BGP EVPN for VXLAN

A SINGLE ROUTING PROTOCOL FOR PHYSICAL AND VIRTUAL TOPOLOGIES

Contents

Introduction

Many data centers today are moving from a legacy layer 2 design to a modern layer 3 web-scale IT architecture. Layer 3 designs using traditional routing protocols like OSPF and BGP allow simplified troubleshooting, clear upgrade strategies, multi-vendor support, small failure domains and less vendor lock-in. However, many applications, storage appliances and tenant considerations still require layer 2 adjacency.

Virtual Extensible LAN (VXLAN) is widely deployed in many layer 3 data centers to provide layer 2 connectivity between hosts for specific applications. For example, as seen in Figure 1, the webservers and the load balancer must be on the same layer 2 network. VXLAN provides that layer 2 connectivity over a layer 3 infrastructure.

Ethernet Virtual Private Network (EVPN) is a feature offered by Cumulus Networks that provides a scalable, interoperable end-to-end control-plane solution for VXLAN tunnels using BGP. It supports redundancy, load sharing and multi-tenant segmentation. EVPN also provides the benefit of fast convergence for host and VM mobility over VXLAN tunnels and ARP suppression.

This white paper discusses deployment benefits, how EVPN works, how to operate EVPN, and different deployment scenarios. This paper also includes sample Cumulus Linux configurations to deploy a scalable, controller-free layer 2 virtualization over a layer 3 IP fabric using the standard well-known routing protocol, BGP.

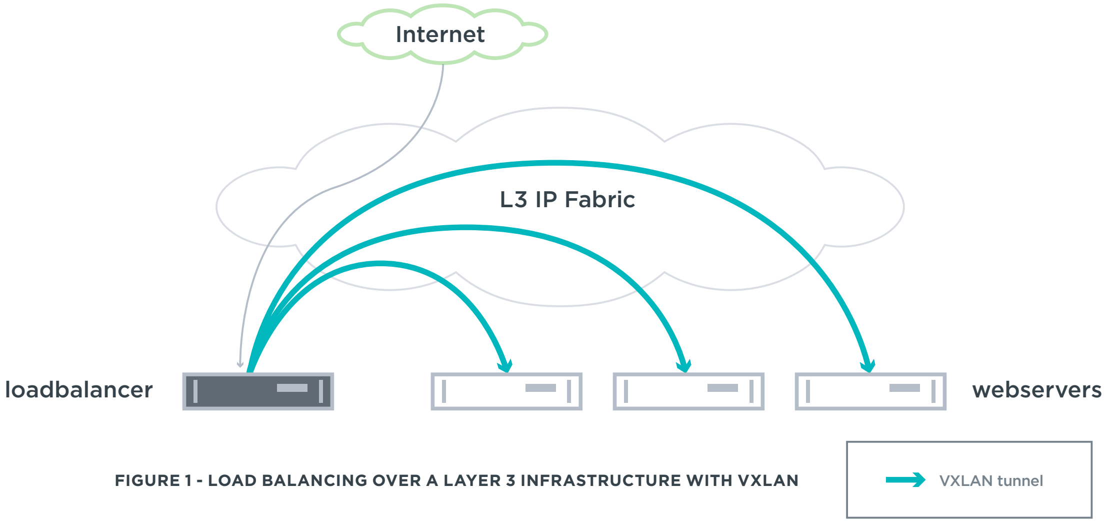


FIGURE 1 - LOAD BALANCING OVER A LAYER 3 INFRASTRUCTURE WITH VXLAN

Deployment benefits summary

Deploying EVPN provides many advantages to a layer 3 data center:

Simplicity: EVPN uses the BGP routing protocol. BGP is also the preferred routing protocol for data center infrastructures. The same routing protocol can be used for both infrastructure and virtual topologies.

Controller-less VXLAN tunnels: No controller is needed for VXLAN tunnels, as EVPN provides peer discovery with authentication natively. This also mitigates the chance of rogue VTEPs in a network and dealing with complicated controller redundancy.

ARP Suppression: Cumulus EVPN reduces broadcast traffic within a data center by allowing the local leaf switch to respond to a host's ARP requests instead of forwarding throughout the data center.

Scale and robustness: EVPN uses the BGP routing protocol. BGP is very mature, scalable, flexible and robust. It is the primary routing protocol for the Internet and data centers. It can hold a very large number of routes. It supports routing policy and filtering, which provides granular control over traffic flow.

Fast convergence and host mobility: Cumulus EVPN supports the new BGP MAC mobility extended community, offering fast convergence and reducing discovery traffic after a MAC or VM move. MAC stickiness is also supported, preventing specific host mobility if desired.

Support for VXLAN active-active mode: Cumulus EVPN integrates with MLAG, thereby providing host dual homing for redundancy.

Multitenancy: EVPN uses the mature multi-protocol BGP VPN technology to separate tenants within a data center.

Interoperability between vendors: The standardized multi-protocol BGP (MP-BGP) is used for the EVPN control plane. As long as vendor implementations maintain adherence to both the VXLAN and EVPN standards, interoperability is assured.

EVPN is a standardized control plane protocol that offers controller-less VXLAN tunnels. It also offers scale, redundancy, fast convergence and robustness while reducing broadcast, unknown unicast, and multicast (BUM) traffic across a data center core. More details on the operations providing these benefits are discussed below.

EVPN overview and operations

Customers are moving from traditional layer 2 data centers to a layer 3 fabric to overcome one or more of these issues:

- **Large broadcast and failure domains:**
  A broadcast packet is sent throughout the data center, increasing utilization and a failure can impact the entire data center.
- **Limited redundancy:**
  MLAG is often deployed for redundancy but it supports only 2 switches.
- **Troubleshooting difficulty:**
  Spanning tree issues can cause a network meltdown and are difficult to troubleshoot.
- **Limited scale for tenant separation:**
  A maximum of only 4094 VLANs are supported.

While moving to a layer 3 fabric should overcome these issues, some applications still require layer 2 connectivity between servers, so VXLAN tunnels are often deployed. VXLAN tunnels are identified by IETF RFC 7348 "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks."

VXLAN provides a scalable solution for layer 2 virtualization over a layer 3 routed infrastructure. It allows up to 16 million different VXLANs in the same domain by allocating a 24-bit segment ID called either the VXLAN network identifier (VNI) or the VXLAN-ID. The VNI is used to distinguish between VXLAN tunnels.

VXLAN provides a scalable solution for layer 2 virtualization over a layer 3 routed infrastructure

Virtual Tunnel Endpoints (VTEPs) are used to originate and terminate the VXLAN tunnel and map end devices such as hosts and VMs to VXLAN segments. The VTEP provides the encapsulation of layer 2 frames into User Datagram Protocol (UDP) segments to traverse across a layer 3 fabric. Likewise, the VTEP also de-capsulates the UDP segments from a VXLAN tunnel to send to a local host. A VTEP requires an IP address (often a loopback address) and uses this address as the source/destination tunnel IP address. The VTEP IP address must be advertised into the routed domain so the VXLAN tunnel endpoints can reach each other as shown in Figure 2. Each switch that hosts a VTEP must have a VXLAN-supported chipset such as Mellanox Spectrum or Broadcom Trident II, Trident II+ or Tomahawk. A list of our compatible hardware can be found in the Hardware Compatibility List. Though it’s not depicted in Figure 2, you can have multiple VNIs (VXLANs) using one VTEP IP address.

In traditional VXLAN, as seen below in Figure 2, the control plane and the data plane are integrated together — meaning MAC address learning happens over the data plane, often called *flood and learn*. This causes limitations, including limited load balancing and slow convergence times especially for VM and host mobility. Further, broadcast, unknown unicast, and multicast (BUM) traffic, such as ARP, is required to traverse across the tunnel for discovery purposes, thereby increasing overall data center traffic.

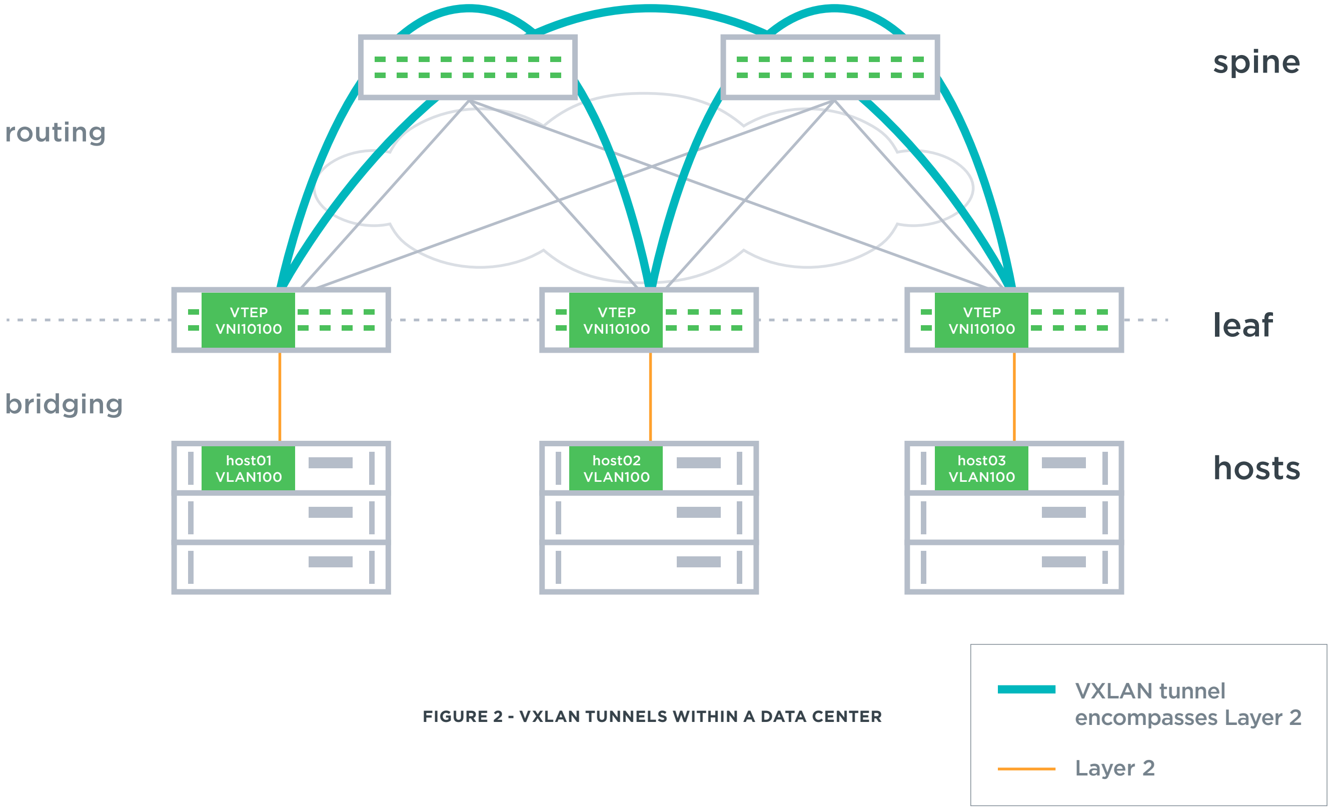


FIGURE 2 - VXLAN TUNNELS WITHIN A DATA CENTER

The Cumulus Networks EVPN implementation provides a separate control plane for VXLAN tunnels. EVPN provides exchange of MAC/IP addresses between VTEPs through the use of a separate control plane, similar to pure IP routing. Cumulus EVPN is an open and standards based solution that implements IETF RFC 7432 "BGP MPLS-Based Ethernet VPN" along with IETF draft "A Network Virtualization Overlay Solution using EVPN" for a VXLAN tunnel control plane.

EVPN introduces a new address family to the MP-BGP protocol family, as depicted in Figure 3.

EVPN provides remote VTEP discovery, thus it doesn't require an external controller. Learning control plane information independently of the data plane offers greater redundancy, load sharing and multipathing while also supporting MAC address filtering and traffic engineering, which can provide granular control of traffic flow. EVPN also provides faster convergence for mobility. Greater redundancy and multipathing can be achieved because all the possible paths are exchanged across the control plane, not just from one data plane path.

When EVPN is implemented with the VXLAN data plane, the evpn address family can exchange either just the MAC layer control plane information (that is, MAC addresses) or it can exchange both the MAC address and IP address information in its updates between VTEPs. Exchanging IP and MAC information together can allow for ARP suppression at the local switch, thereby reducing the broadcast traffic in a data center.

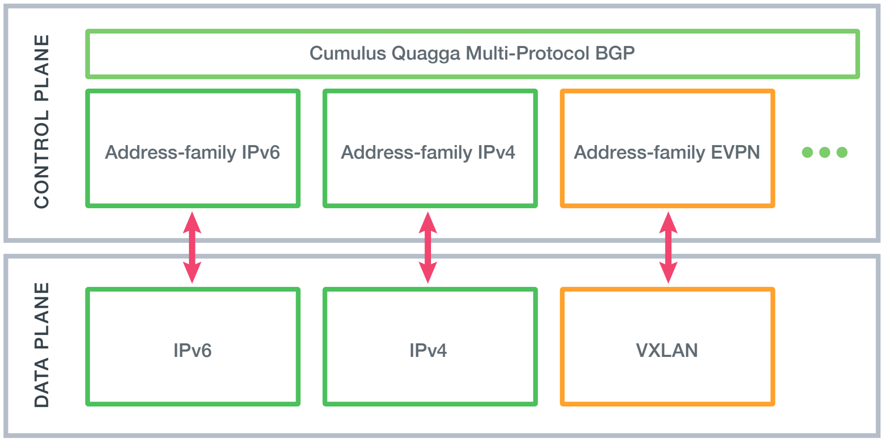


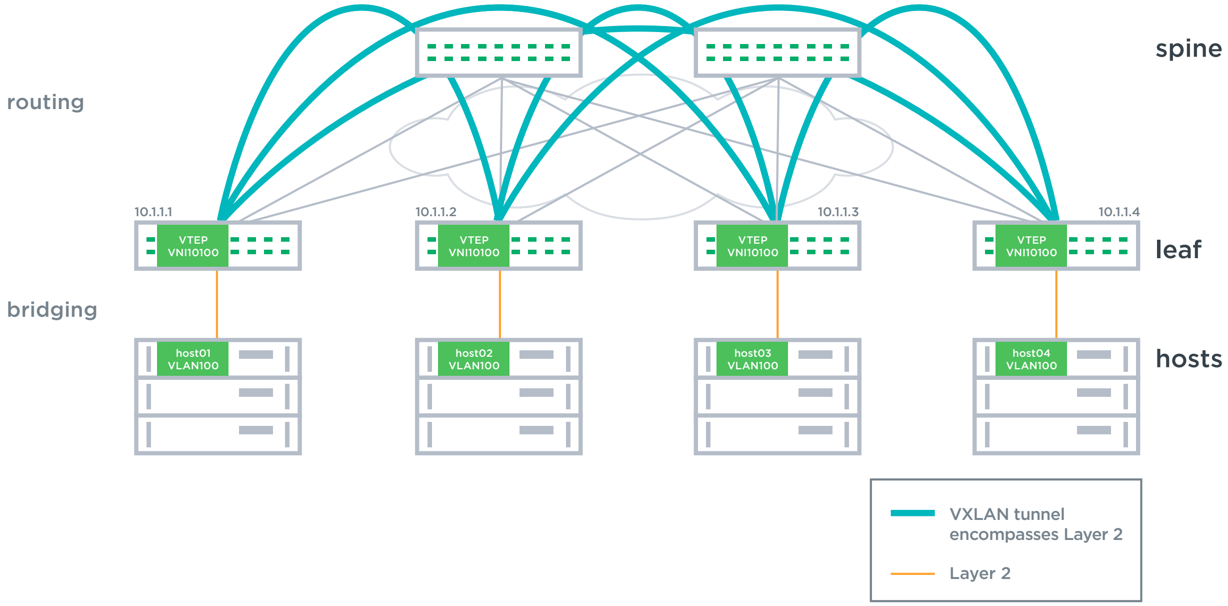
FIGURE 3 - EVPN ADDRESS FAMILY

EVPN VTEP PEER DISCOVERY

One large advantage of deploying EVPN is the ability to deploy controller-free VXLAN tunnels. EVPN uses type 3 EVPN routes to exchange information about the location of the VTEPs on a per-VNI basis, thereby enabling automatic discovery. It also reduces or eliminates the chance of a rogue VTEP being introduced in the data center.

EVPN offers peer discovery, thus requiring no external controller

For example, in Figure 4, the VTEPs are automatically discovered via eBGP and do not need to be explicitly configured or controlled as peers. The spine switches do not need to be configured for VLAN or VXLAN at all. All the discovered VTEPs within a VXLAN can easily be seen from one participating VTEP with a simple show command. The command in Figure 4 below displays the number of remote VTEPs associated with a specific VNI that are automatically discovered, including any rogue VTEPs.



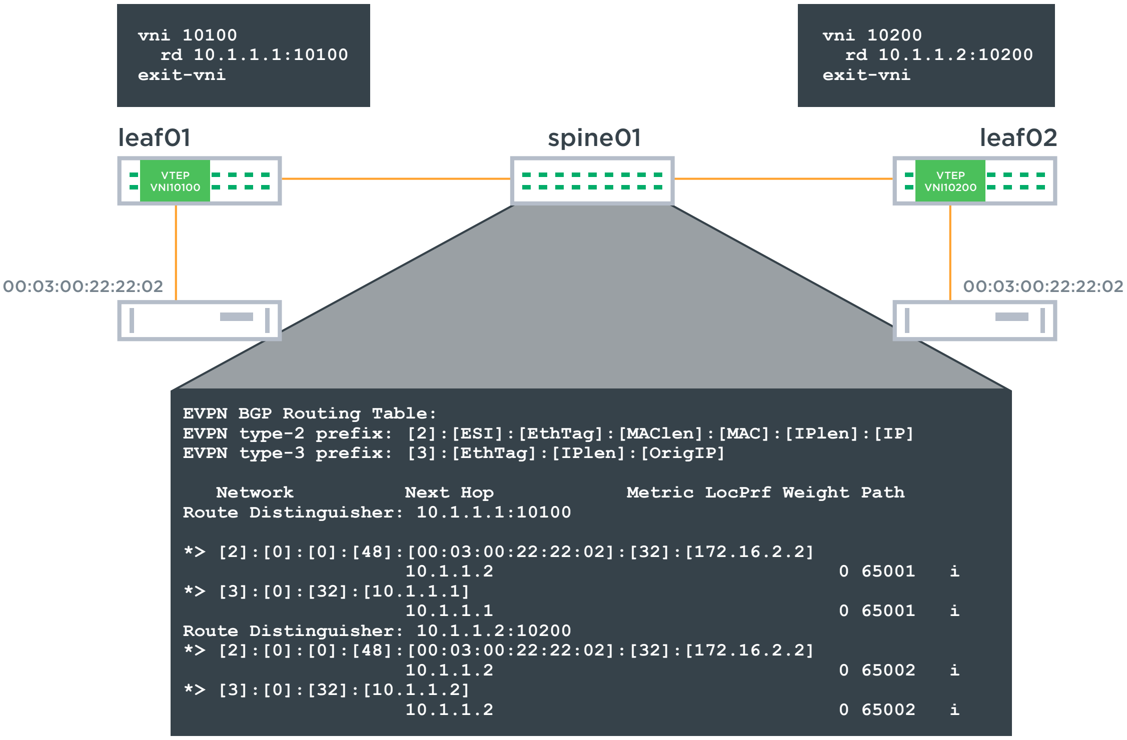
```
cumulus@leaf03~$ net show evpn vni
Number of VNIs: 1
VNI       VxLAN IF     VTEP IP        # MACs    # ARPs    # Remote VTEPs
10100     vxlan_100    10.1.1.3          3         3              3
```

FIGURE 4 - VTEP PEER DISCOVERY

EVPN MULTI-TENANT SUPPORT

The new EVPN address family also provides multi-tenant separation and allows for overlapping addresses between tenants. To maintain the separation, it uses mature MP-BGP VPN technology: Route Distinguishers (RDs) and Route-Targets (RTs).

The RD makes overlapping routes from different tenants look unique to the data center spine switches to provide proper routing. A per-VXLAN 8-byte RD is prepended to each advertised route before the route is sent to its BGP EVPN peer. In Figure 5, the same route is advertised from 2 hosts in separate tenants, but the spine router can distinguish between the routes since they have different route distinguishers. (Some entries left off for brevity)



```
vni 10100
  rd 10.1.1.1:10100
exit-vni
```

```
vni 10200
  rd 10.1.1.2:10200
exit-vni
```

```
EVPN BGP Routing Table:
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 10.1.1.1:10100

*> [2]:[0]:[0]:[48]:[00:03:00:22:22:02]:[32]:[172.16.2.2]
                    10.1.1.2                               0 65001   i
*> [3]:[0]:[32]:[10.1.1.1]
                    10.1.1.1                               0 65001   i
Route Distinguisher: 10.1.1.2:10200
*> [2]:[0]:[0]:[48]:[00:03:00:22:22:02]:[32]:[172.16.2.2]
                    10.1.1.2                               0 65002   i
*> [3]:[0]:[32]:[10.1.1.2]
                    10.1.1.2                               0 65002   i
```

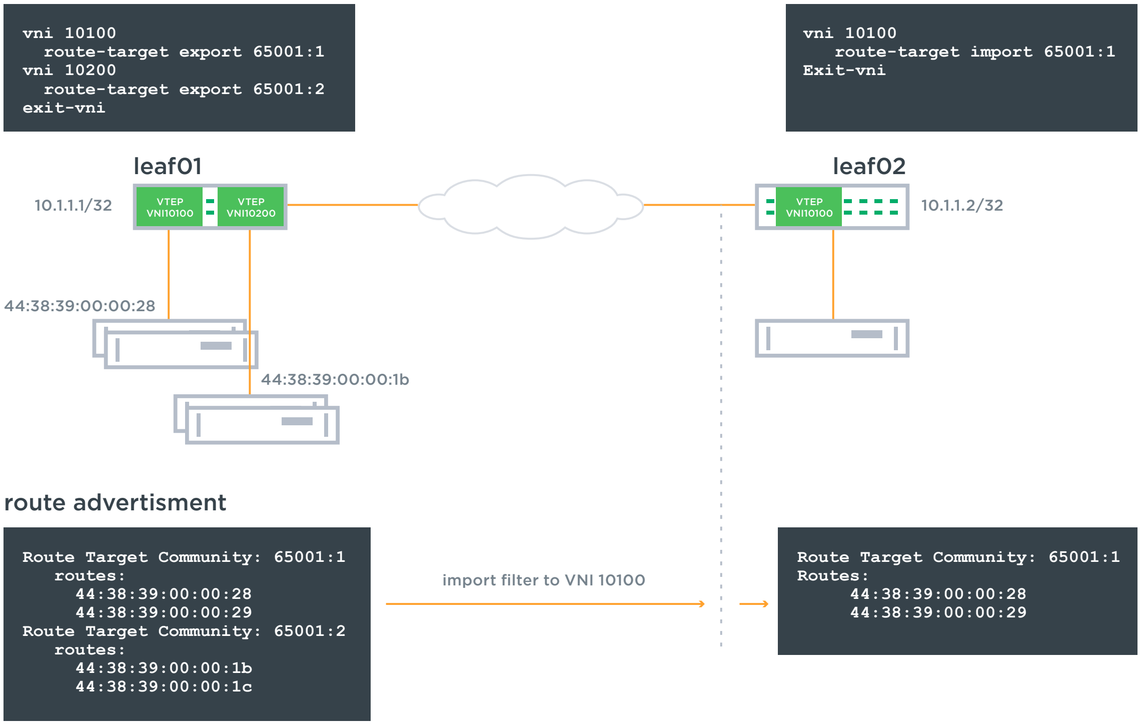
FIGURE 5 - ROUTE DISTINGUISHERS

EVPN also makes use of the RT extended community for route filtering and separating tenants. The RT is advertised in the BGP update message along with the EVPN routes. The RT community distinguishes which routes should be exported from and imported into a specific VNI route table on a VTEP. If the export RT in the received update matches the import RT of a VNI instance on the VTEP receiving the update, the corresponding routes will be imported into that VNI's EVPN Route table. If the RTs do not match, the route will not be imported into that VNI's EVPN route table.

EVPN provides multi-tenant separation with one protocol instance

Figure 6 below depicts leaf01 sending a BGP EVPN MAC route to leaf02 with the attached route-target community. As seen, four MAC routes are sent in the advertisement, two each originating from different VNIs on leaf01. Since leaf02 only has the one route-target import, 65001:1, it will receive only those routes associated with 65001:1, and the route with route-target 65001:2 will not be installed as there is no matching import route-target within VNI 10100 located on leaf02.

Cumulus Linux supports either a default RD and/or RT for configuration ease, or configuring explicit RD and/or RT within BGP for each VNI to allow flexibility. By default, the switch automatically derives the RD and RT from the VNI. In the default case, the RD would be *Router ID: n* (where n is assigned chronologically from *1)* , and the export RT is set at AS:VNI. The import RT community is set to *<any>:VNI,* which allows all routes from that same VXLAN to be imported. If more granular control of importing routes, compatibility with other vendors, and/or if globally unique VNIs are not configured, the RD and RT community can be manually configured as well. Manually configuring the RT and RD overrides the default RD and RT values.



```
vni 10100
  route-target export 65001:1
vni 10200
  route-target export 65001:2
exit-vni
```

```
vni 10100
   route-target import 65001:1
Exit-vni
```

```
Route Target Community: 65001:1
   routes:
     44:38:39:00:00:28
     44:38:39:00:00:29
Route Target Community: 65001:2
   routes:
     44:38:39:00:00:1b
     44:38:39:00:00:1c
```

```
Route Target Community: 65001:1
Routes:
        44:38:39:00:00:28
        44:38:39:00:00:29
```

FIGURE 6 - USING ROUTE TARGETS TO FILTER BETWEEN TENANTS AND VTEPS

MAC + IP ADDRESS LEARNING/EXCHANGE

Cumulus Networks supports advertising either the MAC routes only, or advertising the MAC+IP routes in Cumulus Linux 3.3 and later. The MAC+IP address advertisement is necessary to support ARP suppression.

> Cumulus EVPN reduces broadcast traffic in a data center via ARP suppression

On the leaf switch, each local VLAN is mapped to a VNI. When a local switch learns a new MAC+IP route on a particular VLAN, either via gratuitous ARP (GARP) or via the first data packet, which is typically an ARP request, the MAC address is placed into the local switch's bridge forwarding table. Additionally, the local leaf's ARP/ND table is populated with the IP to MAC layer mapping. The local MP-BGP process learns every new local MAC address from the local forwarding table and learns its corresponding IP route from the ARP/ND table. MP-BGP then advertises the MAC+IP route to the remote VTEPS via Type 2 EVPN routes.

On the remote end, the MAC+IP routes that BGP learns are placed into the BGP table. From there, if the route target community sent with the route matches a local VNI route-target import, the route will be placed into that switch's MAC forwarding table with the appropriate VXLAN tunnel as its destination. The IP address, if included, will be placed in the EVPN ARP cache. This process separates the data and control planes, allowing dynamic learning of MAC addresses, allows overlapping MAC+IP addresses between tenants, and allows granular filtering of MAC+IP addresses, all without requiring a data plane packet to traverse each switch first.

To walk through an example of the MAC+IP address being propagated through the network, consider the example network in Figure 7 where there are two leaf switches, each participating in two independent VXLAN tunnels. This same demo is available here to set this up virtually and follow along.

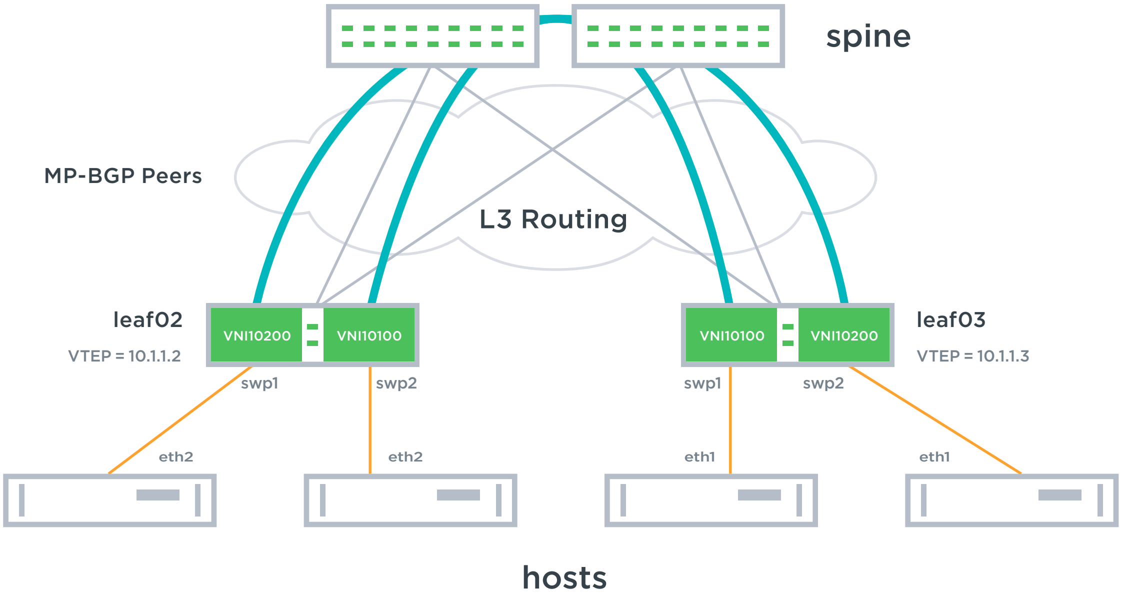


FIGURE 7 - EVPN ADVERTISING MAC+IP ADDRESS TO REMOTE VTEP

The following table outlines the host addressing scheme:

Host	VLAN	MAC address	IPv4 address	IPv6 global address
host01	20	00:03:00:11:11:02	172.16.20.1/24	fd00::21/124
host02	10	00:03:00:22:22:02	172.16.10.2/24	fd00::12/124
host03	10	00:03:00:33:33:02	172.16.10.3/24	fd00::13/124
host04	20	00:03:00:44:44:02	172.16.20.4/24	fd00::24/124

Following the host01's route through the network, leaf02 learns host01's MAC address (00:03:00:11:11:02) It can be seen here:

```
cumulus@leaf02:~$ net show bridge mac
VLAN      Master    Interface    MAC                  TunnelDest    State       Flags
LastSeen
--------  --------  -----------  -----------------  ------------  ---------  -------------  ----------
10        bridge    bridge       0a:54:b5:bb:40:98                permanent                   00:03:54
10        bridge    swp2         00:03:00:22:22:02                                            00:00:22
10        bridge    vxlan_10     00:03:00:33:33:01                             offload        00:03:13
20        bridge    bridge       0a:54:b5:bb:40:98                permanent                   00:03:54
20        bridge    swp1         00:03:00:11:11:02                                            00:00:10
20        bridge    vxlan_20     00:03:00:44:44:01                             offload        00:03:40
untagged            vxlan_10     00:00:00:00:00:00  10.1.1.3      permanent  self           00:03:42
untagged            vxlan_10     00:03:00:33:33:01  10.1.1.3                 self, offload  00:03:13
untagged            vxlan_20     00:00:00:00:00:00  10.1.1.3      permanent  self           00:03:42
untagged            vxlan_20     00:03:00:44:44:01  10.1.1.3                 self, offload  00:03:40
untagged  bridge    swp1         44:38:39:00:00:16                permanent                   00:03:54
untagged  bridge    swp2         44:38:39:00:00:19                permanent                   00:03:54
untagged  bridge    vxlan_10     12:59:88:30:49:2b                permanent                   00:03:54
untagged  bridge    vxlan_20     0a:54:b5:bb:40:98                permanent                   00:03:54
```

In this case, we can see the local MAC address 00:03:00:11:11:02 is located in VLAN 20. The remote MAC addresses can also be seen across the tunnels. For example, host04's MAC address 00:03:00:44:44:01 in VLAN 20, is reachable through interface vxlan_20 and is behind VTEP 10.1.1.3.

The 00:00:00:00:00:00 MAC address associated with vxlan_20 and the 00:00:00:00:00:00 MAC address associated with vxlan_10 entries are added by EVPN when the VTEP is discovered. These entries are the head end replication entries and should never age out as long as a remote VTEP is active.

The EVPN arp-cache that is used for arp suppression can be seen as:

```
cumulus@leaf02:~$ net show evpn arp-cache vni 10200
VNI 10200 #ARP (IPv4 and IPv6, local and remote) 6
IP                   Type   MAC               Remote VTEP
fd00::24             remote 00:03:00:44:44:01 10.1.1.3
172.16.20.1          local  00:03:00:11:11:02
fd00::21             local  00:03:00:11:11:02
fe80::203:ff:fe44:4401 remote 00:03:00:44:44:01 10.1.1.3
172.16.20.4          remote 00:03:00:44:44:01 10.1.1.3
fe80::203:ff:fe11:1102 local  00:03:00:11:11:02
```

To propagate the local MAC+IP routes to the remote VTEP, the local MAC and IP addresses will be learned by MP-BGP, as seen in the following. The type 2 routes are advertising the MAC+IP addresses, and the type 3 routes are advertising the location of the VTEPs in the network. For brevity, we will display only VNI 10200, which shows communication between host01 and host04. The addresses associated with host01 are highlighted. To view the entire output, download the demo here and perform the command *"net show bgp evpn route"*

```
cumulus@leaf02:~$ net show bgp evpn route vni 10200
BGP table version is 0, local router ID is 10.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
   Network          Next Hop            Metric LocPrf Weight Path
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]
                    10.1.1.2                         32768 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[32]:[172.16.20.1]
                    10.1.1.2                         32768 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fd00::21]
                    10.1.1.2                         32768 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fe80::203:ff:fe11:1102]
                    10.1.1.2                         32768 i
*  [2]:[0]:[0]:[48]:[00:03:00:44:44:01]
                    10.1.1.3                             0 65000 65003 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]
                    10.1.1.3                             0 65000 65003 i
*  [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[32]:[172.16.20.4]
                    10.1.1.3                             0 65000 65003 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[32]:[172.16.20.4]
                    10.1.1.3                             0 65000 65003 i
*  [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fd00::24]
                    10.1.1.3                             0 65000 65003 i
```

(continued)

```
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fd00::24]
                    10.1.1.3                               0 65000 65003 i
*  [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fe80::203:ff:fe44:4401]
                    10.1.1.3                               0 65000 65003 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fe80::203:ff:fe44:4401]
                    10.1.1.3                               0 65000 65003 i
*> [3]:[0]:[32]:[10.1.1.2]
                    10.1.1.2                           32768 i
*  [3]:[0]:[32]:[10.1.1.3]
                    10.1.1.3                               0 65000 65003 i
*> [3]:[0]:[32]:[10.1.1.3]
                    10.1.1.3                               0 65000 65003 i
 Displayed 10 prefixes (15 paths)
```

The routes are separated per tenant (VNI), and is identified by the route distinguishers in a full output. The local routes naturally have no AS path, whereas the remote ones do show the AS path to the MAC+IP address and VTEP IP addresses.

From here, the local routes are advertised to the remote BGP neighbor (usually a spine in the case of eBGP) and then propagated to the remote leaf. The eBGP EVPN output on the same VNI from the remote leaf looks like the following:

```
 vagrant@leaf03:~$ net show bgp evpn route vni 10200
 BGP table version is 0, local router ID is 10.1.1.3
 Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
 Origin codes: i - IGP, e - EGP, ? - incomplete
 EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
 EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
    Network          Next Hop            Metric LocPrf Weight Path
*  [2]:[0]:[0]:[48]:[00:03:00:11:11:02]
                    10.1.1.2                               0 65000 65002 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]
                    10.1.1.2                               0 65000 65002 i
*  [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[32]:[172.16.20.1]
                    10.1.1.2                               0 65000 65002 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[32]:[172.16.20.1]
                    10.1.1.2                               0 65000 65002 i
*  [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fd00::21]
                    10.1.1.2                               0 65000 65002 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fd00::21]
                    10.1.1.2                               0 65000 65002 i
*  [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fe80::203:ff:fe11:1102]
                    10.1.1.2                               0 65000 65002 i
*> [2]:[0]:[0]:[48]:[00:03:00:11:11:02]:[128]:[fe80::203:ff:fe11:1102]
                    10.1.1.2                               0 65000 65002 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]
                    10.1.1.3                           32768 i
```

(continued)

```
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[32]:[172.16.20.4]
                    10.1.1.3                          32768 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fd00::24]
                    10.1.1.3                          32768 i
*> [2]:[0]:[0]:[48]:[00:03:00:44:44:01]:[128]:[fe80::203:ff:fe44:4401]
                    10.1.1.3                          32768 i
*  [3]:[0]:[32]:[10.1.1.2]
                    10.1.1.2                              0 65000 65002 i
*> [3]:[0]:[32]:[10.1.1.2]
                    10.1.1.2                              0 65000 65002 i
*> [3]:[0]:[32]:[10.1.1.3]
                    10.1.1.3                          32768 i
 Displayed 10 prefixes (15 paths)
```

As seen above, 00:03:00:11:11:02/172.16.20.1 and fd::21 are now remote addresses with 2 paths, as expected.

Based upon the configured import route targets, BGP then places certain routes within specific VNIs. For example, in this case, we have an import route target of <any>:10200 to be imported into VNI 10200, and an import route-target of <any>:10100 to be imported into VNI 10100, so all the MAC+IP addresses with the same route target will be imported into the respective VNI.

```
cumulus@leaf03:~$ net show bgp evpn import-rt
Route-target: 0:10200
List of VNIs importing routes with this route-target:
  10200
Route-target: 0:10100
List of VNIs importing routes with this route-target:
  10100
```

Finally, looking at leaf03’s forwarding database, those MAC addresses are now reachable through the VXLAN tunnels. They are identified per VLAN and/or VXLAN:

```
cumulus@leaf03:~$ net show bridge mac
VLAN     Master    Interface    MAC                TunnelDest    State      Flags
LastSeen
--------  --------  -----------  -----------------  ------------  ---------  -------------  ----------
10       bridge    bridge       26:cc:6c:96:ba:14                permanent                  00:06:12
10       bridge    swp1         00:03:00:33:33:01                                           00:00:10
10       bridge    vxlan _ 10   00:03:00:22:22:02                           offload         00:06:10
20       bridge    bridge       26:cc:6c:96:ba:14                permanent                  00:06:12
```

(continued)

```
20        bridge    swp2         00:03:00:44:44:01                                       00:00:08
20        bridge    vxlan_20     00:03:00:11:11:02                         offload       00:06:08
untagged            vxlan_10     00:00:00:00:00:00 10.1.1.2     permanent  self          00:06:10
untagged            vxlan_10     00:03:00:22:22:02 10.1.1.2                self, offload 00:06:10
untagged            vxlan_20     00:00:00:00:00:00 10.1.1.2     permanent  self          00:06:10
untagged            vxlan_20     00:03:00:11:11:02 10.1.1.2                self, offload 00:06:08
untagged  bridge    swp1         44:38:39:00:00:24              permanent                00:06:12
untagged  bridge    swp2         44:38:39:00:00:20              permanent                00:06:12
untagged  bridge    vxlan_10     66:7a:ba:48:57:21              permanent                00:06:12
untagged  bridge    vxlan_20     26:cc:6c:96:ba:14              permanent                00:06:12
```

We can also look at the MAC addresses per VNI:

```
cumulus@leaf03:~$ net show evpn mac vni all
VNI 10200 #MACs (local and remote) 2
MAC               Type   Intf/Remote VTEP      VLAN
00:03:00:11:11:02 remote 10.1.1.2
00:03:00:44:44:01 local  swp2                 20
VNI 10100 #MACs (local and remote) 2
MAC               Type   Intf/Remote VTEP      VLAN
00:03:00:22:22:02 remote 10.1.1.2
00:03:00:33:33:01 local  swp1                 10
```

Leaf03's EVPN ARP cache is seen as:

```
cumulus@leaf03:~$ net show evpn arp-cache vni all
VNI 10200 #ARP (IPv4 and IPv6, local and remote) 6
IP                      Type   MAC               Remote VTEP
fd00::24               local   00:03:00:44:44:01
172.16.20.1           remote   00:03:00:11:11:02 10.1.1.2
fd00::21               remote  00:03:00:11:11:02 10.1.1.2
fe80::203:ff:fe44:4401 local   00:03:00:44:44:01
172.16.20.4           local    00:03:00:44:44:01
fe80::203:ff:fe11:1102 remote  00:03:00:11:11:02 10.1.1.2
VNI 10100 #ARP (IPv4 and IPv6, local and remote) 6
IP                      Type   MAC               Remote VTEP
fe80::203:ff:fe22:2202 remote 00:03:00:22:22:02 10.1.1.2
172.16.10.3           local   00:03:00:33:33:01
fd00::12               remote 00:03:00:22:22:02 10.1.1.2
fe80::203:ff:fe33:3301 local  00:03:00:33:33:01
172.16.10.2           remote  00:03:00:22:22:02 10.1.1.2
fd00::13               local  00:03:00:33:33:01
```

As clearly seen above, EVPN is able to learn and exchange MAC+IP addresses via the MP-BGP routing protocol while keeping tenant separation.

If ARP suppression is turned on, the local leaf, having the remote MAC address and IP address, is able to respond to a server's ARP request, thus reducing broadcast traffic throughout the data center.

EVPN VXLAN ACTIVE-ACTIVE MODE

EVPN can also exchange control plane information in a VXLAN active-active mode environment, as depicted in Figure 8. Multi-chassis Link Aggregation Group (MLAG) is configured between the two leaf switches and a logical VTEP is configured using a shared, anycast IP address representing the VTEP. EVPN interacts with MLAG transitions and advertises and withdraws routes appropriately.

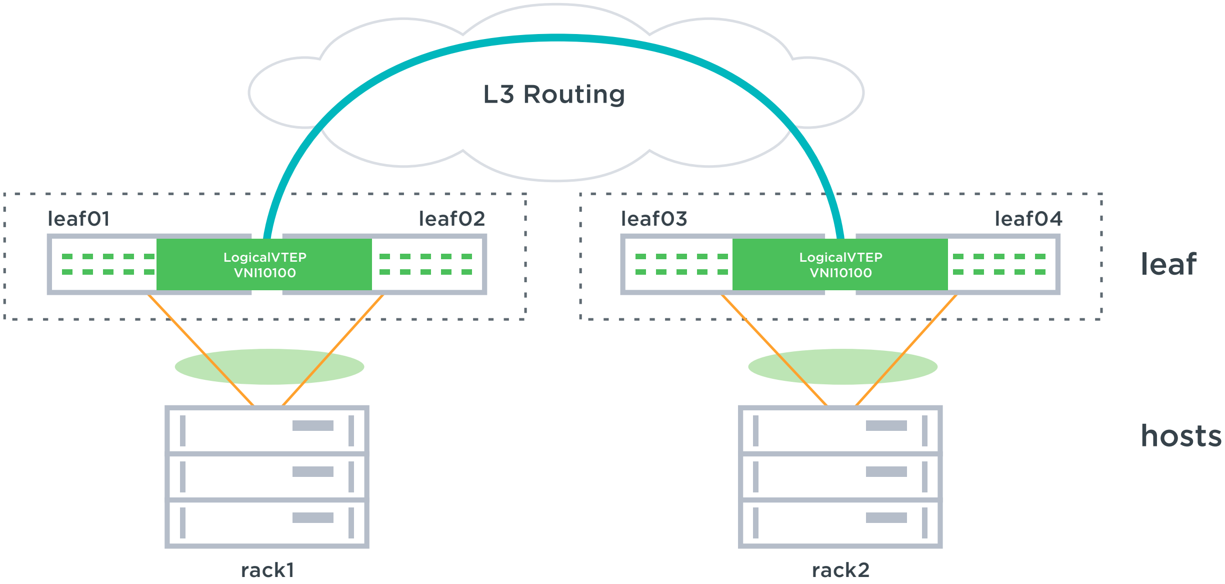


FIGURE 8 - EVPN SUPPORTS VXLAN ACTIVE-ACTIVE MODE

EVPN MAC MOBILITY

Cumulus EVPN supports a new “MAC Mobility” extended BGP community that enables quick sub-second convergence during host or VM moves within the data center. This community conveys sequence numbers along with the MAC+IP address and is advertised in the Type 2 routes.

EVPN offers fast convergence during host or VM moves in a data center

As a local switch learns a new MAC+IP address, MP-BGP sends the EVPN route to its peers without the MAC mobility community. However, upon a first move, an initial sequence number is sent in the MAC mobility community along with the update. The BGP table stores the sequence number along with the route. See Figure 9 below:

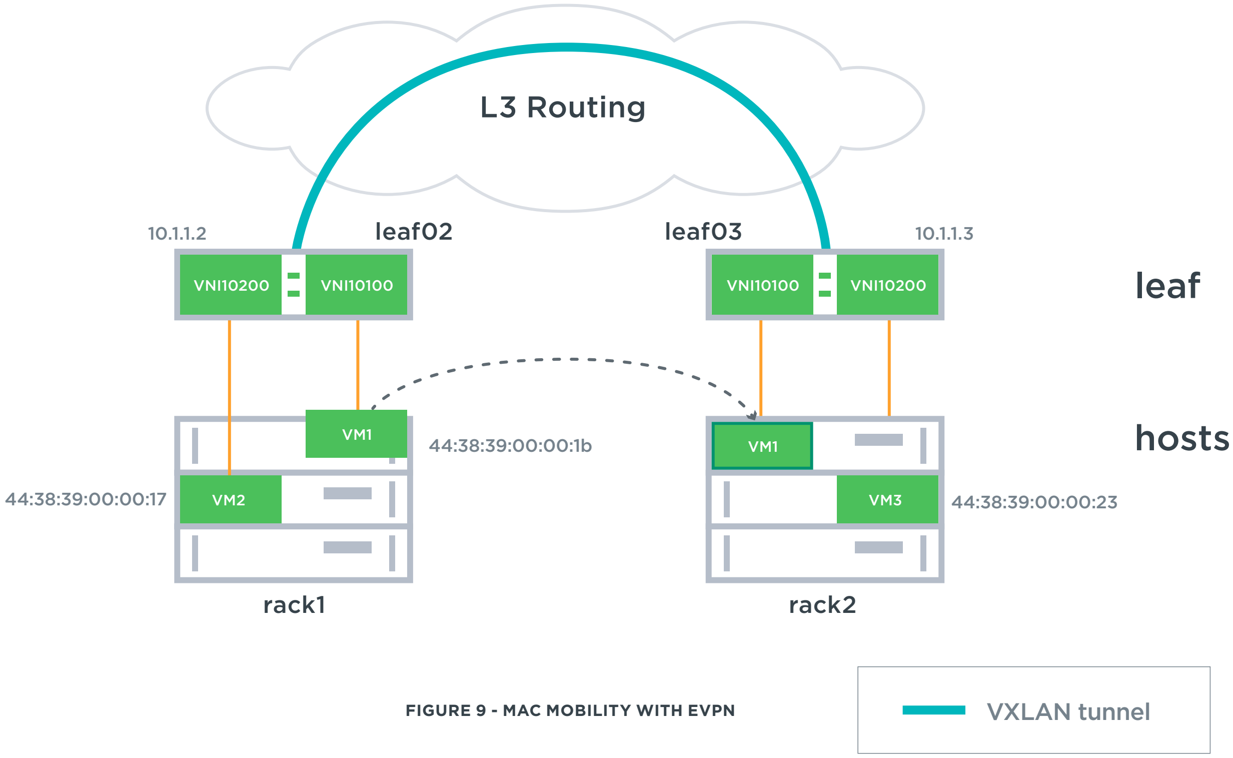


FIGURE 9 - MAC MOBILITY WITH EVPN

In this case, VM1 (MAC address 44:38:39:00:00:1b) is moved to Rack 2. Before it moves, we can see on leaf03 there are two routes to this VM, one through spine01 and one through spine02.

```
cumulus@leaf03:~$ net show bgp evpn route rd 10.1.1.2:10100 mac 44:38:39:00:00:1b
BGP routing table entry for 10.1.1.2:10100:[2]:[0]:[0]:[48]:[44:38:39:00:00:1b]
Paths: (2 available, best #2)
  Advertised to non peer-group peers:
  spine01(swp51) spine02(swp52)
  Route [2]:[0]:[0]:[48]:[44:38:39:00:00:1b] VNI 10100
  65000 65002
    10.1.1.2 from spine02(swp52) (10.10.2.2)
      Origin IGP, localpref 100, valid, external
      Extended Community: RT:65002:10100 ET:8
      AddPath ID: RX 0, TX 35
      Last update: Fri Feb  3 21:50:17 2017
 Route [2]:[0]:[0]:[48]:[44:38:39:00:00:1b] VNI 10100
  65000 65002
    10.1.1.2 from spine01(swp51) (10.10.2.1)
      Origin IGP, localpref 100, valid, external, bestpath-from-AS 65000, best
      Extended Community: RT:65002:10100 ET:8
      AddPath ID: RX 0, TX 22
      Last update: Fri Feb  3 21:11:58 2017
```

When the host or VM moves, the new local switch (in this case leaf03) learns of the change via GARP or the data plane. The local switch then installs the MAC address into its bridge forwarding database, and BGP reads it. BGP then compares the MAC address with its current BGP table. If the MAC address is already there, BGP then increments the sequence number in this community (it is assumed to be 0 if the community is not there) before advertising the address to remote peers. The remote peers similarly compare the BGP extended MAC mobility community's sequence numbers between two identical routes (the new one versus any that are already in the table), and install the route with the highest sequence number into the EVPN table, which then gets installed to the local bridge forwarding database. Use of the MAC mobility community with the sequence numbers ensure all applicable VTEPs converge quickly on the latest route to the MAC address. Below shows the output on the new local leaf (leaf03) after the move. The MAC Mobility community (MM) is now shown and the MAC address has moved once.

```
cumulus@leaf03:~$ net show bgp evpn route vni 10100 mac 44:38:39:00:00:1b
BGP routing table entry for [2]:[0]:[0]:[48]:[44:38:39:00:00:1b]
Paths: (1 available, best #1)
  Not advertised to any peer
  Route [2]:[0]:[0]:[48]:[44:38:39:00:00:1b] VNI 10100
  Local
    10.1.1.3 from 0.0.0.0 (10.1.1.3)
      Origin IGP, localpref 100, weight 32768, valid, sourced, local, bestpath-from-AS Local, best
```

(continued)

```
          Extended Community: RT:65003:10100 ET:8 MM:1
          AddPath ID: RX 0, TX 18
          Last update: Sat Feb  4 02:26:56 2017
    Displayed 1 paths for requested prefix
```

The new remote leaf (leaf02) shows the following:

```
    cumulus@leaf02:~$ net show bgp evpn route vni 10100 mac 44:38:39:00:00:1b
    BGP routing table entry for [2]:[0]:[0]:[48]:[44:38:39:00:00:1b]
    Paths: (2 available, best #2)
      Not advertised to any peer
      Route [2]:[0]:[0]:[48]:[44:38:39:00:00:1b] VNI 10100
      Imported from 10.1.1.3:10100:[2]:[0]:[0]:[48]:[44:38:39:00:00:1b]
      65000 65003
        10.1.1.3 from spine01(swp51) (10.10.2.1)
          Origin IGP, localpref 100, valid, external
          Extended Community: RT:65003:10100 ET:8 MM:1
          AddPath ID: RX 0, TX 68
          Last update: Sun Feb  5 18:35:37 2017
      Route [2]:[0]:[0]:[48]:[44:38:39:00:00:1b] VNI 10100
      Imported from 10.1.1.3:10100:[2]:[0]:[0]:[6]:[44:38:39:00:00:1b]
      65000 65003
        10.1.1.3 from spine02(swp52) (10.10.2.2)
          Origin IGP, localpref 100, valid, external, bestpath-from-AS 65000, best
          Extended Community: RT:65003:10100 ET:8 MM:1
          AddPath ID: RX 0, TX 66
          Last update: Sun Feb  5 18:35:37 2017
```

Cumulus Linux also supports MAC stickiness. When this is configured on a switch for a specific host, the MAC is prevented from moving. The configuration for sticky MACs is located in the Cumulus Linux user guide.

In the case below, a new server with MAC address 00:03:00:55:55:02 was added to leaf02 swp3 to VLAN10. It is seen that the new server's MAC address is now considered static and EVPN will not allow this MAC address to move.

```
cumulus@leaf02:~$ net show bridge macs
VLAN      Master    Interface  MAC                TunnelDest  State      Flags          LastSeen
----      --------  --------   -------            ---------   ---------  ------         -------
10        bridge    bridge     44:38:39:00:00:16              permanent                 13:49:35
10        bridge    swp2       00:03:00:22:22:02                                        never
10        bridge    swp3       00:03:00:55:55:02              static                    00:09:33
10        bridge    vxlan_10   00:03:00:33:33:01                         offload        13:12:09
20        bridge    bridge     44:38:39:00:00:16              permanent                 13:49:35
20        bridge    swp1       00:03:00:11:11:02                                        never
20        bridge    vxlan_20   00:03:00:44:44:01                         offload        13:11:55
untagged            vxlan_10   00:00:00:00:00:00  10.1.1.3    permanent  self           13:46:03
untagged            vxlan_10   00:03:00:33:33:01  10.1.1.3               self, offload  13:46:03
untagged            vxlan_20   00:00:00:00:00:00  10.1.1.3    permanent  self           13:46:03
untagged            vxlan_20   00:03:00:44:44:01  10.1.1.3               self, offload  13:46:03
untagged  bridge    swp1       44:38:39:00:00:16              permanent                 13:49:35
untagged  bridge    swp2       44:38:39:00:00:19              permanent                 13:49:35
untagged  bridge    vxlan_10   ee:ad:25:f8:73:b5              permanent                 13:49:35
untagged  bridge    vxlan_20   4e:8e:ae:ba:95:55              permanent                 13:49:35
```

We can see the same MAC address show up as a sticky MAC below. A sticky MAC always has MAC mobility (MM) set to 0:

```
cumulus@leaf02:~$ net show bgp evpn route vni 10100 mac 00:03:00:55:55:02
BGP routing table entry for [2]:[0]:[0]:[48]:[00:03:00:55:55:02]
Paths: (1 available, best #1)
  Not advertised to any peer
  Route [2]:[0]:[0]:[48]:[00:03:00:55:55:02] VNI 10100
  Local
    10.1.1.2 from 0.0.0.0 (10.1.1.2)
      Origin IGP, localpref 100, weight 32768, valid, sourced, local, bestpath-from-AS Local, best
      Extended Community: ET:8 RT:65002:10100 MM:0, sticky MAC
      AddPath ID: RX 0, TX 88
      Last update: Tue Jun  6 15:22:47 2017
```

(continued)

EVPN deployment scenarios and configuration

EVPN is used as the control plane solution for extending layer 2 connectivity across a data center using layer 3 fabric or it can be used to provide layer 2 connectivity between data centers.

Naturally, VTEPs must be configured on the leaf switches for the data plane traffic. Below is a snippet of a sample VXLAN configuration in a leaf with VXLAN active-active mode. The MLAG and layer 3 configurations are left off for brevity.

```
interface lo
   address 10.0.0.11/32
   clagd-vxlan-anycast-ip 10.0.0.20
interface swp2
   alias host facing interface
   bridge-access 10
interface swp51
   alias spine facing interface bgp unnumbered
interface vxlan _ 10
   bridge-access 10
   bridge-arp-nd-suppress on
   bridge-learning off
   mstpctl-bpduguard yes
   mstpctl-portbpdufilter yes
   vxlan-id 10100
   vxlan-local-tunnelip 10.0.0.11
interface bridge
   bridge-ports vxlan _ 1 swp2
   bridge-vids 10-20
   bridge-vlan-aware yes
interface vlan _ 10
   ip-forward off
   ip6-forward off
   vlan-id 10
   vlan-raw-device bridge
```

As seen above, the active-active mode VXLAN VNI 10100 is configured with the anycast address of 10.0.0.20. There is only one VXLAN tunnel to the remote leaf switch. To prevent data plane learning, bridge-learning is turned off. The locally connected bridge is associated with both the host facing interface (swp2) as well as the VXLAN interface (vxlan_1). A VXLAN interface and bridge interface must be configured on every switch with a desired VTEP.   For the active-active scenario, the routing protocol must advertise the anycast VTEP IP address (10.0.0.20) to remote VTEPs. More information about configuring VXLAN in active-active mode with EVPN can be found in the Cumulus Linux user guide.

The MP-BGP EVPN control plane running Cumulus Linux can be deployed in three layer 3 routed environments:

- **eBGP between the VTEPs (leafs) and spines**
- **iBGP between the VTEPs (leafs) with OSPF underlay**
- **iBGP between the VTEPs (leafs) and route reflectors (spines)**

(continued)

Although Cumulus Linux supports all options mentioned above, Cumulus Networks recommends deploying eBGP for greenfield deployments. eBGP is already the most preferred data center routing protocol for the underlay network and the same session can carry the overlay EVPN routes also.

Cumulus recommends deploying EVPN with eBGP for simplicity

EVPN IN AN EBGP ENVIRONMENT

In this scenario, you peer the leafs and the spines together as in a typical eBGP data center, activating the neighbors in the *evpn* address family. Cumulus Linux also supports eBGP unnumbered to further simplify configuration. See Figure 10.

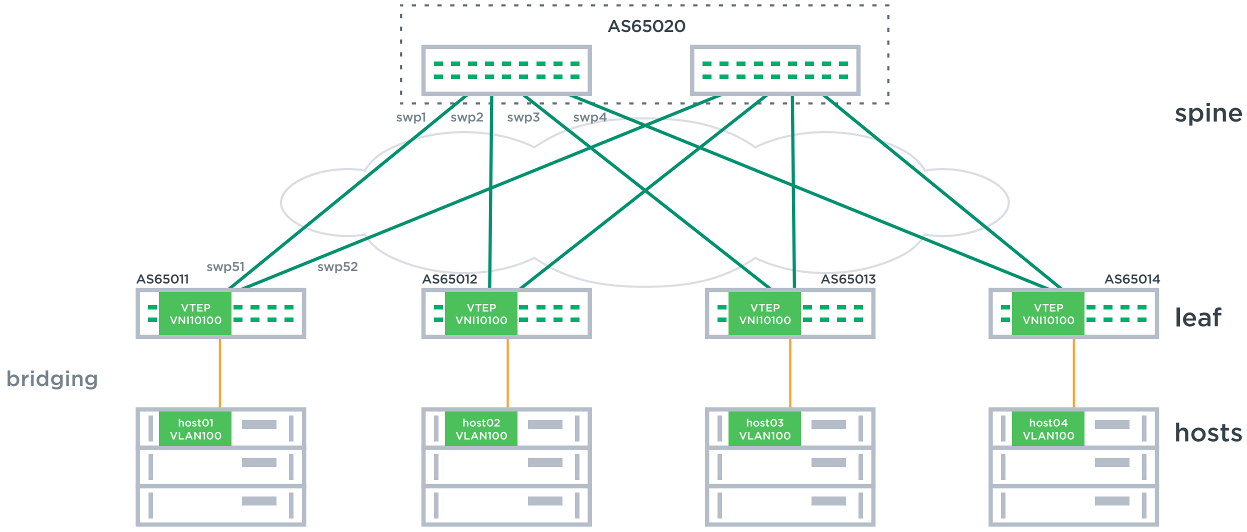


FIGURE 10 - EBGP PEERING WITH ADDRESS-FAMILY EVPN

— EBGP Peering

(continued)

Using the Figure 10 scenario with eBGP unnumbered, an example simple leaf EVPN configuration is shown below using automatic RD/RT assignment.

```
router bgp 65001
 bgp router-id 10.0.0.11
 neighbor swp51 interface remote-as external
 neighbor swp52 interface remote-as external
 !
 address-family ipv4 unicast
  network 10.0.0.11/32
 exit-address-family
 !
 address-family evpn
  neighbor swp51 activate
  neighbor swp52 activate
  advertise-all-vni
 exit-address-family
```

A sample spine configuration is shown below.

```
router bgp 65020
 bgp router-id 10.0.0.21
 neighbor swp1 interface remote-as external
 neighbor swp2 interface remote-as external
 neighbor swp3 interface remote-as external
 neighbor swp4 interface remote-as external
 !
 address-family ipv4 unicast
  network 10.0.0.21/32
 exit-address-family
 !
 address-family evpn
  neighbor swp1 activate
  neighbor swp2 activate
  neighbor swp3 activate
  neighbor swp4 activate
 exit-address-family
```

Note the EVPN address family is needed on the spines to forward the EVPN routes, but the command *advertise-all-vni* is not needed on the spines unless VTEPs are also located on the spines.

More information on configuring EVPN with eBGP can be found in the Cumulus Linux user guide.

EVPN IN AN IBGP ENVIRONMENT WITH OSPF UNDERLAY

EVPN can also be deployed with an OSPF or static route underlay if needed, but is more complex than the eBGP solution. In this case, iBGP advertises EVPN routes directly between VTEPs and the spines are unaware of EVPN or BGP. The leaf switches peer with each other in a full mesh within the EVPN address family, and generally peer to the leaf loopback addresses which is advertised in OSPF. The receiving VTEP imports routes into a specific VNI with a matching route target community.

EVPN IN AN IBGP ENVIRONMENT WITH ROUTE REFLECTORS

With this scenario, the spines are route reflectors (RR) and reflect EVPN routes between the leafs. This scenario may be necessary for scale, and/or if iBGP is desired with no OSPF underlay. The EVPN address family must be run on the spines (RRs), but the command “advertise-all-vni” is not needed. Although the RRs receive all the MAC address routes associated with the VXLANs, they are not put into hardware on the RRs allowing for greater scale.

To provide redundancy, two spine switches should be configured as RRs within the EVPN address family. It is recommended to use the same cluster-ID on the redundant route reflectors to reduce the total number of stored routes. More information on configuring RRs can be found in the Cumulus Linux user guide. See Figure 11.

If a three tier Clos network is desired without an OSPF underlay, tiers of route reflectors must be deployed.

If more than one pod is needed or the data center expands with all iBGP, the use of additional clusters is recommended. A cluster consists of one or more route reflectors and their clients. Each route reflector in each cluster peers with each other as well as any other cluster’s route reflectors. Assigning different cluster IDs (a BGP attribute) to each cluster prevents looping of routes between different clusters.

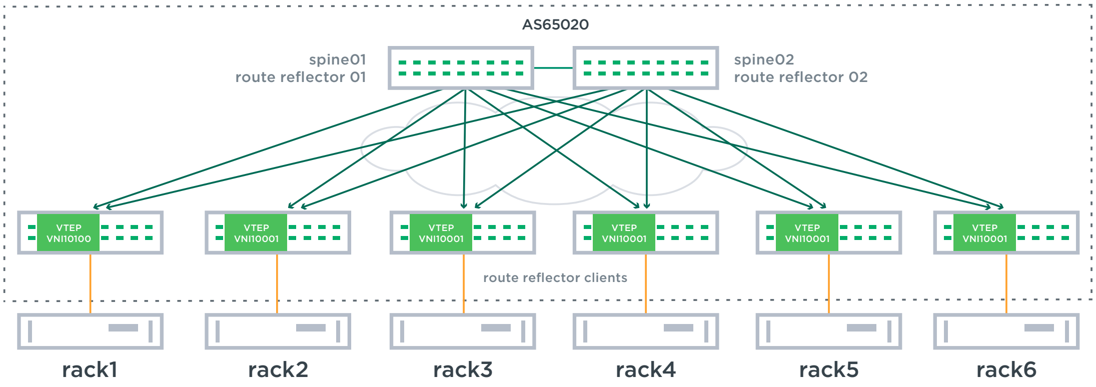


FIGURE 11 - EVPN IBGP DEPLOYMENT WITH ROUTE REFLECTORS

Conclusion

Data centers are moving towards a layer 3 fabric in order to scale, provide ease of troubleshooting, and provide redundancy with multi-vendor interoperability. However, some applications still require layer 2 connectivity. For these applications, VXLAN tunnels are being widely deployed to provide a scalable layer 2 overlay solution over a layer 3 fabric.

Cumulus EVPN is the ideal control plane solution for VXLAN tunnels. It uses the same routing protocol preferred for data center infrastructures, BGP. EVPN provides controllerless VXLAN tunnels that also scale, provide redundancy and enable fast convergence. It reduces unnecessary BUM traffic, thereby reducing overall data center traffic as well as providing multi-tenant segmentation.

Try it out for yourself on this ready-to-go demo using Cumulus VX and Vagrant.

ABOUT CUMULUS NETWORKS®

Cumulus Networks is leading the transformation of bringing web-scale networking to enterprise cloud. Its network switch, Cumulus Linux, is the only solution that allows you to affordably build and efficiently operate your network like the world's largest data center operators, unlocking vertical network stacks. By allowing operators to use standard hardware components, Cumulus Linux offers unprecedented operational speed and agility, at the industry's most competitive cost. Cumulus Networks has received venture funding from Andreessen Horowitz, Battery Ventures, Capital, Peter Wagner and four of the original VMware founders.

For more information visit cumulusnetworks.com or follow @cumulusnetworks.

07062017