VXLANs

This chapter describes how to identify and resolve problems that might occur when implementing Virtual Extensible Local Area Networks (VXLANs).

Overview

The VXLAN creates LAN segments by using an overlay approach with MAC in IP encapsulation. The encapsulation carries the original Layer 2 (L2) frame from the VM that is encapsulated from within the VEM. Each VEM is assigned an IP address which is used as the source IP address when encapsulating MAC frames to be sent on the network. You can have multiple VXLAN tunnel endpoints (VTEPs) per VEM that are used as sources for this encapsulated traffic. The encapsulation carries the VXLAN identifier which is used to scope the MAC address of the payload frame.

The connected VXLAN is indicated within the port profile configuration of the vNIC and is applied when the VM connects. Each VXLAN uses an assigned IP multicast group to carry broadcast traffic within the VXLAN segment.

When a VM attaches to a VEM, if it is the first to join the particular VXLAN segment on the VEM, an IGMP join is issued for the VXLAN's assigned multicast group. When the VM transmits a packet on the network segment, a lookup is made in the L2 table using the destination MAC of the frame and the VXLAN identifier. If the result is a match, the L2 table entry contains the remote IP address to use to encapsulate the frame and the frame is transmitted within an IP packet destined to the remote IP address. If the result is not a match (broadcast/multicast/unknown unicasts fall into this bucket), the frame is encapsulated with the destination IP address set to be the VXLAN segment's assigned IP multicast group.

When an encapsulated packet is received from the network, it is decapsulated and the source MAC address of the inner frame and VXLAN ID is added to the L2 table as the lookup key and the source IP address of the encapsulation header will be added as the remote IP address for the table entry.

VEM L3 IP Interface for VXLAN

When a VEM has a vEthernet interface connected to a VXLAN, the VEM requires at least one IP/MAC pair to terminate VXLAN packets. In this regard, the VEM acts as an IP host. The VEM only supports IPv4 addressing for this purpose.

Similar to how the VEM Layer 3 (L3) control is configured, the IP address to use for VXLAN is configured by assigning a port profile to a vtep that has the **capability vxlan** command in it.

To support carrying VXLAN traffic over multiple uplinks, or sub-groups, in server configurations where vPC-HM MAC-Pinning is required, up to four vteps with **capability vxlan** may be configured. We recommend that all the VXLAN vteps within the same KVM host are assigned to the same port profile which must have the **capability vxlan** parameter. We can also use the default gateway for different subnets through **transport ip** command.

VXLAN traffic sourced by local vEthernet interfaces is distributed between these vteps based on the source MAC address in their frames. The VEM automatically pins the multiple VXLAN vteps to separate uplinks. If an uplink fails, the VEM automatically repins the vtep to a working uplink.

When encapsulated traffic is destined to a VEM connected to a different subnet, the VEM does not use the VMware host routing table. Instead, the vtep initiates an ARP for the remote VEM IP addresses. The upstream router must be configured to respond by using the Proxy ARP feature.

Fragmentation

The VXLAN encapsulation overhead is 50 bytes. In order to prevent performance degradation due to fragmentation, the entire interconnection infrastructure between all VEMs exchanging VXLAN packets should be configured to carry 50 bytes more than what the VM vNICs are configured to send. For example, using the default vNIC configuration of 1500 bytes, the VEM uplink port profile, upstream physical switch port, and interswitch links, and any routers if present, must be configured to carry an MTU of at least 1550 bytes. If that is not possible, it is suggested that the MTU within the guest VMs be configured to be smaller by 50 bytes, For example, 1450 bytes.

If this is not configured, the VEM attempts to notify the VM if it performs Path MTU (PMTU) Discovery. If the VM does not send packets with a smaller MTU, the VM fragments the IP packets. Fragmentation only occurs at the IP layer. If the VM sends a frame that is too large to carry, after adding the VXLAN encapsulation, and the frame does not contain an IP packet, the frame is dropped.

Scalability

Maximum Number of VXLANs

The Cisco Nexus 1000V supports a total of 4000 VLANs or VXLANs or any combination adding to no more than 2048. This number matches the maximum number of ports on the Cisco Nexus 1000V. Thereby, allowing every port to be connected to a different VLAN or VXLAN.

Supported Features

This section contains the following topics:

Jumbo Frames

The Cisco Nexus 1000V supports jumbo frames as long as these requirements are met:

- There is room to accommodate the VXLAN encapsulation overhead of at least 50 bytes
- The physical switch or router infrastructure can transport these jumbo sized IP packets.

Disabling the VXLAN Feature Globally

As a safety precaution, the **no feature segmentation** command will not be allowed if there are any ports associated with a VXLAN port profile. You must remove all the associations before disabling the feature. The **no feature segmentation** command will cleanup all the VXLAN Bridge Domain configurations on the Cisco Nexus 1000V.

VXLAN Troubleshooting Commands

Use the following commands to display VXLAN attributes.

VSM Commands

To display ports belonging to a specific segment:

```
switch(config)# show system internal seg_bd info segment 10000
Bridge-domain: A
Port Count: 11
Veth1
Veth2
Veth3
```

To display the vEthernet bridge domain configuration:

```
switch(config)# show system internal seg_bd info port vethernet 1
Bridge-domain: A
segment_id = 10000
Group IP: 225.1.1.1
```

To display the vEthernet bridge configuration with ifindex as an argument:

```
switch(config)# show system internal seg_bd info port ifindex 0x1c000050
Bridge-domain: A
segment_id = 10000
Group IP: 225.1.1.1
```

To display the total number of bridge domain ports:

```
switch(config)# show system internal seg_bd info port_count
Number of ports: 11
```

To display the bridge domain internal configuration:

```
switch(config)# show system internal seg_bd info bd vxlan-home

Bridge-domain vxlan-home (2 ports in all)
Segment ID: 5555 (Manual/Active)
Group IP: 235.5.5.5
State: UP              Mac learning: Enabled
```

```
is_bd_created: Yes
current state: SEG_BD_FSM_ST_READY
pending_delete: 0
port_count: 2
action: 4
hwbd: 28
pa_count: 0
Veth2, Veth5
switch(config)#
```

To display VXLAN vEthernet information:

```
switch# show system internal seg_bd info port
if_index = <0x1c000010>
Bridge-domain vxlan-pepsi
rid = 216172786878513168
swbd = 4098

if_index = <0x1c000040>
Bridge-domain vxlan-pepsi
rid = 216172786878513216
swbd = 4098

switch#
```

Additional **show** commands:

```
show system internal seg_bd info {pss | sdb | global | all}

show system internal seg_bd {event-history | errors | mem-stats | msgs}
```

VEM Commands

To verify VXLAN vEthernet programming:

```
~ # vemcmd show port segments
                           Native  Seg
  LTL   VSM Port  Mode    SegID   State
   50      Veth5   A       5555   FWD
   51      Veth9   A       8888   FWD
~ #
```

To verify VXLAN VTEP programming:

```
~ # vemcmd show vxlan interfaces
LTL           IP       Seconds since Last
                       IGMP Query Received
(* Interface on which IGMP Joins are sent)
-----------------------------------------
 49        10.3.3.3        50         *
 52        10.3.3.6        50
~ #
Use "vemcmd show port vlans" to verify that the vteps are in the correct transport VLAN.
```

To verify bridge domain creation on the VEM:

```
~ # vemcmd show bd  bd-name vxlan-home
BD 31, vdc 1, segment id 5555, segment group IP 235.5.5.5, swbd 4098, 1 ports,
"vxlan-home"
Portlist:
     50  RedHat_VM1.eth0

~ #
```

To verify remote IP learning:

```
~ # vemcmd show l2 bd-name vxlan-home
Bridge domain   31 brtmax 4096, brtcnt 2, timeout 300
Segment ID 5555, swbd 4098, "vxlan-home"
Flags:  P - PVLAN  S - Secure  D - Drop
       Type         MAC Address   LTL   timeout   Flags    PVLAN    Remote IP
    Dynamic   00:50:56:ad:71:4e   305         2                     10.3.3.100
     Static   00:50:56:85:01:5b    50         0                      0.0.0.0

~ #
```

To display statistics:

```
~ # vemcmd show vxlan-stats
  LTL  Ucast   Mcast   Ucast   Mcast    Total
       Encaps  Encaps  Decaps  Decaps   Drops
   49       5   14265       4      15       0
   50       6   14261       4      15     213
   51       1      15       0       0      10
   52       0      11       0       0      15

~ #
```

To display detailed per-port statistics for a VXLAN vEthernet/vtep:

```
~ # vemcmd show vxlan-stats ltl 51
```

To display detailed per-port-per-bridge domain statistics for a VXLAN vtep for all bridge domains:

```
~ # vemcmd show vxlan-stats ltl <vxlan_vtep_ltl> bd-all
```

To display detailed per-port-per-bridge domain statistics for a VXLAN vtep for a specified bridge domain:

```
~ # vemcmd show vxlan-stats ltl vxlan_vtep_ltl bd-name bd-name
```

VEM Packet Path Debugging

Use the following commands to debug VXLAN traffic from a VM on VEM1 to a VM on VEM2.

- VEM1: Verify that packets are coming into the switch from the segment vEthernet.

  vempkt capture ingress ltl *vxlan_veth*

- VEM1: Verify VXLAN ecapsulation.

  vemlog debug sflisp all

  vemlog debug sfvnsegment all

- VEM1: Verify that the remote IP address has been learned.

  vemcmd show l2 bd-name *segbdname*

  If the remote IP address has not been learned, then packets are sent as encapsulated multicast packets. For example, an initial ARP request from the VM is sent in this manner.

- VEM1: Find out which uplink is being used and verify that the encapsulated packets are going out the uplink.

  vemcmd show vxlan-encap ltl *ltl*

  vempkt capture egress ltl *uplink*

- VEM1: Display VXLAN statistics and look for any failures.

  vemcmd show vxlan-stats all

  vemcmd show vxlan-stats ltl *veth/vxlanvtep*
- VEM2: Verify that encapsulated packets are arriving on the uplink.

  vempkt capture ingress ltl *uplink*
- VEM2: Verify VXLAN decapsulation.

  vemlog debug sflisp all

  vemlog debug sfvnsegment all
- VEM2: Verify that the decapsulated packets go out on the VXLAN vEthernet port.

  vempkt capture egress ltl *vxlan_veth*
- VEM2: Display VXLAN statistics and look for any failures:

  vemcmd show vxlan-stats all

  vemcmd show vxlan-stats ltl *veth/vxlanvtep*

VEM Multicast Debugging

Use the following command to debug VEM multicast issues.

- IGMP state on the VEM:

  vemcmd show igmp *vxlan_transport_vlan* **detail**

Note This command does not show any output for the segment multicast groups. To save multicast table space, segment groups are not tracked by IGMP snooping on the VEM.

- IGMP queries:

Use the **vemcmd show vxlan interfaces** command to verify that IGMP queries are being received.

- IGMP joins from the VTEP:

Use the **vempkt capture ingress ltl** *first_vxlan_vtep_ltl* command to see if the Openstack is sending **join** messages.

Use the **vempkt capture egress ltl** *uplink_ltl* command to see if the **join** messages are being sent to the upstream switch.

VXLAN Datapath Debugging

Use the commands listed in this section to troubleshot VXLAN problems.

Debugging Using the vemlog Command

Command	Result
vemlog debug sfbd all	Displays information to help debug the bridge domain setup or configuration.
vemlog debug sfporttable all	Displays information to help debug the port configuration, CBL, and vEthernet LTL pinning.
vemlog debug sfvnsegment all	Displays information for encapsulation and decapsulation setup and decisions.
vemlog debug sflisp all	Displays information about actual packet editing, VXLAN interface handling, and multicast handling.
echo "debug dpa_allplatform all" > /tmp/dpafifo	Displays multicast joins or leaves on the DPA socket.
echo “debug sfl2agent all” > /tmp/dpafifo	Displays the bridge domain configuration.
echo “debug sfportagent all” > /tmp/dpafifo	Displays debug port configuration information.
echo “debug sfportl2lisp_cache all” > /tmp/dpafifo	Displays debug hitless reconnect (HR) for capability l2-lisp information.
echo “debug sfpixmagent all” > /tmp/dpafifo	Displays debug CBL programming.

HR

To debug segment information for HR, use the following command:

echo “debug sfsegment_cache all” > /tmp/dpafifo (to debug segment info HR)

(now has details of cached and temp segment info list)

echo "show vsm cache *vsm control mac***" > /tmp/dpafifo**

Vempkt

The **vempkt** command has been enhanced to display VLAN/SegmentID. Use the **vempkt** command to trace the packet path through VEM.

- Encapsulation: Capture ingress on Seg-VEth LTL – Egress on uplink
- Decapsulation: Capture ingress on uplink – Egress on Seg-VEth LTL

Statistics

Command	Result
vemcmd show vxlan-stats	Displays a summary of per-port statistics.
vemcmd show vxlan-stats ltl *vxlan_vtep_ltl*	Displays detailed per-port statistics for VXLAN vtep.
vemcmd show vxlan-stats ltl *vxlan_veth_ltl*	Displays detailed per-port statistics for vEthernet in a VXLAN.

Command	Result
vemcmd show vxlan-stats ltl *vxlan_vtep_ltl* **bd-all**	Displays detailed per-port-per-bridge domain statistics for a VXLAN vtep for all bridge domains.
vemcmd show vxlan-stats ltl *vxlan_vtep_ltl* **bd-name** *bd-name*	Displays detailed per-port-per-bridge domain statistics for a VXLAN vtep for the specified bridge domain.
vemcmd show vxlan-encap ltl *vxlan_veth_ltl*	Displays which VXLAN vtep is used for encap and subsequent pinning to uplink PC for static MAC learned on port.
vemcmd show vxlan-encap mac *vxlan_vm_mac*	Displays which VXLAN vtep is used for encapsulation and subsequent pinning to uplink PC.

Show Commands

Command	Result
vemcmd show vxlan interfaces	Displays the VXLAN encapsulated interfaces.
vemcmd show port vlans	Checks the port programming and CBL state for the bridge domain.
vemcmd show bd	Displays the bridge domain segmentId/group/list of ports.
vemcmd show bd bd-name *bd-name-string*	Displays one segment bridge domain.
vemcmd show l2 all	Displays the remote IP being learned.
vemcmd show l2 bd-name *bd-name-string*	Displays the Layer 2 table for one segment bridge domain.
vemcmd show arp all	Displays the IP-MAC mapping for the outer encapsulated header.