PE2 (and vice-versa).

>> Explain how the source VLAN (VXLAN connection) is identified.

>> Discuss the limitations of the VXLAN connection when more than 2 remote sites exist.

Note: **This Linux bash commands are not permanent!** They will disappear upon a reboot. To make them permanent write them in file /etc/init.d/adv_network. These commands will be run on every start of the Docker container.

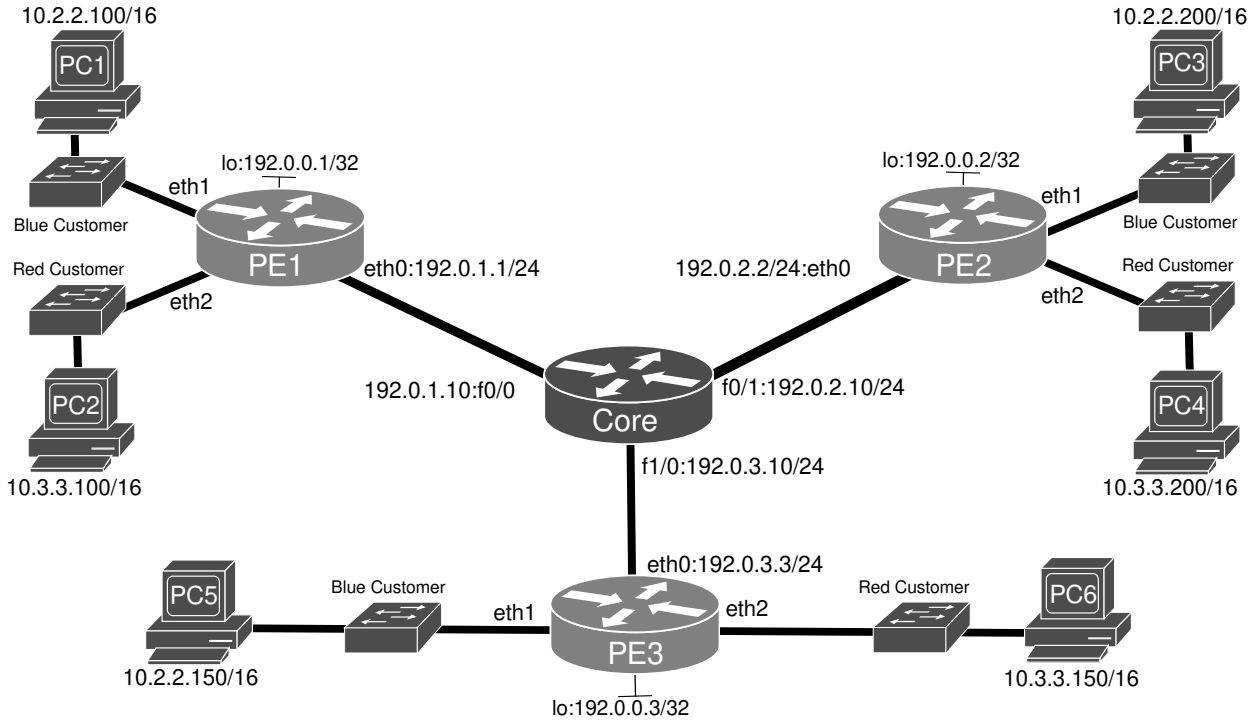
L2VPN/EVPN with VXLAN transport

The following network setup is an evolution of the previous one. Now, you may have multiple remote locations (three in this scenario) and multiple different costumers (Blue and Red). Only a single VLAN from each costumer exists (VLAN 2), however multiple VLAN per costumer may be possible. All VLAN devices must be seamless connected at Layer 2 over an IP (Layer 3) network. PC end-devices must not have a default gateway configured (to ensure Layer2-only connectivity).

A VNI is considered to be either Layer-2 (tied to a MAC-VRF) or Layer-3 (tied to an IP-VRF), which indicates what kind of information is represented by the VRF. An IP-VRF represents a routing table (operating in much the same way as a VRF traditionally operates in L3VPN), while a MAC-VRF represents a bridging table.

In order to allow the costumers the usage of the same private networks, each costumer should be in a different IP-VRF. Therefore, this setup will have one IP-VRF for each costumer, and all MAC-VRFs of the same costumer will be associated with the respective IP-VRF. In summary, it will have:

- An IP-VRF named vrf2000, associated with L3 VNI 2000 (Blue Costumer);
- A MAC-VRF using VLAN 2, associated with L2 VNI 202 and IP-VRF vrf2000 (Blue Costumer);
- An IP-VRF named vrf3000, associated with L3 VNI 3000 (Blue Costumer);
- A MAC-VRF using VLAN 2, associated with L2 VNI 302 and IP-VRF vrf3000 (Red Costumer).



1. Assemble the previous network setup in GNS3. The Core router should be a Cisco device and Routers PE1 to PE2 should be a local/frrouting docker device. Configure all IPv4 addresses and OSPF protocol at the core connections, including the loopback (lo) interfaces. For PE1:

```
bash-5.1# vtysh
PE1# configure terminal
PE1(config)# int eth0
PE1(config-if)# ip address 192.0.1.1/24
PE1(config-if)# ip ospf 1 area 0
PE1(config-if)# no shutdow
PE1(config-if)# int lo
PE1(config-if)# ip add 192.0.0.1/32
PE1(config-if)# ip ospf 1 area 0
PE1(config-if)# router ospf 1
PE1(config-if)# end
PE1# write
```

Make a similar configuration in PE2 and PE3.

Verify the IPv4 routing table at the PE routers using the commands:

```
#show ip route
#show ip fib
```

>> Verify the connectivity between PE1, PE2 and PE3 loopback interfaces.

2. Create the IP-VRF for both clients, in PE1:

```
bash-5.1#
ip link add vrf2000 type vrf table 2000
ip link add vrf3000 type vrf table 3000
ip link set vrf2000 up
ip link set vrf3000 up
vtysh
PE1# configure
PE1(config)#vrf vrf2000
PE1(config-vrf)# vni 2000
PE1(config)#vrf vrf2000
PE1(config-vrf)#vni 2000
PE1(config-if)# end
PE1# write
```

Verify the VRF/VNI status with:

```
PE1# show vrf vni
```

Make equal configurations in PE2 and PE3.

Note: These IP-VRFs will not be associated with a VXLAN interface. When associated to a VXLAN interface, they allow the creation of a Layer3 VPN over EVPN/VXLAN (exchange of Type-5 EVPN routes).

3. Configure a Layer2 VPN between the customer networks for both customers, using a BGP EVPN with VXLAN transport. Consider a Spine-Leaf relation between the PE routers where PE1 is the Spine. Assume ASN 100. Consider internal BGP relations within the same AS with the usage of Route Reflectors.

For PE1 (Spine - Route Reflector):

```
bash-5.1# vtysh
PE1# configure terminal
PE1(config)# router bgp 100
PE1(config-router)# neighbor EVPN peer-group
PE1(config-router)# neighbor EVPN remote-as 100
PE1(config-router)# neighbor EVPN update-source 192.0.0.1
PE1(config-router)# neighbor 192.0.0.2 peer-group EVPN
PE1(config-router)# neighbor 192.0.0.3 peer-group EVPN
PE1(config-router)# address-family l2vpn evpn
PE1(config-router-af)# neighbor EVPN activate
PE1(config-router-af)# neighbor EVPN route-reflector-client
PE1(config-router-af)# advertise-all-vni
PE1(config-router-af)# end
PE1# write
```

For PE2 (Leaf - Route Reflector client):

```
bash-5.1# vtysh
PE1# configure terminal
PE1(config)# router bgp 100
PE1(config-router)# neighbor 192.0.0.1 remote-as internal
PE1(config-router)# neighbor 192.0.0.1 update-source 192.0.0.2
PE1(config-router)#  address-family l2vpn evpn
PE1(config-router-af)# neighbor 192.0.0.1 activate
PE1(config-router-af)# advertise-all-vni
PE1(config-router-af)# end
PE1# write
```

Make similar configurations for PE3, just changing the update-source address.

Analyze the BGP neighbors status and BGP table of each PE router, and identify the learned EVPN type-3 and type-2 prefixes:

```
PE1# show bgp neighbors
PE1# show bgp l2vpn evpn
```

>> Explain the advantages of a BGP Spine-Leaf architecture with Route Reflectors versus a full BGP mesh.

4. Configure the VXLAN and bridge interfaces for each customer. For PE1:

```
bash-5.1#
  ip link add br202 type bridge
  ip link set br202 master vrf2000
  ip addr add 10.2.2.2/16 dev br202

  ip link add br302 type bridge
  ip link set br302 master vrf3000
  ip addr add 10.3.3.2/16 dev br302

  ip link add vni202 type vxlan local 192.0.0.2 dstport 4789 id 202 nolearning
  ip link add vni302 type vxlan local 192.0.0.2 dstport 4789 id 302 nolearning
  ip link set vni202 master br202
  ip link set vni202 type bridge_slave neigh_suppress on learning off
  ip link set vni302 master br302
  ip link set vni302 type bridge_slave neigh_suppress on learning off

  ip link add link eth1 name eth1.2 type vlan id 2;
  ip link add link eth2 name eth2.2 type vlan id 2;
  ip link set eth1.2 master br200
  ip link set eth2.2 master br300

  ip link set eth1.2 up
  ip link set vni202 up
  ip link set br202 up
  ip link set eth2.2 up
  ip link set vni302 up
  ip link set br302 up
```

Make similar configurations for PE2 and PE3.

>> Explain why now the VXLAN connections do not have a remote address defined.

5. Start packet captures in the three core links. At PE1 router restart the BGP process:

```
PE1$ clear ip bgp *
```

>> Analyze the captured EVPN BGP UPDATE packets.

>> Explain how EVPN type-3 and type-2 prefixes are exchanged.

6. Analyze the BGP table of each PE router, and identify the learned EVPN type-3 and type-2 prefixes:

```
PE1# show bgp l2vpn evpn
```

In order to each PE router learn the MAC addresses of the terminals in each network, ping from each terminal the respective PE (gateway). Identify, after each ping, in the BGP table of each PE router the learned EVPN type-2 prefixes.

Verify the learned IP-MAC mappings in each EVPN:

```
PE1# show evpn mac vni 202
PE1# show evpn mac vni 302
```

>> Explain how and when EVPN type-2 prefixes, related to the customer terminals, are exchanged.

>> Explain why after some time EVPN type-2 prefixes are withdrawn from the BGP table.

7. Start a packet capture in the network core. Test the connectivity between all end-devices. From the PCs, ping non-existing IP addresses, and IP/MAC addresses not exchanged between the EVPN peers.

>> Explain why and when ARP packets are transported between sites (based on learned Type-2 routes).

>> Explain how the client data packets are transported and differentiated between the customer networks.

(Optional) Deploy a Layer3 VPN over EVPN/VXLAN.

See: https://docs.frrouting.org/en/latest/evpn.html

(Optional) Change the Spine-Leaf BGP relations setup from a single AS with Route Reflector to a setup with no Route Reflectors. Consider the usage of private AS within the network: (i) all Leaf devices in a single private AS and (ii) each Leaf device with its individual private AS.