



ACI Deep Dive

Intra fabric forwarding and Tools

18th Jun 2019

Roland Ducomble – rducombl@cisco.com

Cisco TS Technical Leader – ACI Solution Support Team

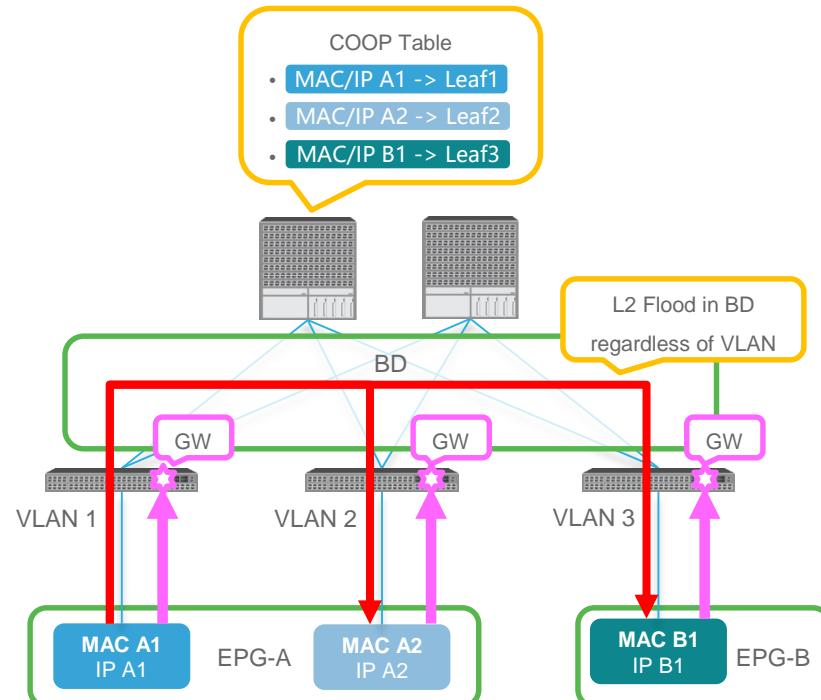
CCIE 3745

V3.11

ACI very high level forwarding summary

ACI Forwarding Basic and Important New Concepts

- End Point (EP)
 - All hosts data (MAC/IP) are handled as an EP.
- End Point Group (EPG)
 - Group of EPs which decides who can talk to who from security perspective (contract)
- Bridge Domain
 - L2 forwarding domain in ACI is Bridge Domain (BD) which could have multiple subnets on it
- Pervasive Gateway
 - Each Leaf could be a default gateway for directly attached EndPoints
- COOP
 - All EndPoints MAC/IP are stored in COOP table in each Spine
- Spine Proxy
 - If ingress Leaf doesn't know the destination, Leaf sends packet to one of Spines for proxy

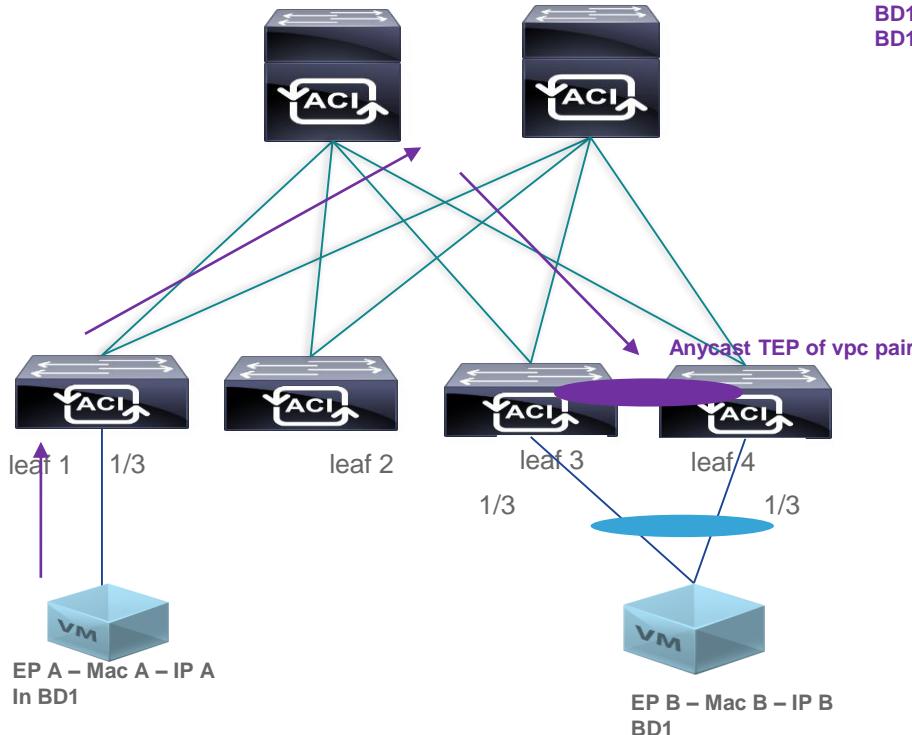


L2 only – 2 EP in same BD

Leaf 1 – Table
BD1 – Mac A → Local 1/3
BD1 – Mac B → Tu 34 to vpc TEP

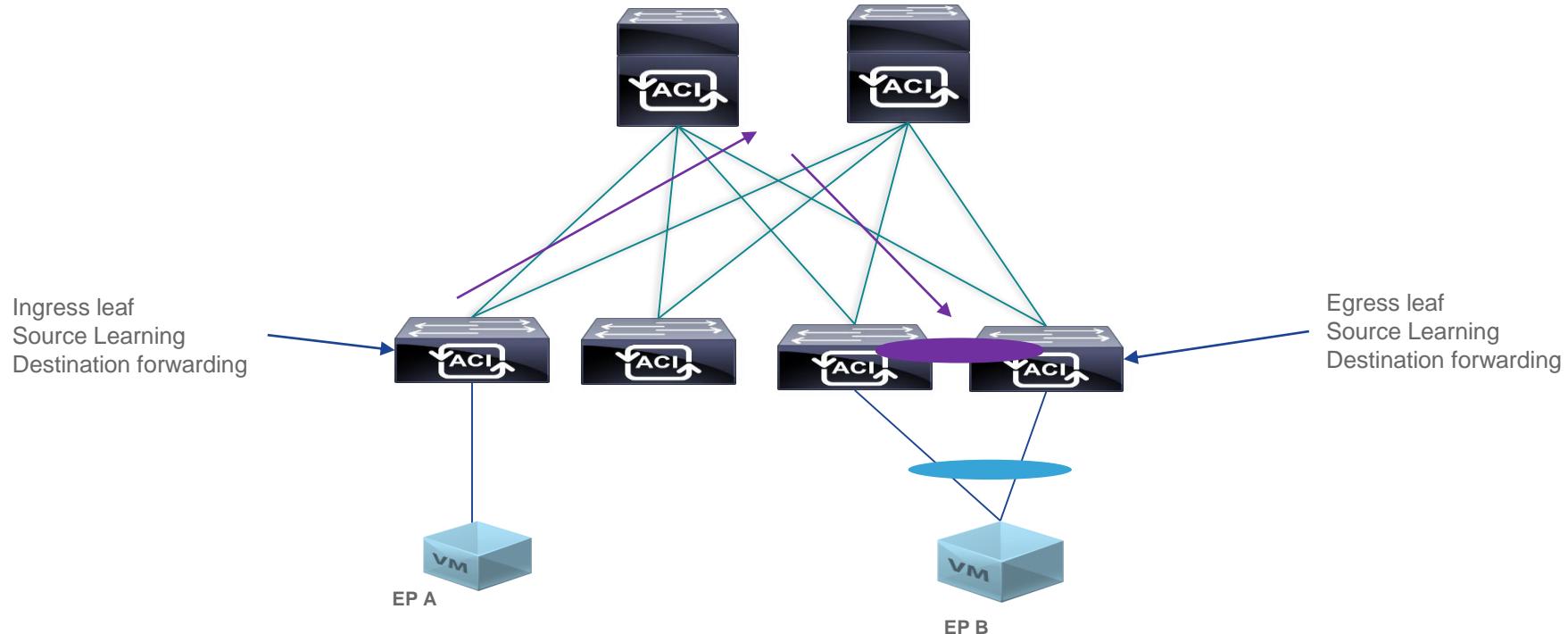
Packet inside fabric
Outer IP – Src Leaf1 TEP → Acast vpc34 TEP
Vnid – BD VNID

Leaf 3 and 4 Table
BD1 – Mac A → Tunnel 1 to Leaf1 TEP
BD1 – Mac B → Po3



High level forwarding from EP A to EP B

Packet inside fabric - VXLAN
Outer Src IP – Outer Dest IP – VXLAN VNID – Inner packet



L3 – 2 EP In different BD/Subnet Forwarding

a) assume IP are known already

Leaf 1 – Table

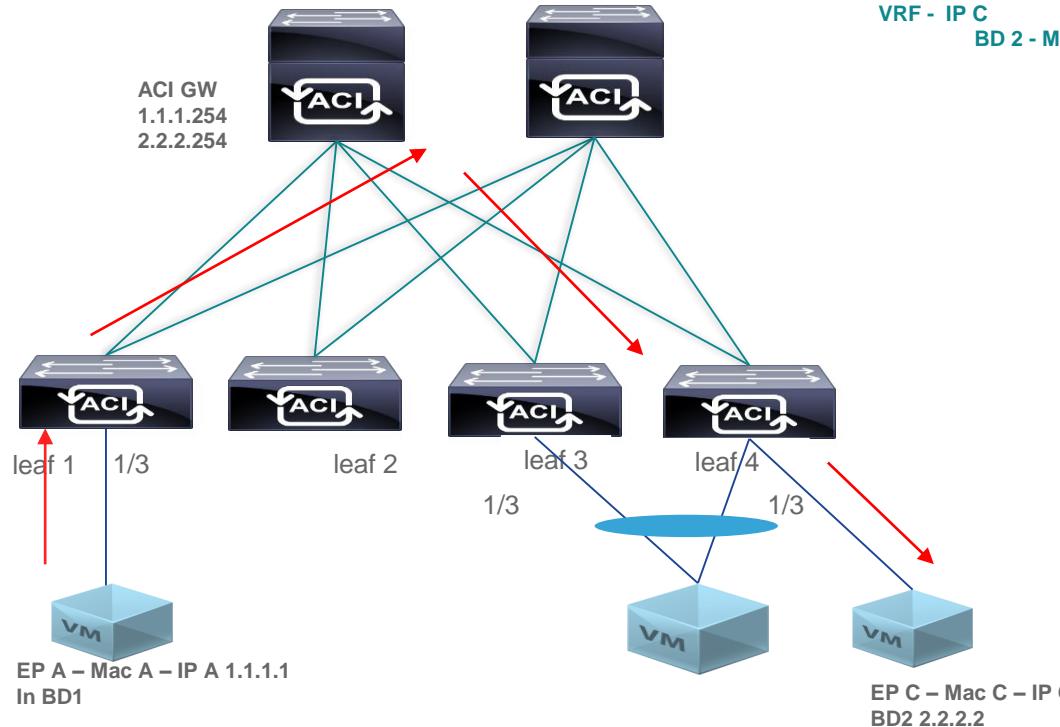
VRF – IP C → Tunnel 4 to Leaf 4 TEP

Packet inside fabric
Outer IP – Src Leaf1 TEP → Leaf 4 TEP
Vnid – VRF VNID

Leaf 4 Table

VRF - IP C

BD 2 - MAC C → 1/3

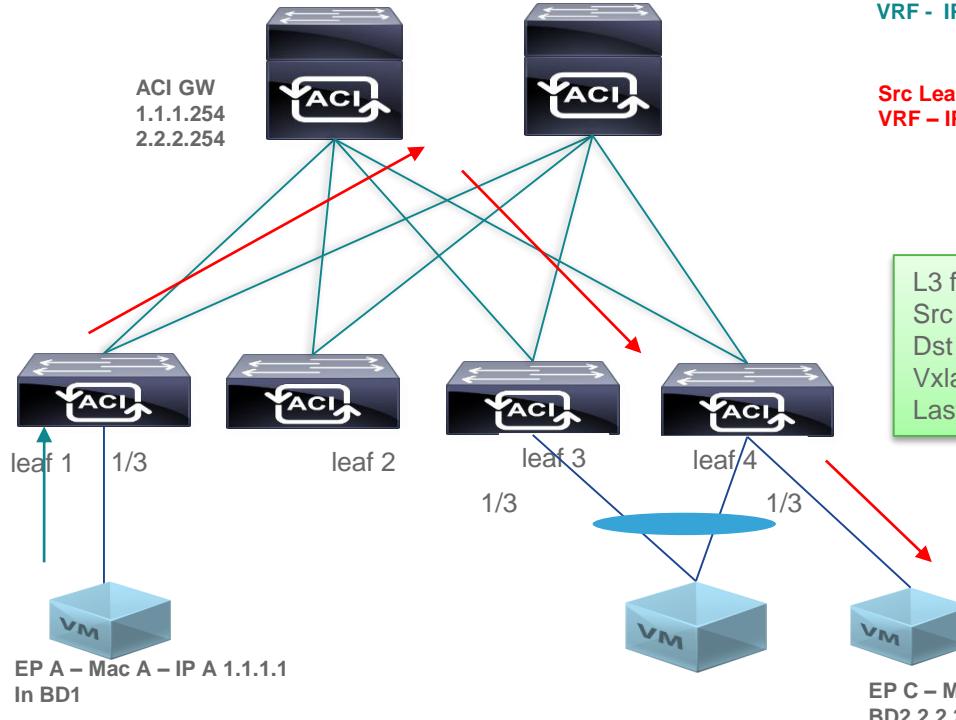


L3 – 2 EP In different BD/Subnet Learning

**Dest Lookup :
Leaf 1 – Table
VRF – IP C → Tunnel 4 to Leaf 4 TEP**

Src learning: VRF – IP A

Packet inside fabric
Outer IP – Src Leaf1 TEP → Leaf 4 TEP
Vnid – VRF VNID



Leaf 4 Table VRF - IP C BD 2 - MAC C → 1/3

Src Learning : VRF – IP A → Tunnel 1 to Leaf 1 TEP

- L3 forwarding
- Src IP/MAC learning in VRF/BD
- Dst IP forwarding in VRF
- Vxlan packet tagged with VRF VNID
- Last hop leaf makes MAC Rewrite

L3 – 2 EP In different BD/Subnet Forwarding

b) assume Dest IP is not in table

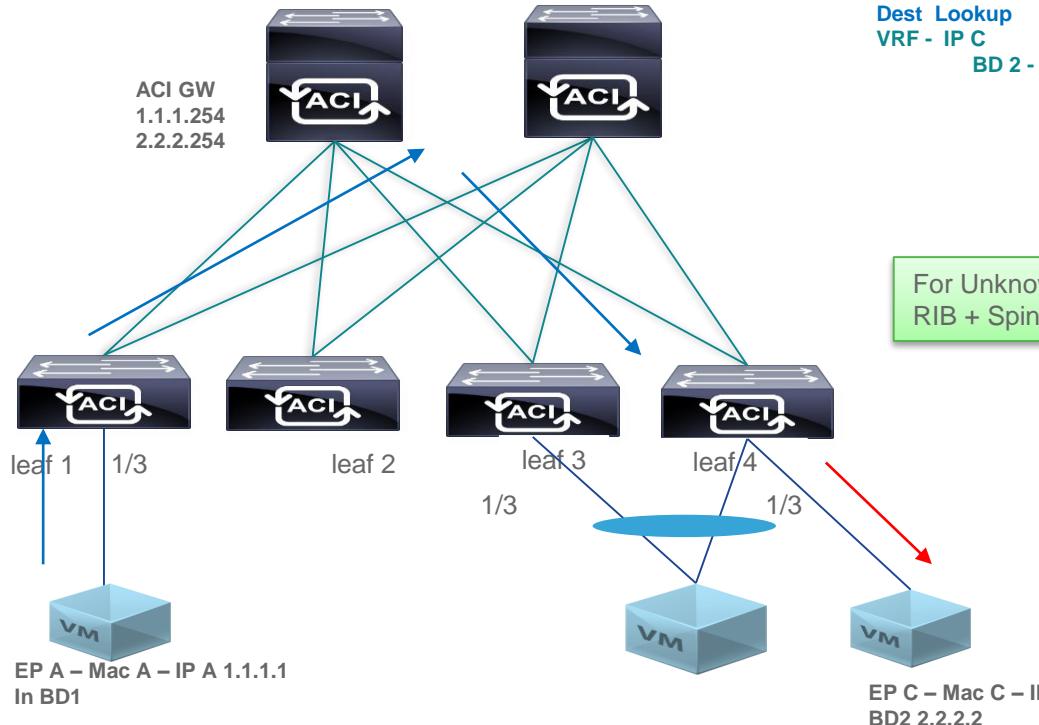
Leaf 1 – Table
Table miss IP C unknown

Show ip route vrf XXX
2.2.2.0/24 → NH
→ Anycast spine IP

Packet inside fabric
Outer IP – Src Leaf1 TEP → Anycast Spine
Vnid – VRF VNID

Spine :
COOP Lookup IP C → LEaf4
Or
ARP for IP C in BD 2

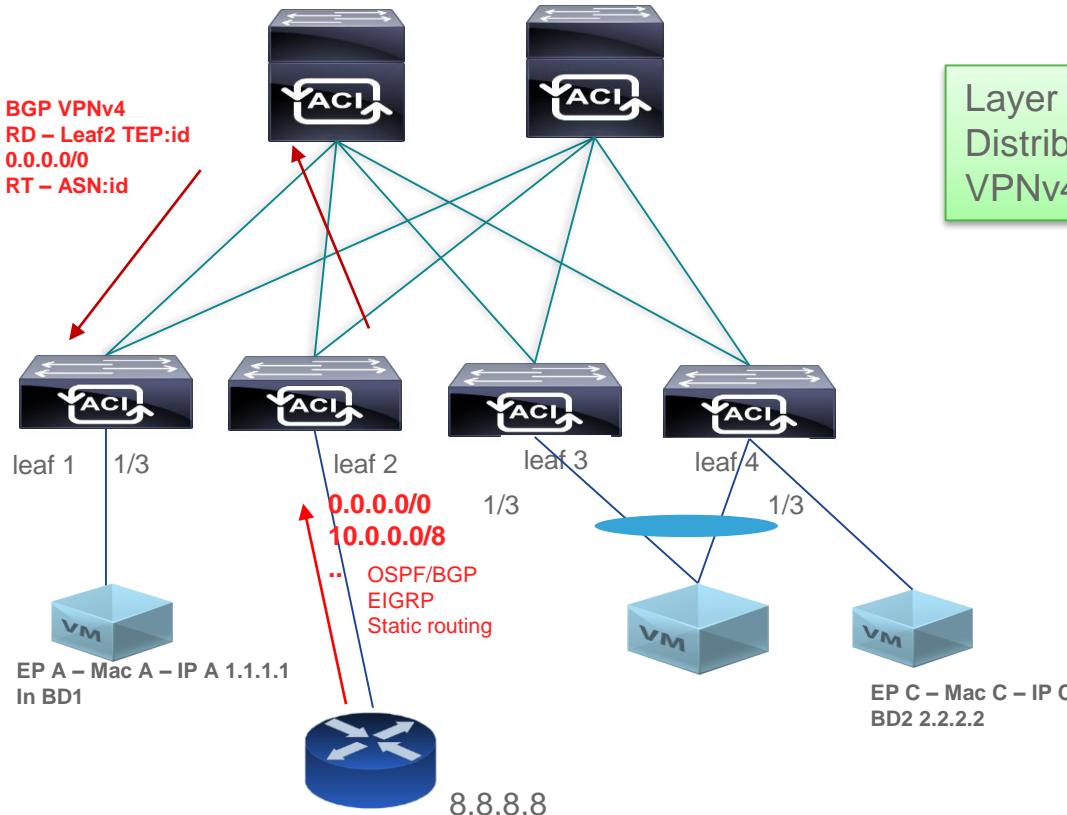
Leaf 4 Table
Dest Lookup
VRF - IP C
BD 2 - MAC C → 1/3



L3 – L3 out Route Propagation

Show ip route vrf XXX
2.2.2.0/24 → NH
→ Anycast spine IP

0.0.0.0/0
→ TEP Leaf 2
10.0.0.0/8
→ TEP Leaf 2



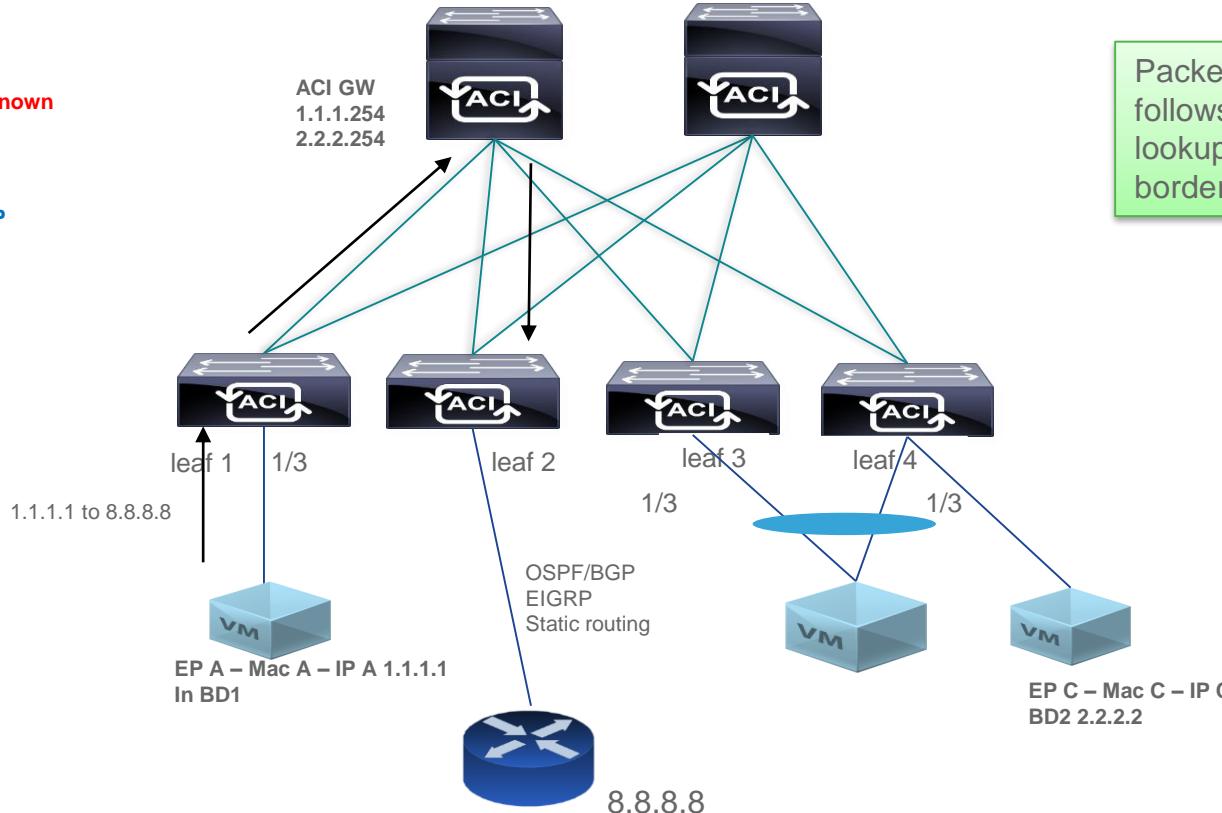
L3 – EP to Layer 3 out Data path

Leaf 1 – Table
Table miss 8.8.8.8 unknown

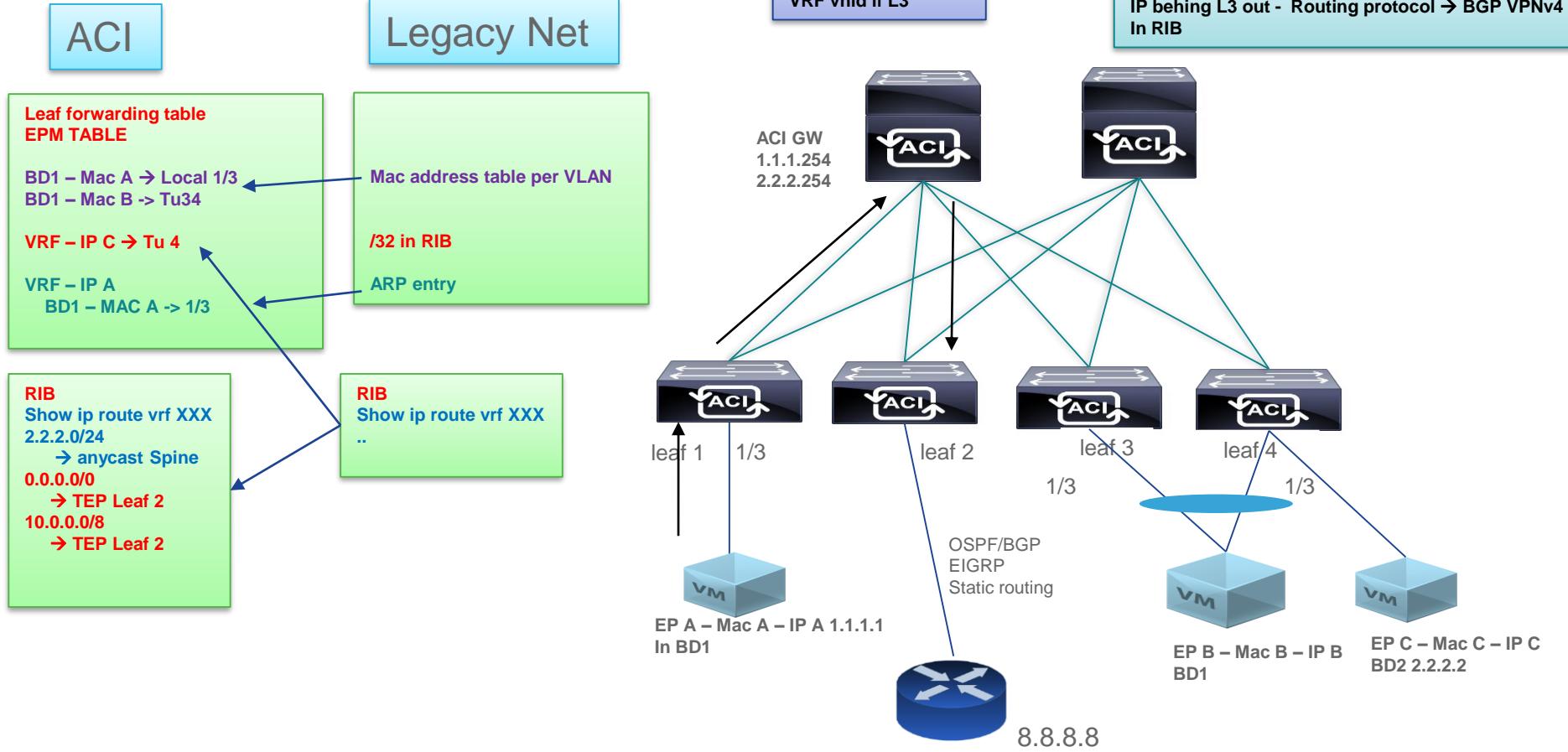
Show ip route vrf XXX
2.2.2.0/24 → NH
→ Anycast spine IP

0.0.0.0/0
→ TEP Leaf 2
10.0.0.0/8
→ TEP Leaf 2

Packet inside fabric
Outer IP – Src Leaf1 TEP → Leaf 2 TEP
Vnid – VRF VNID



All together





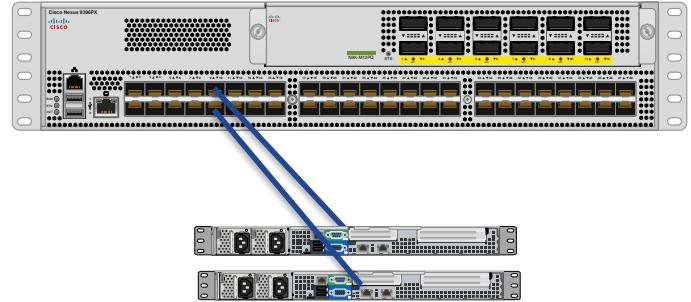
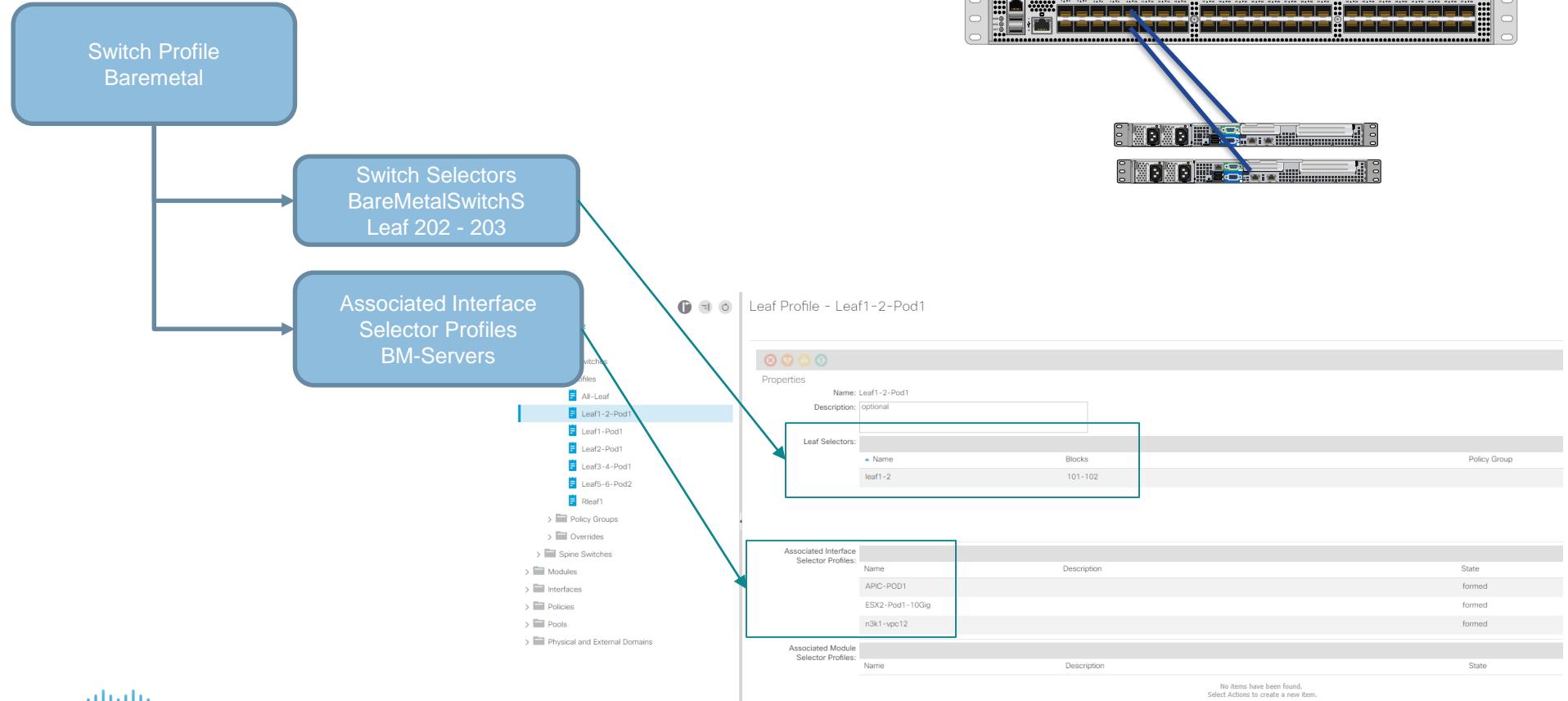
Access Policies – Domain

Physical Domain	VLAN Pool
Policy Group	
Tenant	Interface Policy
Switch Policies	Profiles
EPG	Application Profile
Attachable Access Entity Profile	Bridge domain

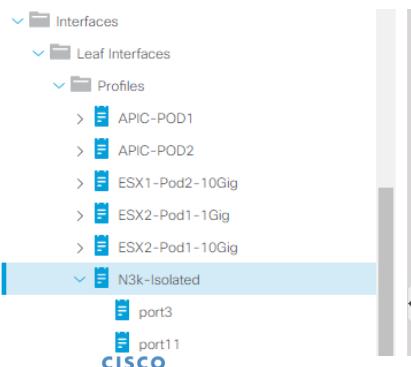
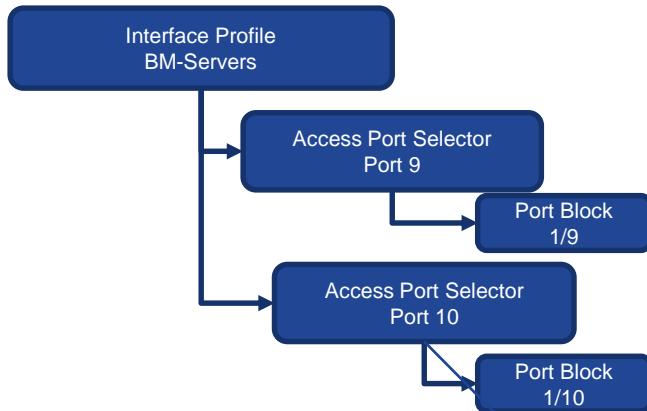
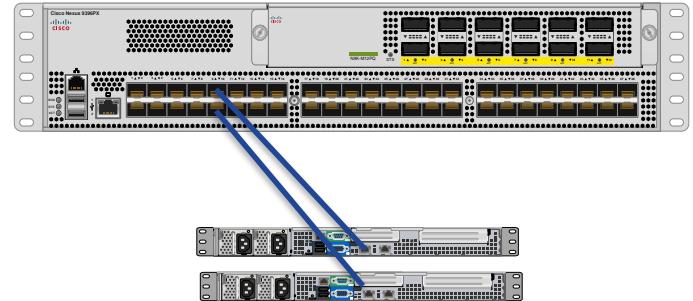
Selector

Which Port will I configure together
?

First we select the switches !



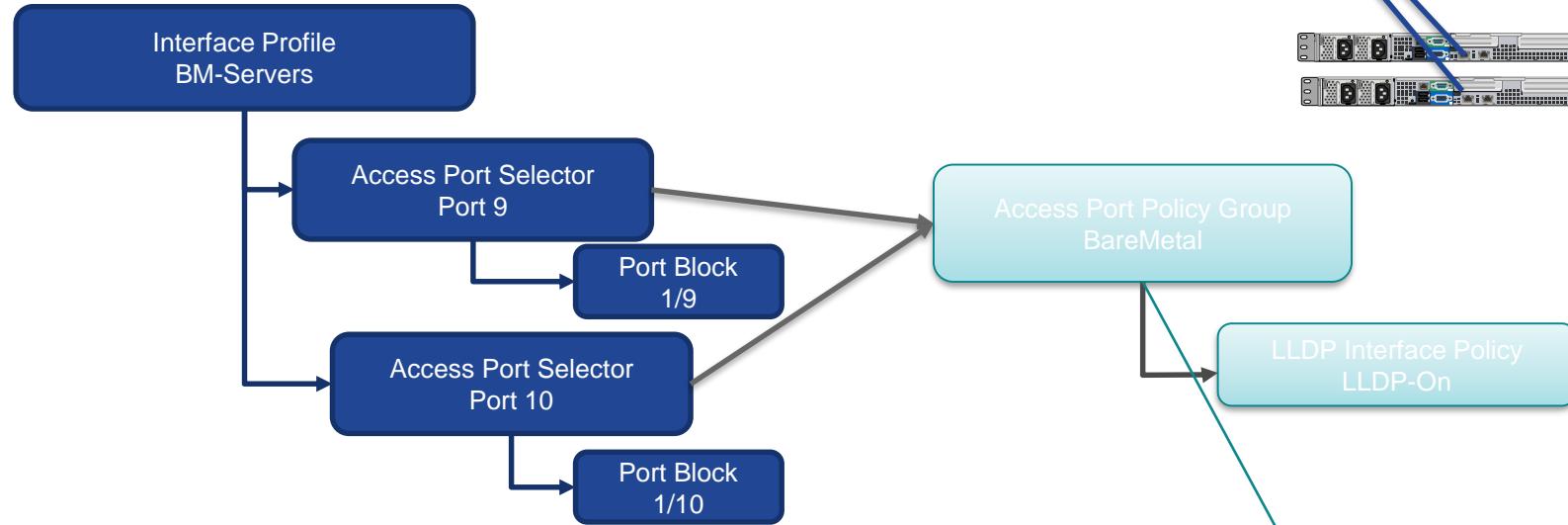
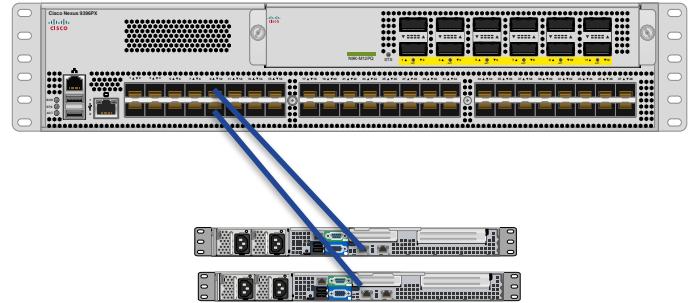
second, we define the port



A screenshot of the "Properties" dialog for an interface selector named "N3k-Isolated". The "Name" field is set to "N3k-Isolated". The "Description" field contains "optional". The "Alias" field is empty. The "Interface Selectors" table lists two entries:

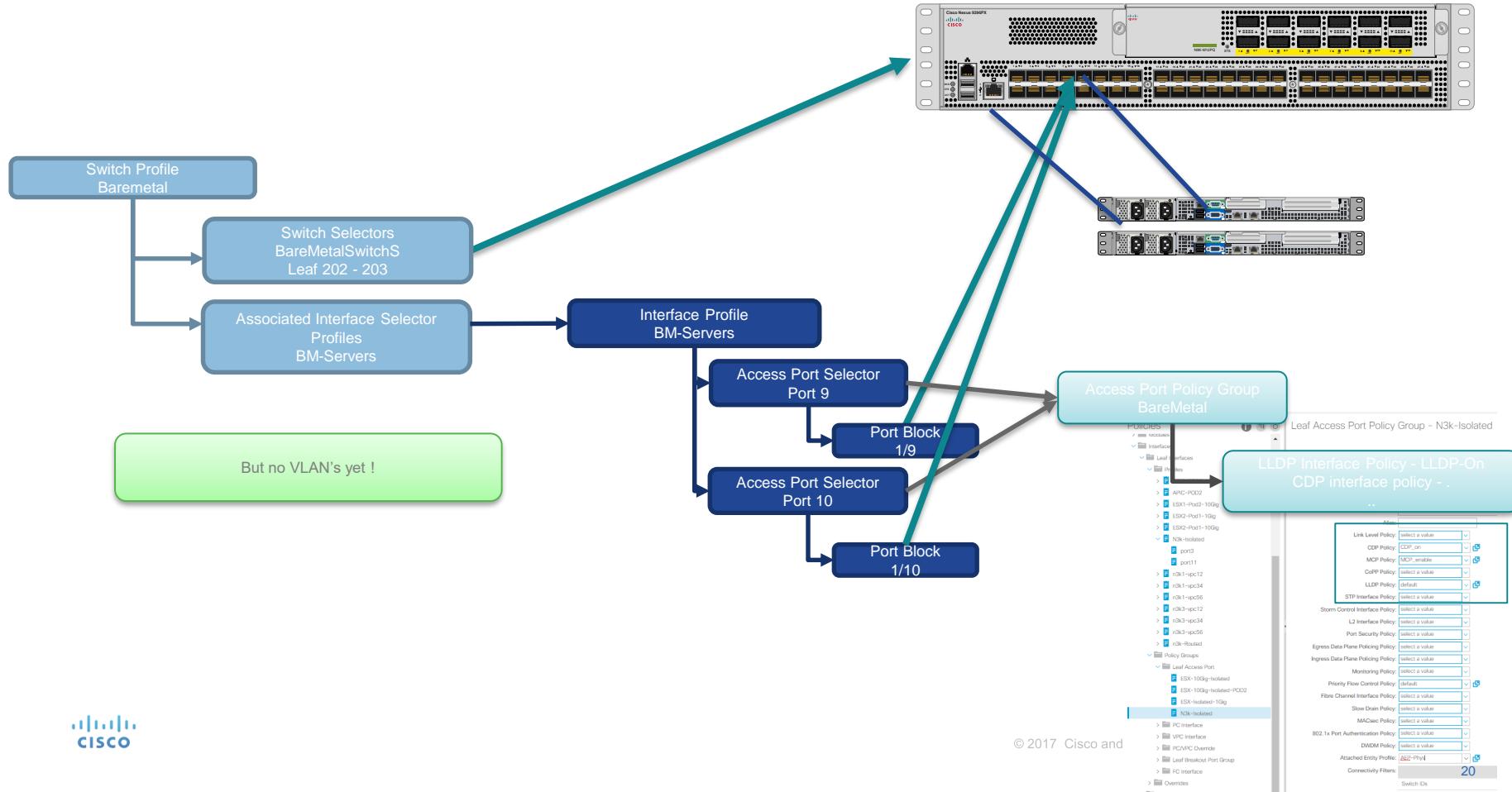
Name	Blocks	Policy Group
port11	1/11	N3k-Isolated
port3	1/3	N3k-Isolated

Then, we add properties to the port



Name	Blocks
port11	1/1
port3	1/3

Now our switch port goes up



Domain and vlan pool

How to connect to VLANs ?



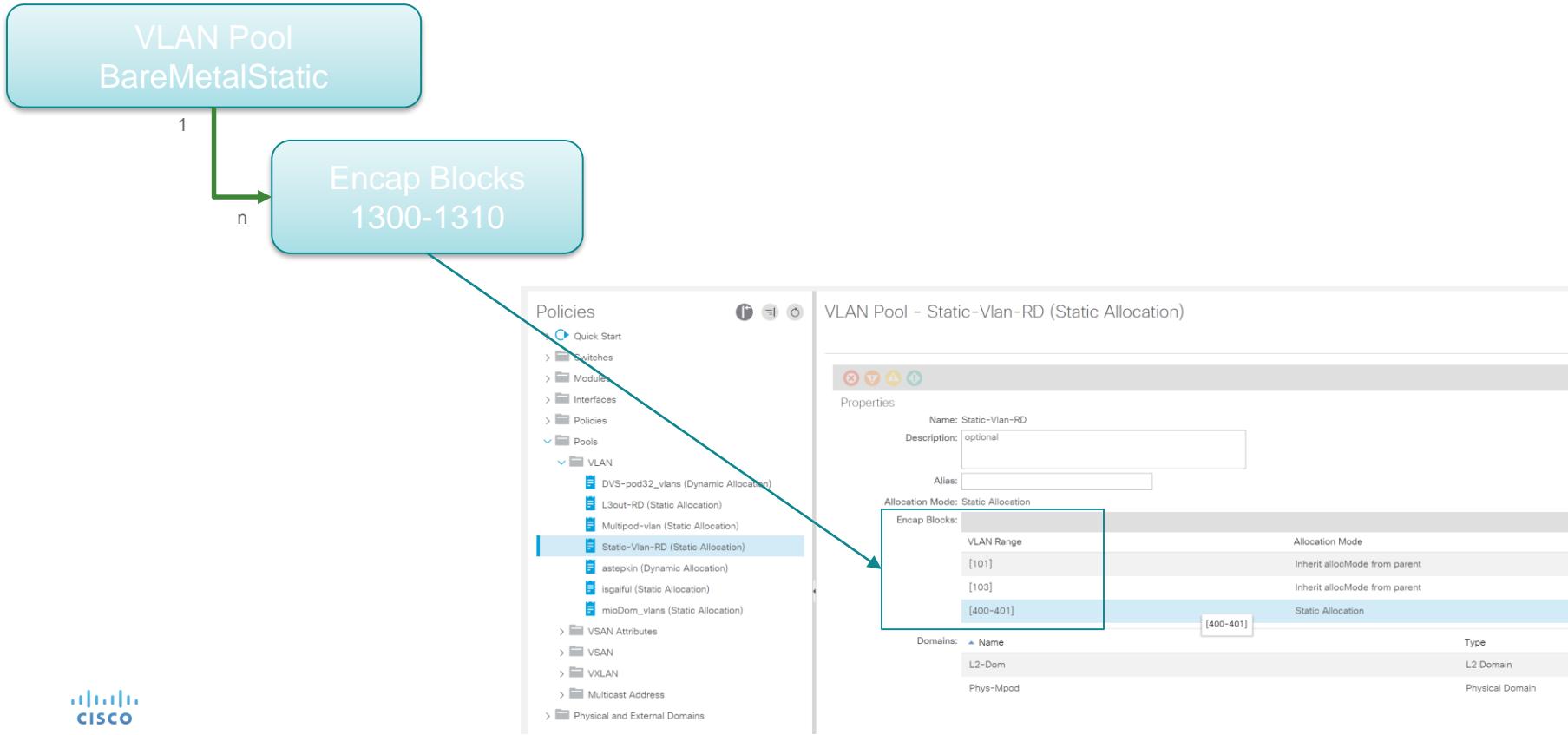
The screenshot shows the configuration interface for the **Leaf Access Port Policy Group - N3k-isolated**. The left pane displays a hierarchical list of policies, and the right pane shows the properties for the selected policy group.

Policies (Left Pane):

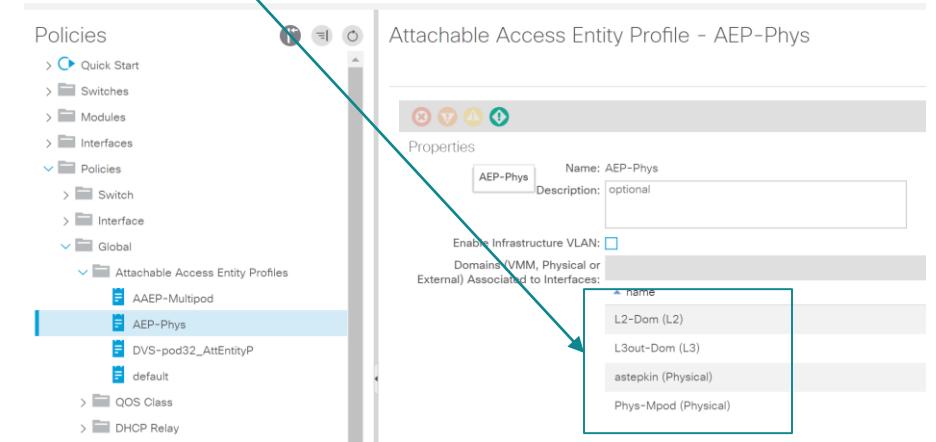
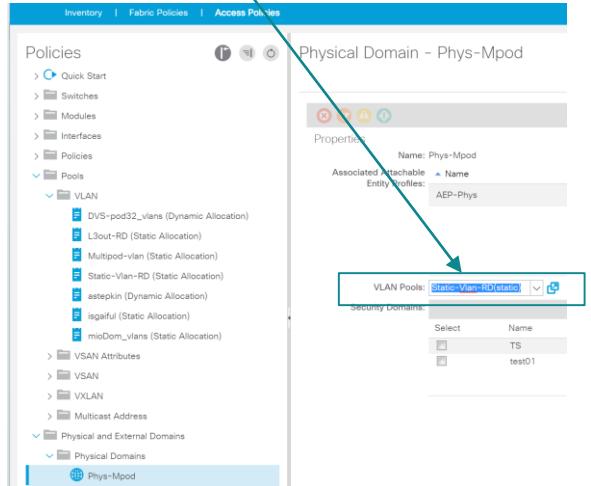
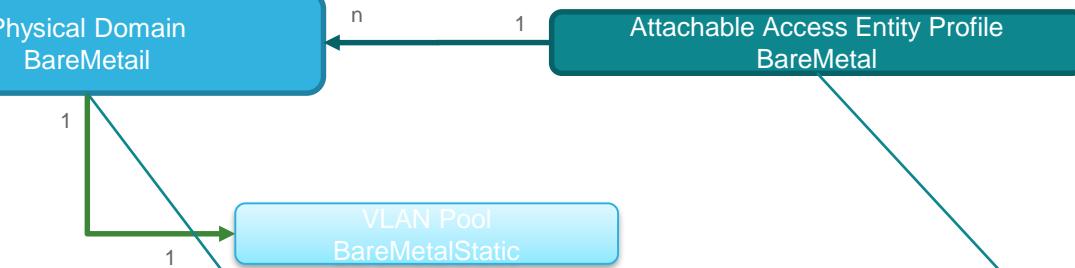
- Interfaces
 - Profiles
 - APIC-POD1
 - APIC-POD2
 - ESX1-Po02-10Gig
 - ESX2-Po01-10Gig
 - ESX2-Po01-10Gig
 - N3k-isolated
 - port3
 - port11
 - r3k1-vpc12
 - r3k1-vpc34
 - r3k1-vpc56
 - r3k3-vpc12
 - r3k3-vpc34
 - r3k3-vpc56
 - r3k-Routed
 - Policy Groups
 - Leaf Access Port
 - ESX-10Gig-isolated
 - ESX-10Gig-isolated-POD2
 - ESX-isolated-10G
 - N3k-isolated
 - PC Interface
 - VPC Interface
 - PC/VPC Override
 - Leaf Breakout Port Group
 - FC Interface
 - Overrides
 - Spine Interfaces

© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

VLANs belong in VLAN pools



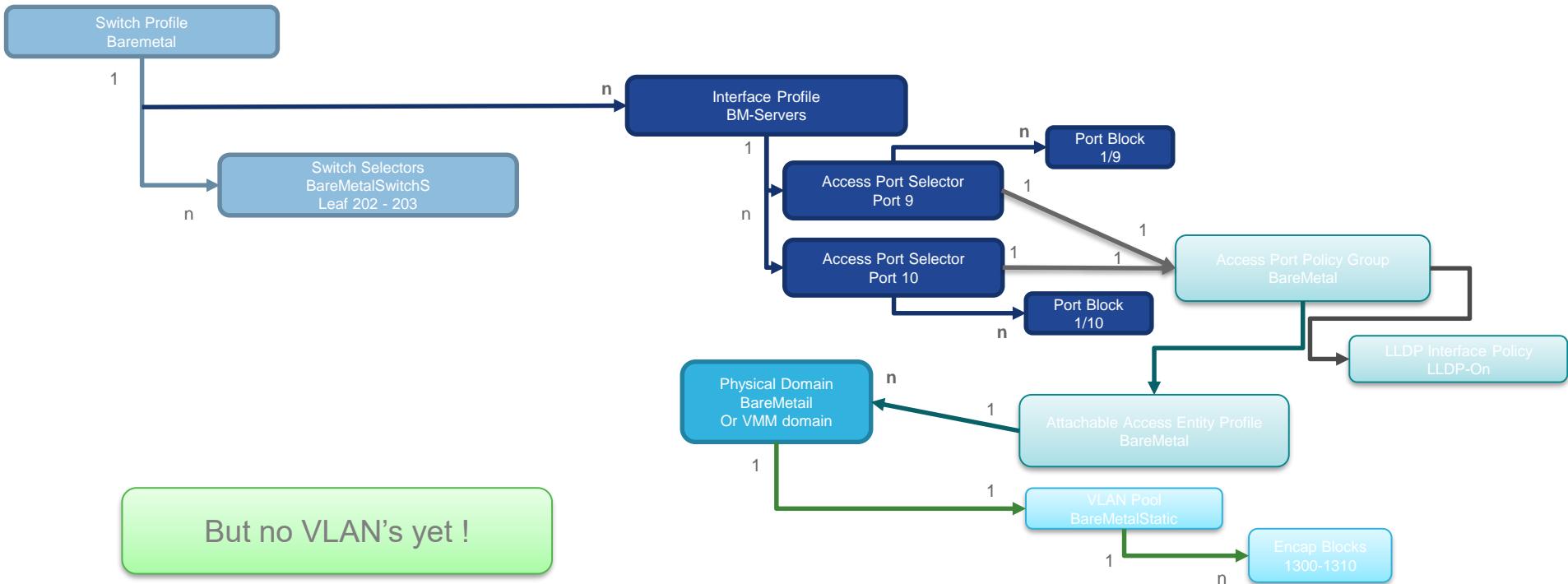
Physical domains glue it all together



Physical domain policies contain physical infrastructure specifications, such as ports and VLAN, used by a tenant or endpoint group.

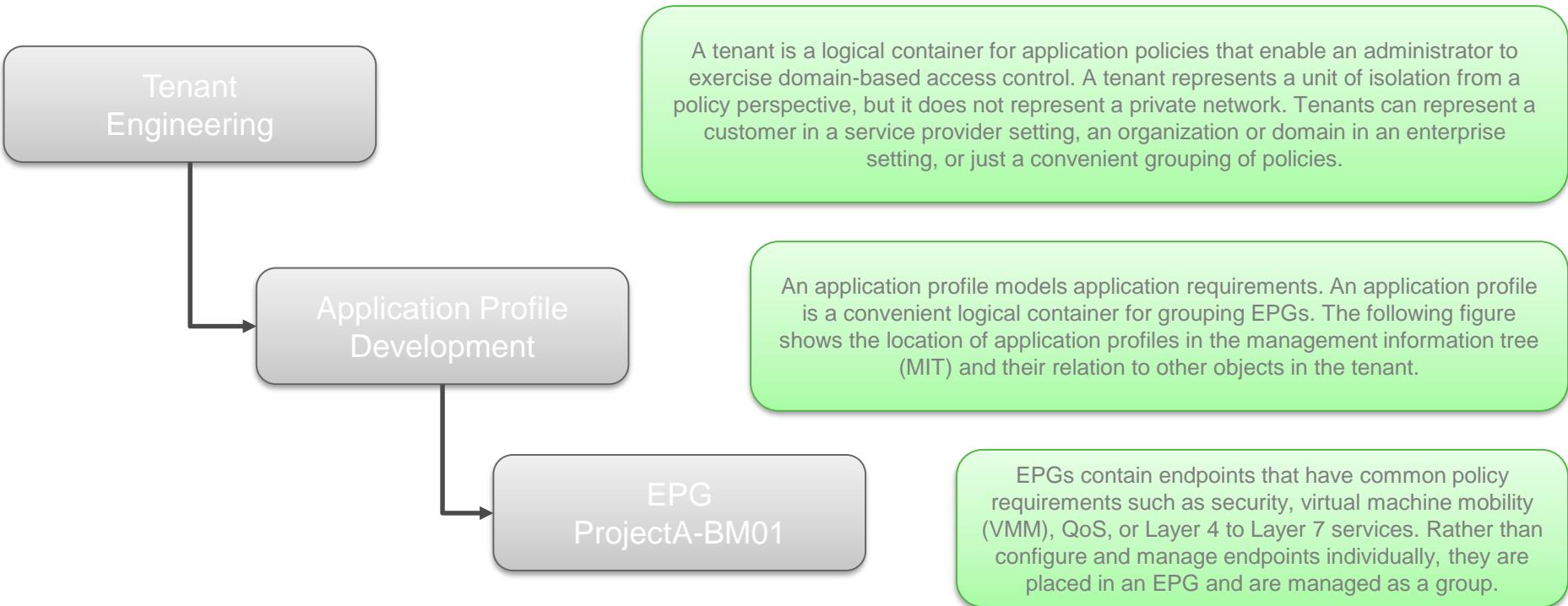


All together now

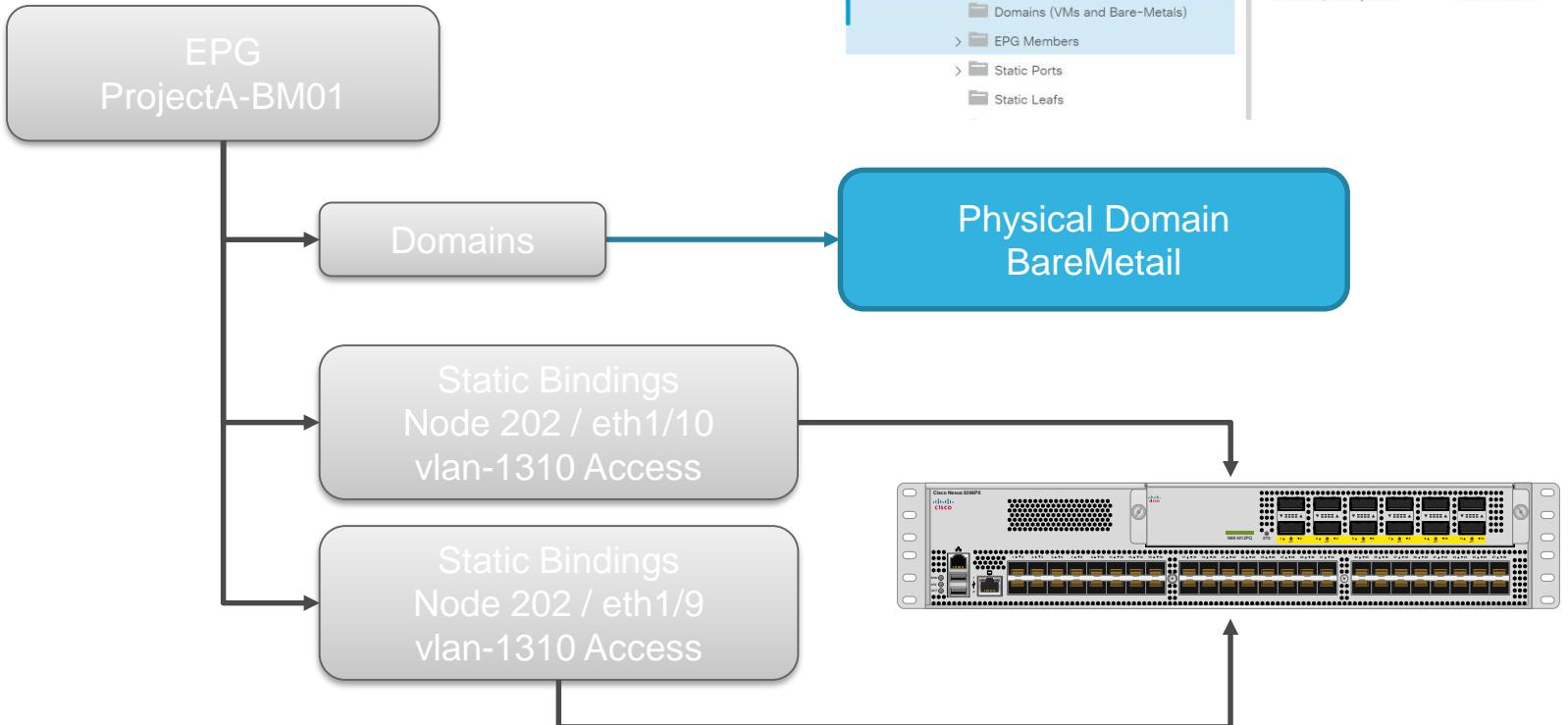
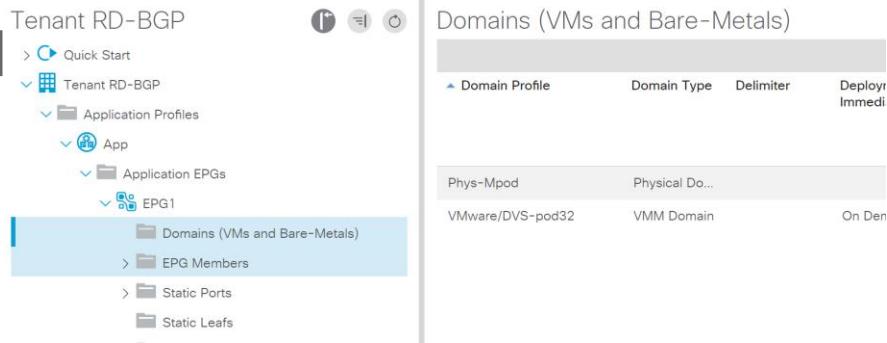


EPG domain and physical domain

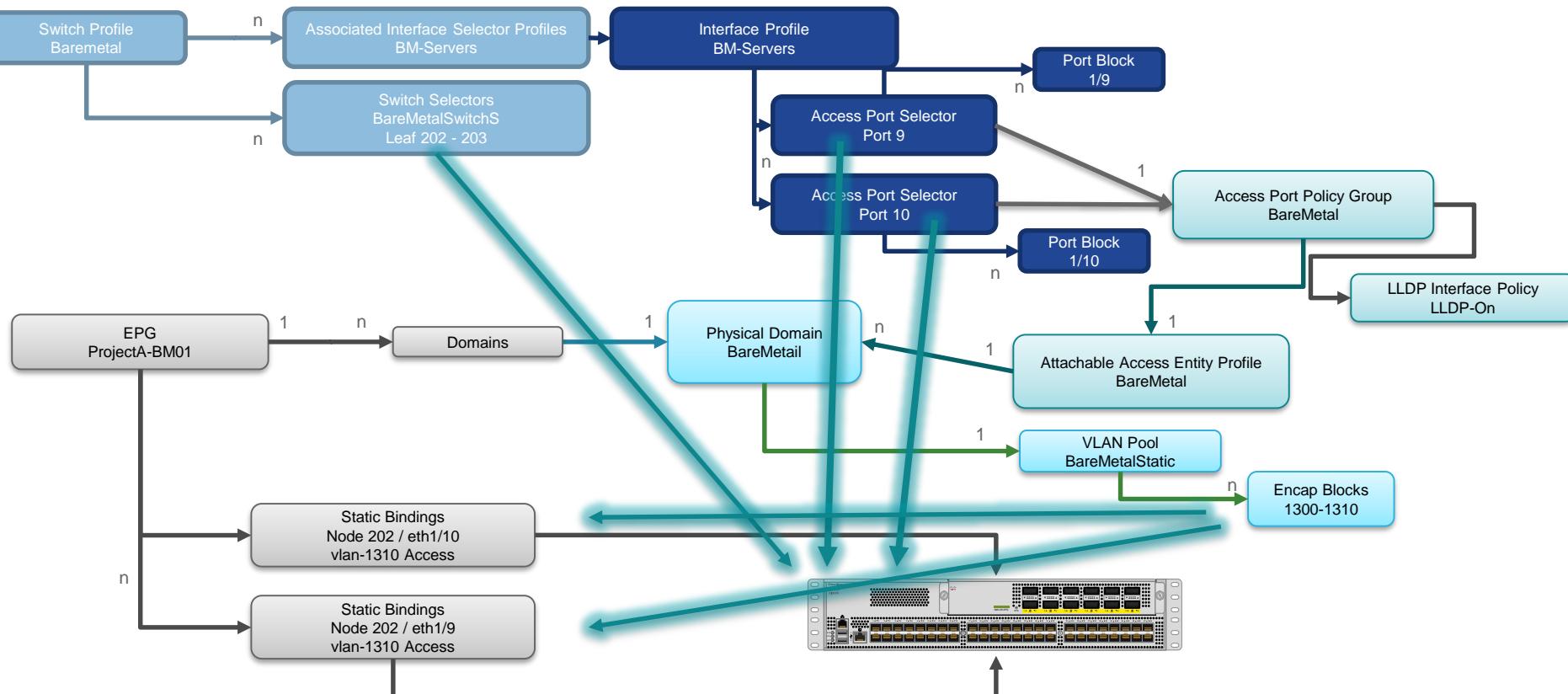
Tenant / Application Profile / EPG



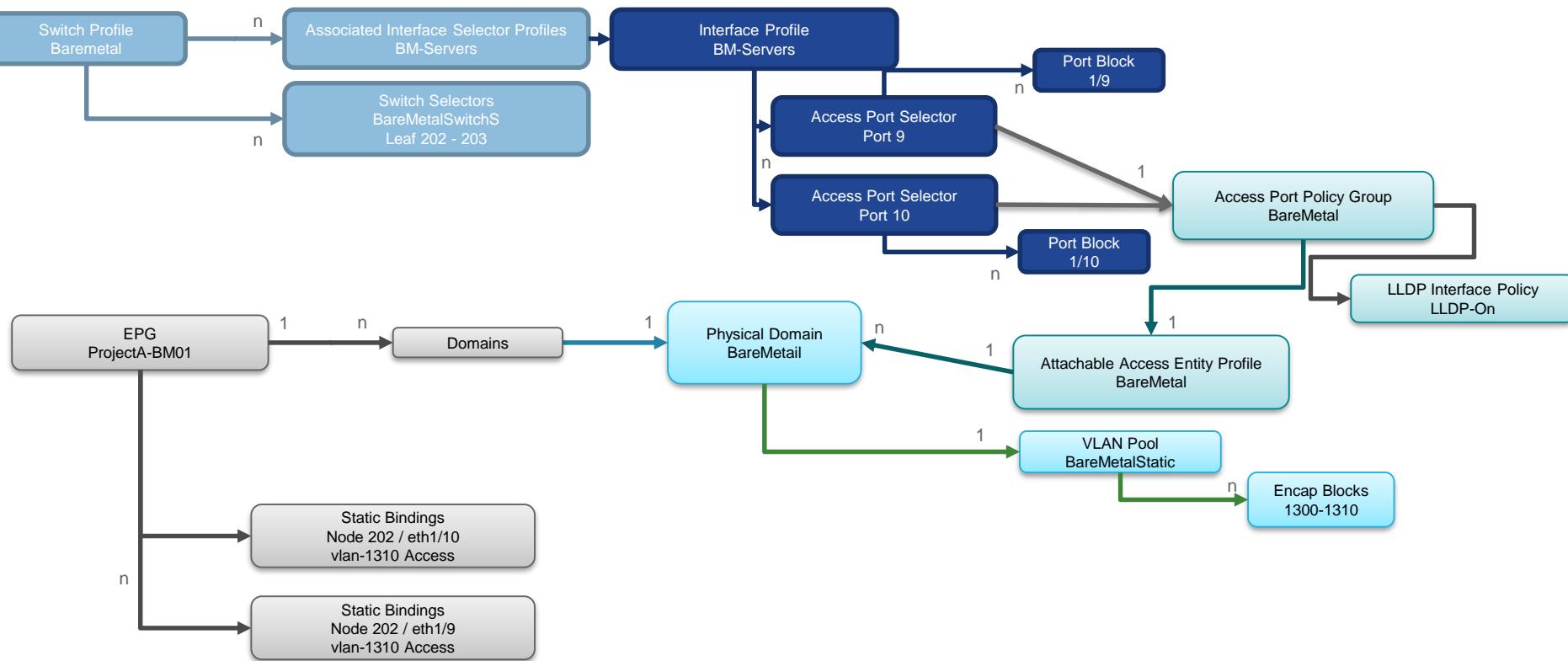
EPG's define where to put the VLAN



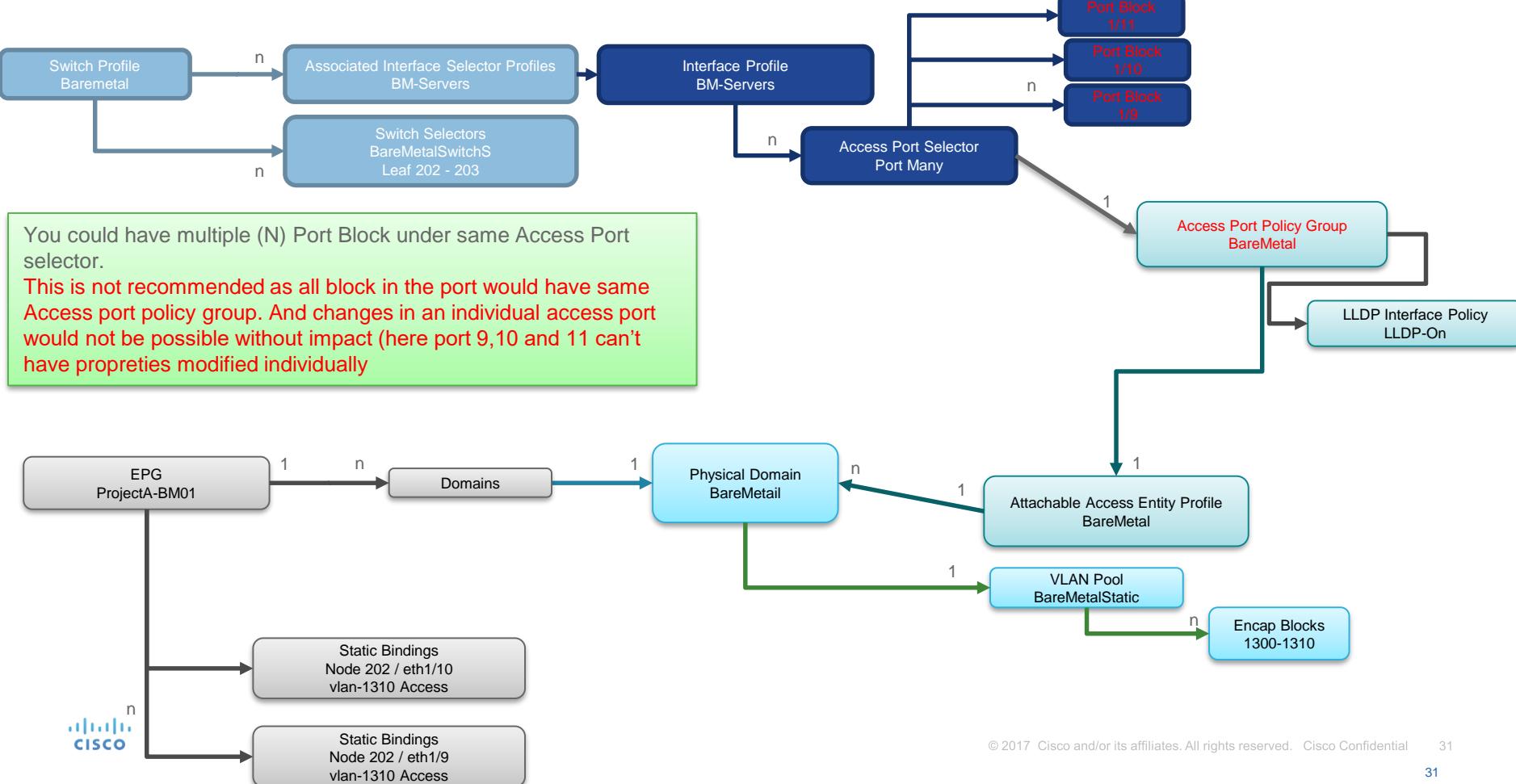
Now we have a VLAN on the port



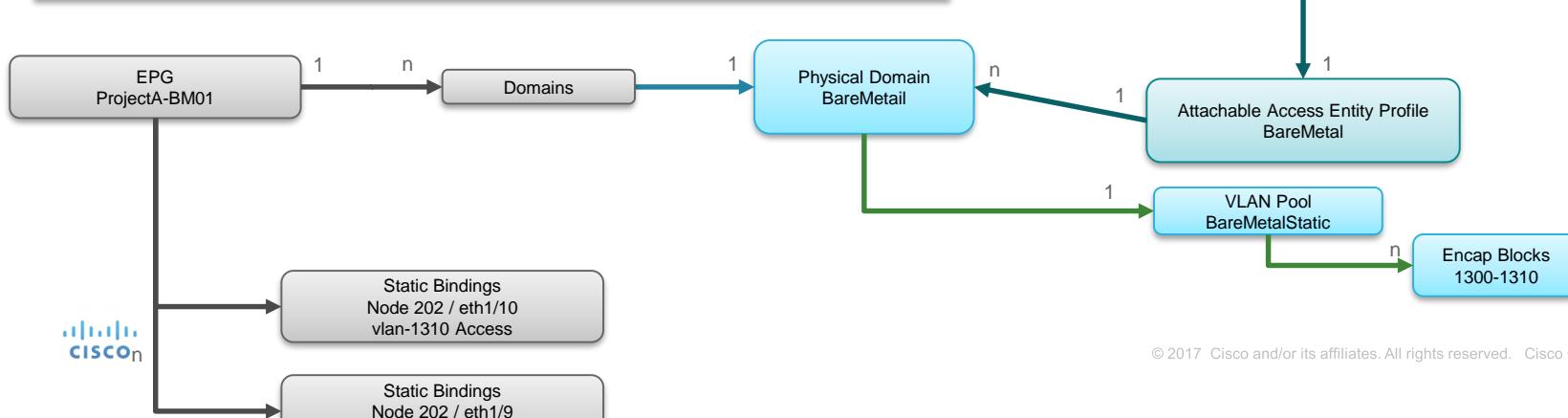
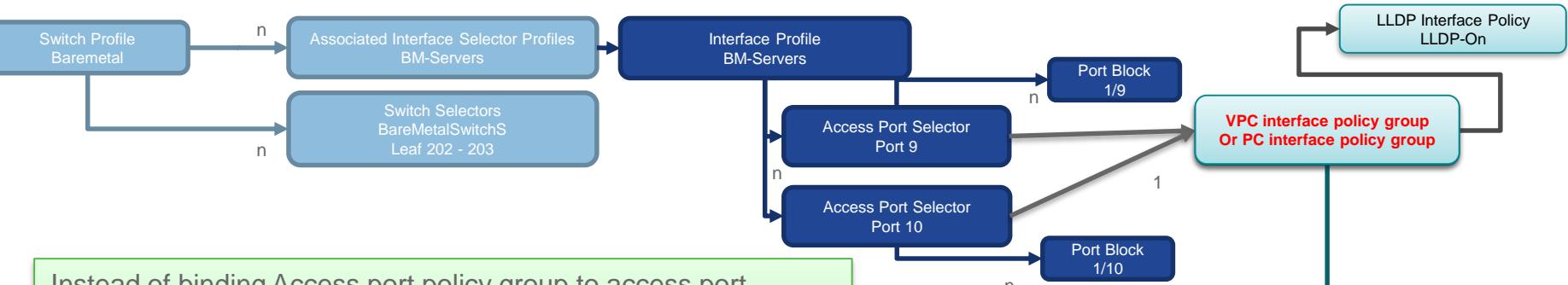
Big Picture 1 – Static path on EPG



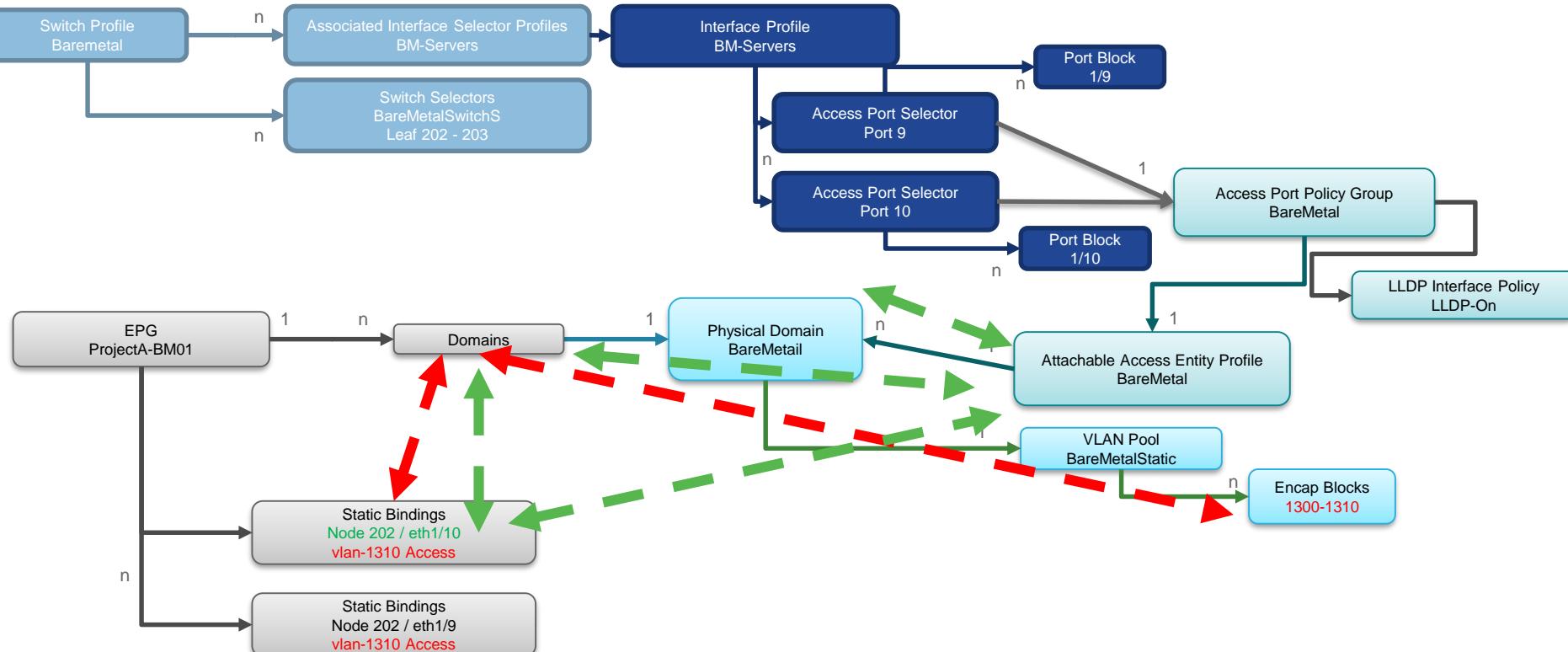
Big Picture 1 – Remark on Access port policy group



Big Picture 1 – PC or VPC interface policy group



Big Picture 1 – Static path on EPG - Validation



Vlan Validation – access vlan part of pool linked to phys Dom of the EPG → if not F0467 invalid VLAN
 Path Validation – node/port has an AEP containing that port
 AND AEP should contain phys domain attached to EPG
 → If not F0467 invalid Path

What happens when we configure a static path

- Static path setup up in EPG1 with DomainA on port LeafX/eth1/Y with vlan Z
 1. Validate that Vlan Z is part of domainA attached to EPG1 → if not Fault F0467 invalid Vlan
 2. Validate that port LeafX/eth1/Y has AEP containing domainA → if not Fault F0467 with invalid path
 3. Validate that domain A is part of AEP attached to the port LeafX/Eth1/Y → if not Fault F0467 invalid path
- Check 2 and 3 will for sure prevent config to be deployed
- Check 1 **may** not prevent correct operation
 - Will break if some specific config in use (e.g. local vlan scope)
 - Will break if flag enforce domain validation is globally set
 - Best Practices : always fix the fault even if not impact 😊

System Settings

- > Quota
- APIC Connectivity Preferences
- BD Enforced Exception List
- Control Plane MTU
- Endpoint Controls
- Fabric Wide Setting
- Port Tracking
- System Global GIPo
- BGP Route Reflector

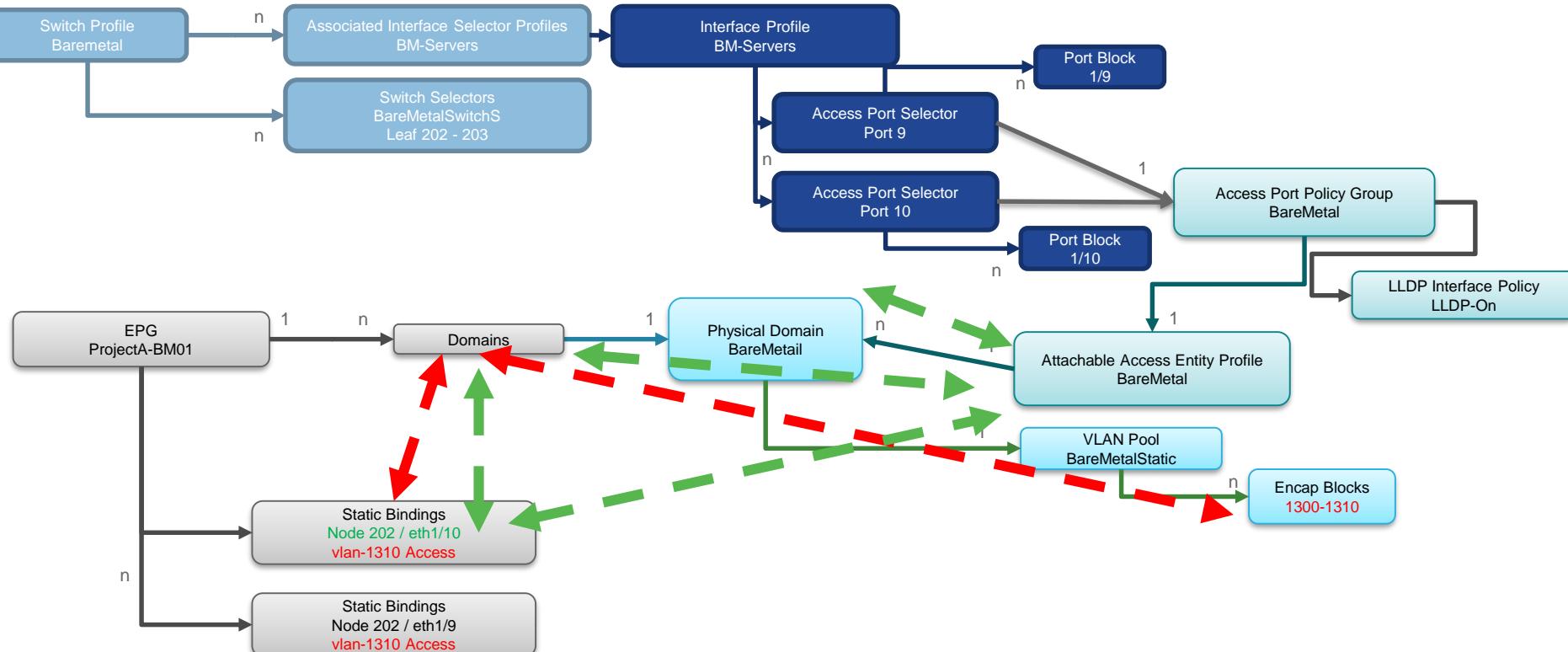


Fabric Wide Setting Policy

Properties

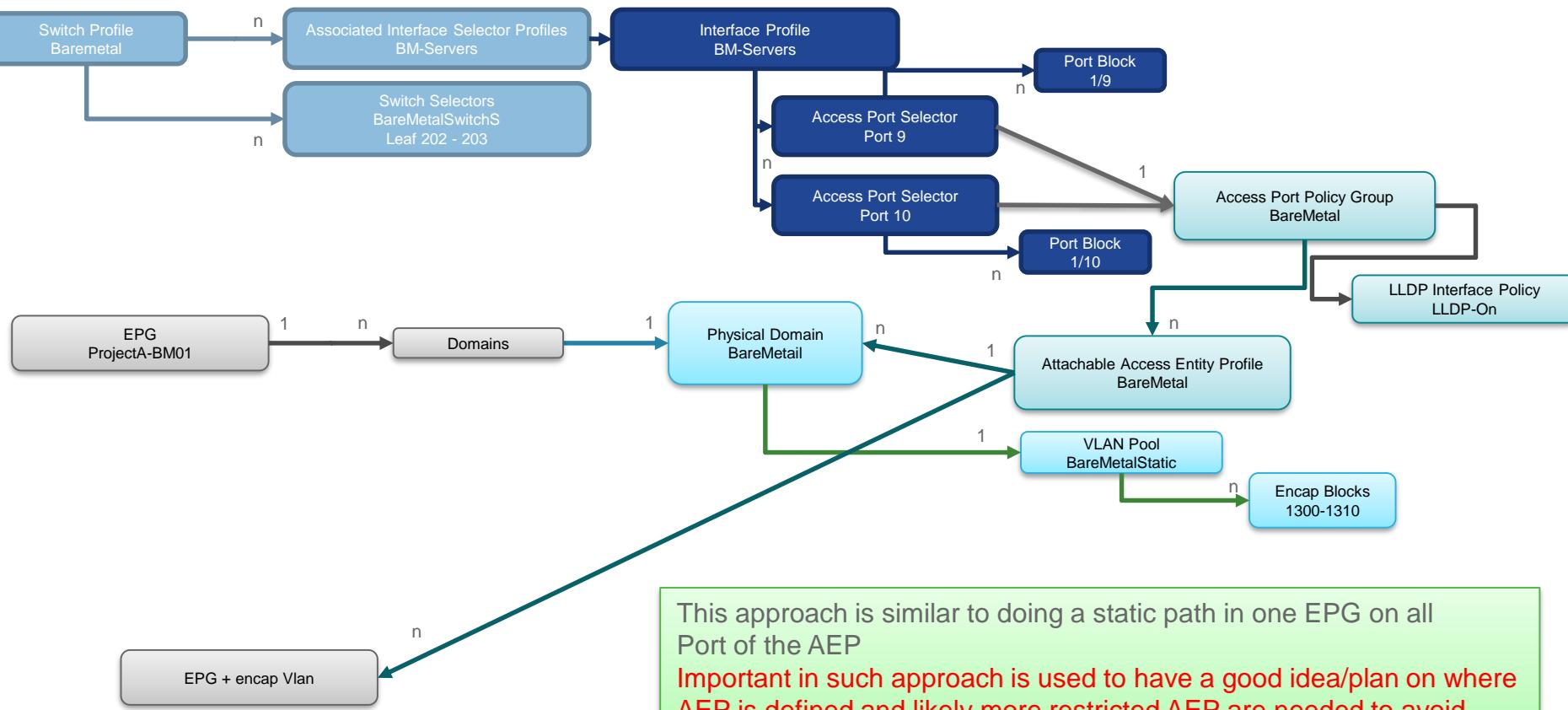
- Disable Remote EP Learning: To disable remote endpoint learning in VRFs containing external bridged/routed domains
- Enforce Subnet Check: To disable IP address learning on the outside of subnets configured in a VRF, for all VRFs
- Reallocate Gipo: Relocate some non-stretched BD gipos to make room for stretched BDs.
- Enforce Domain Validation: Validation check if a static path is added but no domain is associated to an EPG
- Opxlex Client Authentication: To enforce Opxlex client certificate authentication for GOLF and Linux

Big Picture 1 – Static path on EPG - Validation



Vlan Validation – access vlan part of pool linked to phys Dom of the EPG → if not F0467 invalid VLAN
 Path Validation – node/port has an AEP containing that port
 AND AEP should contain phys domain attached to EPG
 → If not F0467 invalid Path

Big Picture 2 – Vlan in AEP



Vlan scaling on leaf
Impact of encapsulation
normalization !

Example on a leaf – step 1

- BD Web - Subnet 10.20.1.254/24
 - EPG1
 - Static path vlan-1050

Need SVI to attach subnet !

Need to know which vxlan encap to use for BD

```
apic1# moquery -d uni/tn-RD-BGP/BD-Web | egrep "dn|seg"  
dn : uni/tn-RD-BGP/
```

Need vlan to map external vlan encap-1050

Need to reflect vlan-1050 is part of BD with Subnet 10.20.1.254/24

```
bdsol-aci32-leaf3# show system internal epm vlan all | egrep "VLAN ID|Type|---|1050| 15 "  
+-----+-----+-----+-----+-----+-----+  
 VLAN ID    Type      Access Encap      Fabric      H/W id   BD VLAN    Endpoint  
          (Type Value)           Encap           Encap  
+-----+-----+-----+-----+-----+-----+-----+-----+  
 15        Tenant BD  NONE          0  15892461    82     15        0  
 62        FD  vlan  802.1Q       1050 8341      83     15        1
```

```
bdsol-aci32-leaf3# vsh -c 'show ip interface vrf RD-BGP:RD' | egrep -B 1 "10.20.1.254"  
Vlan15, Interface status: protocol-up/link-up/admin-up, iod: 124, mode: pervasive, vrf_vnid: 2654211  
IP address: 10.20.1.254, IP subnet: 10.20.1.0/24
```

Example on a leaf – step 2

- BD Web - Subnet 10.20.1.254/24
 - EPG1
 - Static path vlan-1050
 - Dyn path (vmm) vlan-2025

Need SVI to attach subnet !

Need to know which vxlan encap to use for BD

```
apic1# moquery -d uni/tn-RD-BGP/BD-Web | egrep "dn|seg"  
dn : uni/tn-RD-BGP/
```

Need vlan to map external vlan encap-1050 and 2025

Need to reflect vlan-1050 and 2025 is part of BD with Subnet 10.20.1.254/24

bdsol-aci32-leaf3# show system internal epm vlan all egrep "VLAN ID Type --- 1050 15 "						
VLAN ID	Type	Access Encap (Type Value)	Fabric Encap	H/W id	BD VLAN	Endpoint Count
15	Tenant BD	NONE	0 15892461	82	15	0
62	FD	vlan 802.1Q	1050 8341	83	15	0
63	FD	vlan 802.1Q	2025 8417	86	15	0

```
bdsol-aci32-leaf3# vsh -c 'show ip interface vrf RD-BGP:RD' | egrep -B 1 "10.20.1.254"  
Vlan15, Interface status: protocol-up/link-up/admin-up, iod: 124, mode: pervasive, vrf_vnid: 2654211  
IP address: 10.20.1.254, IP subnet: 10.20.1.0/24
```

Example on a leaf – step 3

- BD Web - Subnet 10.20.1.254/24
 - EPG1
 - Static path vlan-1050
 - Dyn path (vmm) vlan-2025
 - EPG2
 - Dyn path (vmm) vlan-2034

Need SVI to attach subnet !

Need to know which vxlan encap to use for BD

```
apic1# moquery -d uni/tn-RD-BGP/BD-Web | egrep "dn|seg"  
dn : uni/tn-RD-BGP/
```

Need vlan to map external vlan encap-1050 and 2025

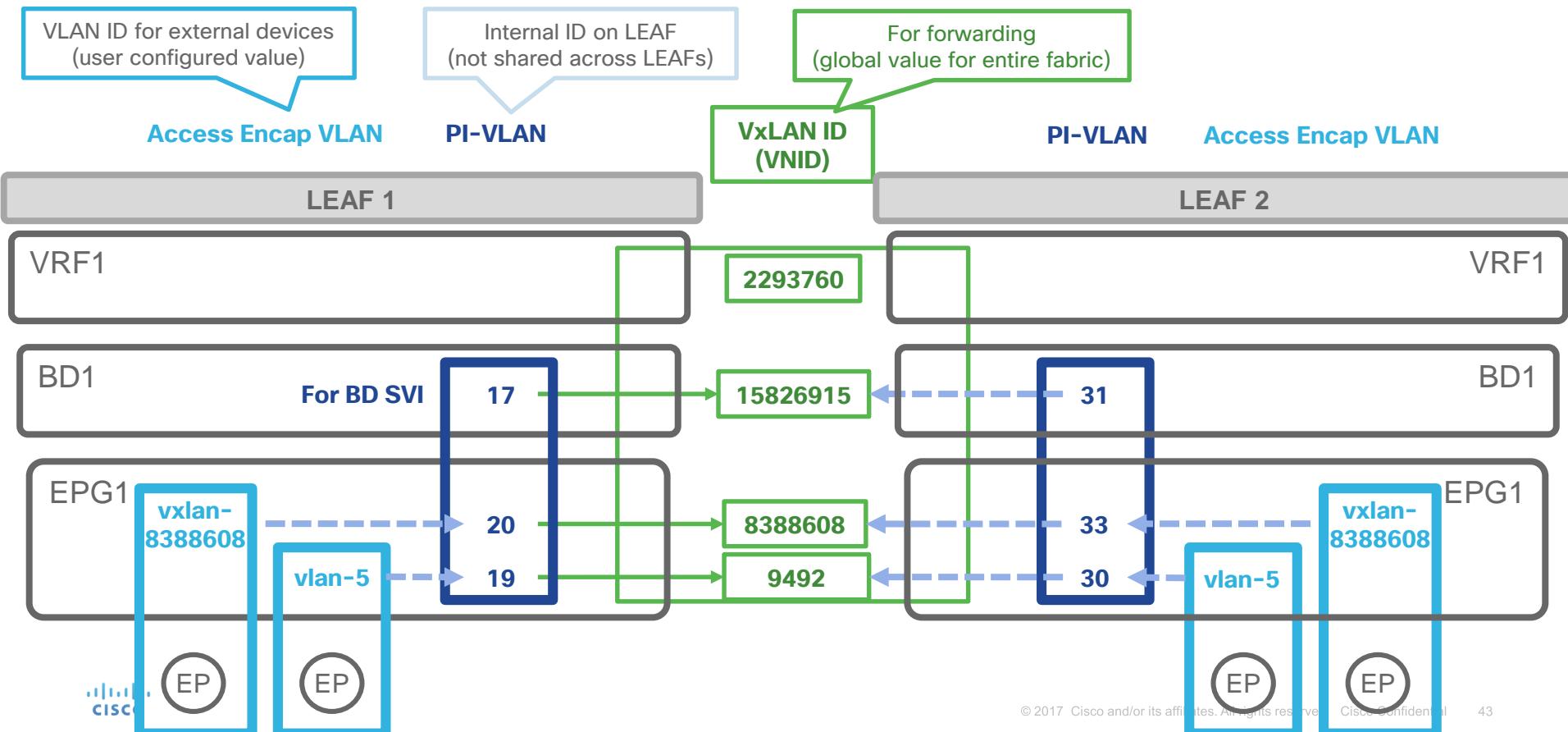
Need to reflect vlan-1050 and 2025 and 2034 is part of BD with Subnet 10.20.1.254/24

bdsol-aci32-leaf3# show system internal epm vlan all egrep "VLAN ID Type --- 1050 15 "						
VLAN ID	Type	Access Encap (Type Value)	Fabric Encap	H/W id	BD VLAN	Endpoint Count
15	Tenant BD	NONE	0 15892461	82	15	0
62	FD	vlan 802.1Q	1050 8341	83	15	0
63	FD	vlan 802.1Q	2025 8417	86	15	0
75	FD	vlan 802.1Q	2034 8426	87	15	0

```
bdsol-aci32-leaf3# vsh -c 'show ip interface vrf RD-BGP:RD' | egrep -B 1 "10.20.1.254"  
Vlan15, Interface status: protocol-up/link-up/admin-up, iod: 124, mode: pervasive, vrf_vnid: 2654211  
IP address: 10.20.1.254, IP subnet: 10.20.1.0/24
```

VLAN types in ACI

* PI-VLAN : Platform Independent VLAN





PI-VLAN for EPG and BD CLI

- Endpoint Table

leaf1# show endpoint ip 192.168.0.51				
19	vlan-5	0000.5555.1111	L	eth1/1
TK:VRF1	vlan-5	192.168.0.51	L	eth1/1

PI-VLAN

Access Encap VLAN

- VLAN Table

NOT Access Encap VLAN.
PI-VLAN 17, 19

"extended" option to display Access Encap VLAN

leaf1# show vlan id 17,19 extended				
VLAN	Name	Type	Vlan-mode	Encap
17	TK:BD1	enet	CE	vxlan-15826915
19	TK:AP1:EPG1	enet	CE	vlan-5
VLAN	Name	Type	Vlan-mode	Encap
17	TK:BD1	enet	CE	vxlan-15826915
19	TK:AP1:EPG1	enet	CE	vlan-5

PI-VLAN

Access Encap VLAN



PI-VLAN for EPG and BD CLI

```
leaf1# show vlan id 17,19 extended
```

VLAN	Name	Status	Ports
17	TK:BD1	active	Eth1/1, Eth1/2, Po6
19	TK:AP1:EPG1	active	Eth1/1

VLAN	Type	Vlan-mode	Encap
17	enet	CE	vxlan-15826915
19	enet	CE	vlan-5

EPG
PI-VLAN

```
leaf1# show system internal epm vlan 19
```

VLAN ID	Type	Access Encap (Type Value)	Fabric Encap	H/W id	BD VLAN	Endpoint Count
19	FD vlan	802.1Q	5	8294	14	17

CISCO

Example summary

- Here one BD with 2 EPG and a total of 3 external encap consume 4 internal vlan
 - One BD vlan – we attach IP and subnet there - Aci forwards based on BD not EPG
 - 3 encap vlan (called FD_VLAN) – one for each external encap and all bound to the BD VLAN
- Whether ext encap are part of same same EPG or not DOES not matter for forwarding perspective (only BD matters)
- EPG comes into play only when we check contract !

Tshooting Command 1

- VLAN brief in ELTMC

```
Vsh_lc →
module-1# show system internal eltmc info vlan brief
VLAN-Info
VlanId HW_VlanId Type Access_enc Access_enc Fabric_enc Fabric_enc BDVlan
Type Type
=====
 13      1   BD_CTRL_VLAN 802.1q    4093    VXLAN  16777209    0
 17      2   BD_VLAN     Unknown    0        VXLAN  16023498    17
 18     12   FD_VLAN     802.1q    2080    VXLAN  8372     17
 19     13   FD_VLAN     802.1q    2091    VXLAN  8383     17
 20      3   BD_VLAN     Unknown    0        VXLAN  16514958    20
 22      4   BD_VLAN     Unknown    0        VXLAN  16613251    22
 23      5   BD_VLAN     Unknown    0        VXLAN  16220082    23
 24     14   FD_VLAN     802.1q    3300    VXLAN  8492     23
 25     15   FD_VLAN     802.1q    3301    VXLAN  8493     23
 27      6   BD_VLAN     802.1q    3397    VXLAN  16711545    27
 28      9   BD_EXT_VLAN 802.1q    2021    VXLAN  14778357    28
 29     10   BD_EXT_VLAN 802.1q    2020    VXLAN  14712828    29
 30     16   FD_VLAN     802.1q    3312    VXLAN  8504     23
 32     11   BD_EXT_VLAN 802.1q    2200    VXLAN  15269817    32
 34      8   BD_VLAN     Unknown    0        VXLAN  15138761    34
 35     21   FD_VLAN     802.1q    3399    VXLAN  8591     34
 36      7   FD_VLAN     802.1q    3398    VXLAN  8590     34
 37     17   BD_VLAN     Unknown    0        VXLAN  15957972    37
 38     37   FD_VLAN     802.1q    2083    VXLAN  8375     37
 39     38   BD_VLAN     802.1q    2085    VXLAN  16744308    39
 44     18   FD_VLAN     802.1q    2174    VXLAN  8665     20
 45     19   FD_VLAN     802.1q    2145    VXLAN  8636     20
 46     20   FD_VLAN     802.1q    2101    VXLAN  8592     20
```

Here :

Type :

BD_VLAN is vlan for a BD

FD_VLAN is vlan for an epg with vlan encaps

FD_VXLAN is vlan used for epg using VXLAN as access encaps

BD_EXT_VLAN is for ext network

BD_CTRL_VLAN is the infra vlan 4093 (default)

..

VlanId is the internal vlan (same as seen in show vlan)

Hw_VlanId is the ASIC VLAN (likely not relevant in most case)

Access_enc is the encapsulation VLAN or **VXLAN**

Fabric_enc is the VXLAN (VNID) used inside the fabric

BDVlan is the parent vlan for EPG

What if we do not know
what to do ?
To Flood or not to flood !

To flood or not to flood ?

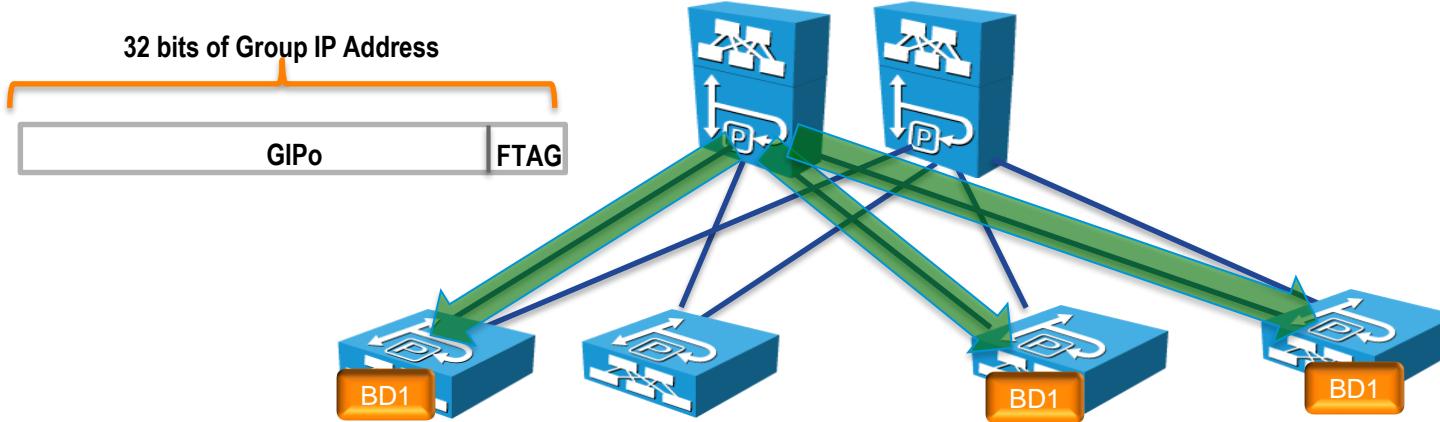
- We flood (legacy networking behavior) :
 - Layer 2 (same BD) :
 - for Broadcast, unknown multicast always
 - For unknown unicast if the BD is in flood mode
 - For known multicast : restricted flood to IGMP receiver
 - Rely on Multicast in the fabric (outer IP multicast)
- We do not flood (proxy behavior)
 - For Layer 3 : always
 - For Layer 2 : only for unkwnon unicast when BD is in hw-proxy mode
- Proxy is on spine and involved the COOP protocol

Flooding in ACI fabric

Multicast in Fabric - Terminology

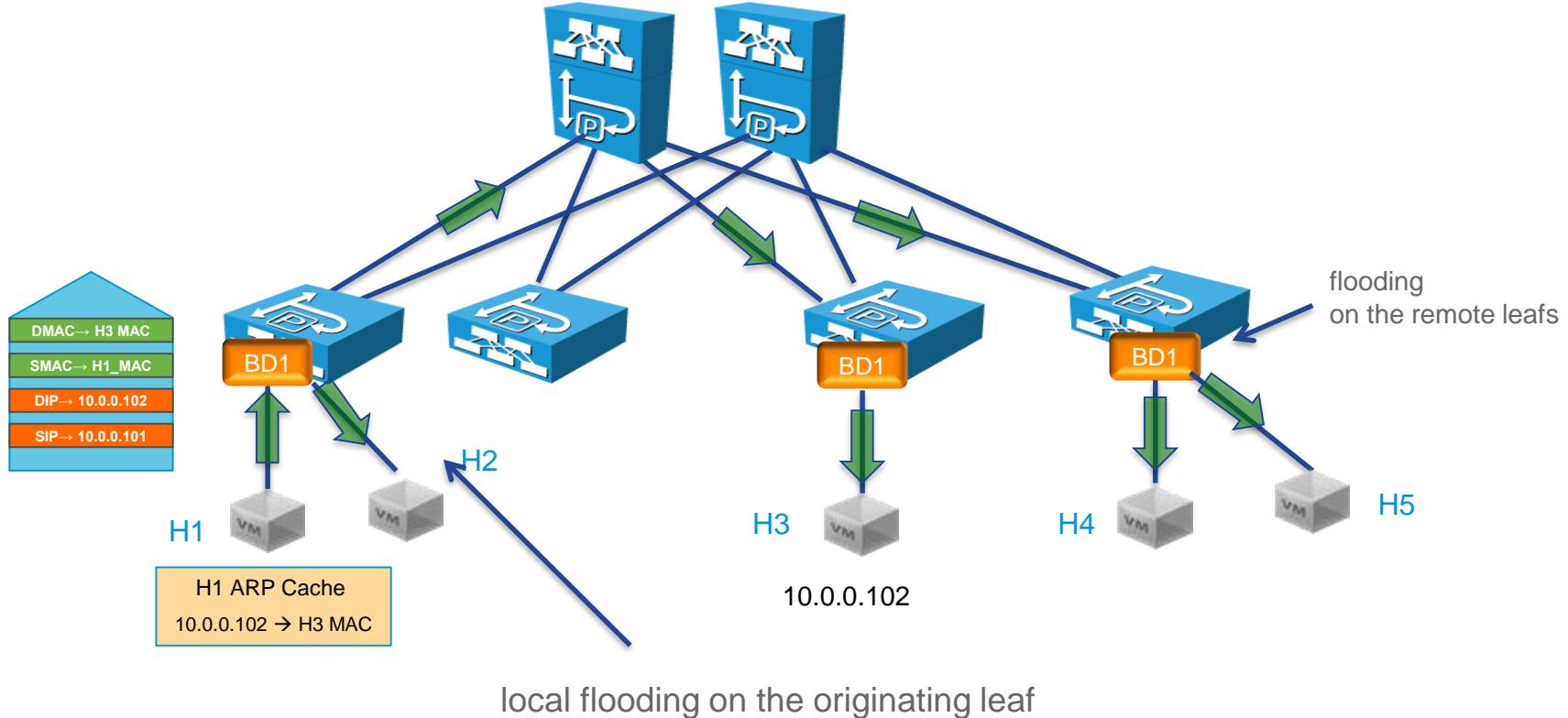
- Type of Multicast Group
 - GIPo – Group IP outer – per BD used between leaf and spine vxlan packets
 - GIPi – Group IP inner – Tenant space, inner packets
 - GIPo' – Group IP outer prime – Used between Phys Leaf and Vleaf running vxlan only
- Fabric flooding – multicast flow for a BD GIPo

Bridge Domain Multi-Destination Tree – High level



- Each Bridge Domain is associated with a VXLAN Group IP Address (GIPo) or outer Multicast (Group) address
- APIC configuration defines which Leafs have which BD-VNIDs enabled on them
- The APIC configuration also specifies the BD-VNID to GIPo mapping (part of fv.BD MO)
- Learning fabric wide GIPo membership information is learnt via IS-IS Group Membership LSP's (GM-LSPs)
- Traffic from a Leaf is forwarded to the destination GIPo adding the specific FTAG bits matching the FTAG hash selection
- Dataplane floods traffic in vxlan in BD VNID with dest IP based on BD GIPo

Forwarding with Flooding for unknown unicast



Finding BD GiPo and VNID

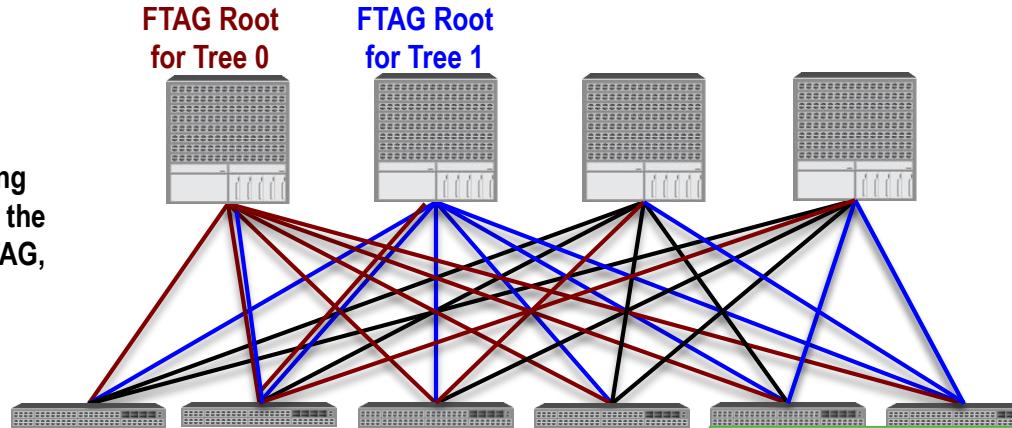
- Type of Multicast Group
 - **GiPo** – Group IP outer – per BD used between leaf and spine vxlan packets
 - Flooded packet encapsulated in BD VNID (vxlan) with destination IP being Multicast GiPo
 - One per BD assinged by APIC at BD creation

```
apic1# moquery -c fvBD | egrep "#|bcast|dn|seg"  
# fv.BD  
bcastP : 225.1.184.176  
dn : uni/tn-RD-OSPF/BD-BD1  
seg : 16220088  
# fv.BD  
bcastP : 225.0.176.192  
dn : uni/tn-RD-OSPF/BD-BD2  
seg : 16089027
```

BD1 in tenant RD-OSPF will be flooded with vxlan 16220088 in GiPo based 225.1.184.176 (+ftag)

Multicast Forwarding – FTAG Topologies

FTAG topologies are advertised via IS-IS using martian addresses with the final 4 bits set to the FTAG, 0.0.0.<ftag-id>

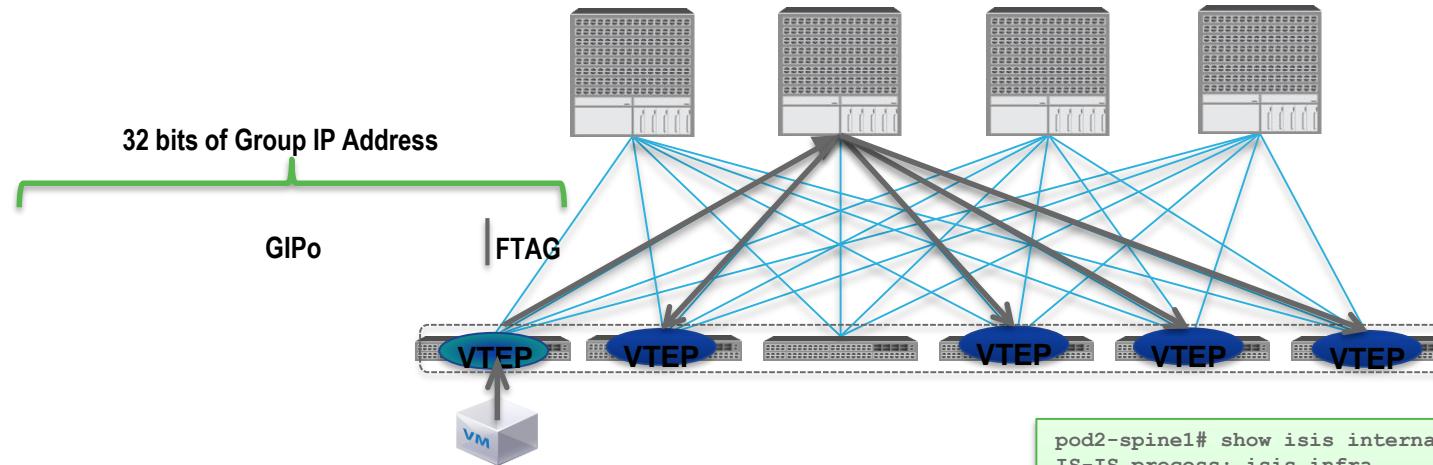


FTAG topologies are limited to spine to leaf links and only if a downlink is lost will the FTAG topology pass through a leaf

- To improve load balancing **Multicast traffic is distributed across 16 FTAG topologies in the Fabric**
- Inner IP payload are hashed across the FTAG topologies
- FTAG trees are rooted at spine switches (roots determined by APIC)
- FTAG tree calculation is performed by IS-IS and will create the FTAG trees as maximally redundant graphs
- As FTAG should give a non looped path reaching every leaf and spine**
- FTAG nodes are advertised using GM-LSP with martian addresses 0.0.0.<ftag-id> representing the FTAG (the last 4 bits indicate the FTAG)

```
pod2-spine1# show isis internal mcast route ftag
IS-IS process: isis_infra
  VRF : default
  FTAG Routes
  -----
  FTAG ID:  0 [Disabled]
  FTAG ID:  1 [Enabled] Cost:( 2/   8/   0)
  -----
    Root port: Ethernet1/34.34
    OIF List:
  FTAG ID:  2 [Root] [Enabled] Cost:( 0/   0/   0)
  -----
    Root port: -
    OIF List:
      Ethernet1/33.33
      Ethernet1/34.34
      Ethernet1/35.35
      Ethernet1/36.36..
```

Multicast Forwarding - GIPo



- Each Bridge Domain is associated with a VXLAN Group IP Address (GIPo) or outer Multicast (Group) address
- APIC configuration defines which Leafs have which BD-VNIDS enabled on them
- The APIC configuration also specifies the BD-VNID to GIPo mapping
- Learning fabric wide GIPo membership information is learnt via IS-IS Group Membership LSP's (GM-LSPs)
- No need to flood to a leaf where the BD do not exist

CISCO

```
pod2-spine1# show isis internal mcast route gipo
IS-IS process: isis_infra
VRF : default
```

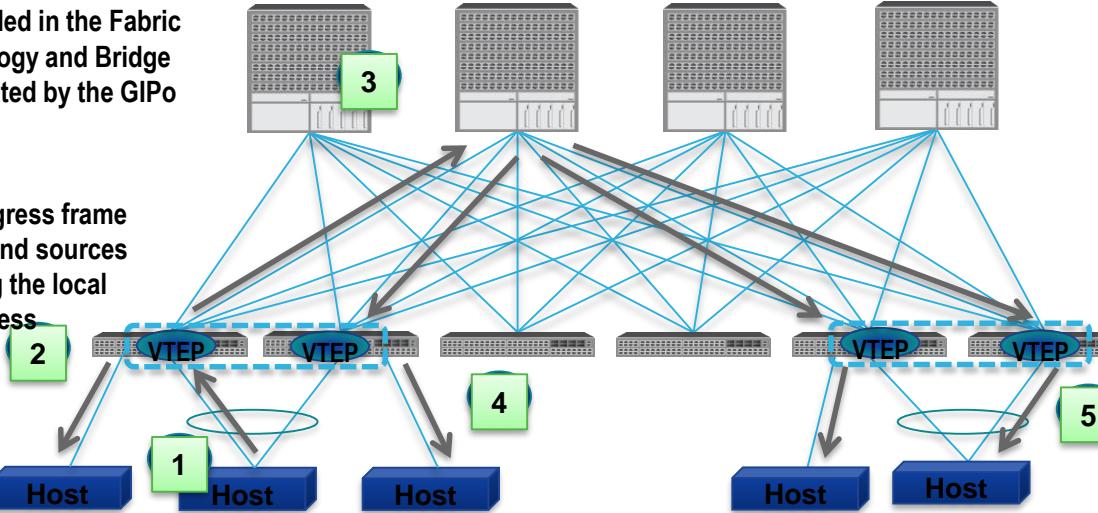
GIPo Routes

```
=====
GIPo: 225.0.0.16 [TRANSIT]
OIF List:
Ethernet1/33.33
Ethernet1/34.34
Ethernet1/35.35
Ethernet1/36.36
```

```
GIPo: 225.0.3.160 [TRANSIT]
OIF List:
Ethernet1/34.34
Ethernet1/35.35
```

vPC & Multicast

Multicast traffic is forwarded in the Fabric according to FTAG topology and Bridge Domain interest as indicated by the GIPo address



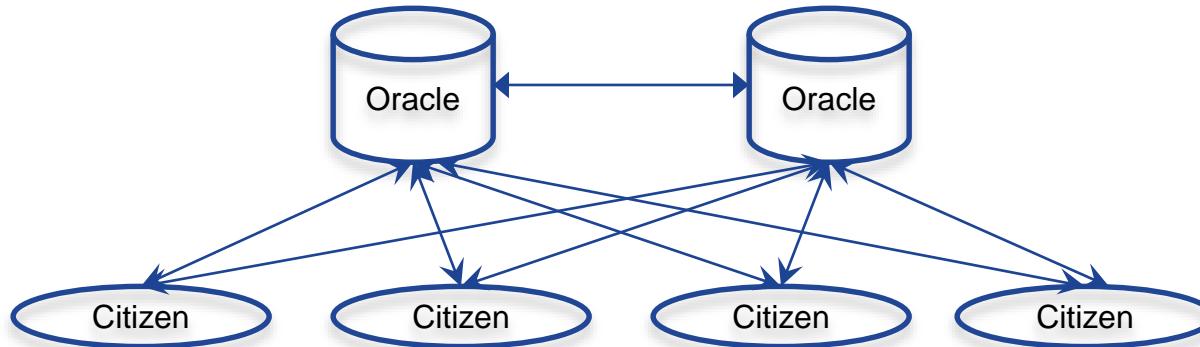
- Multicast traffic leverages the same source vPC VTEP as unicast traffic
- Multicast behaviour has to support
 - One copy to VPC hosts (avoid fabric duplication of frames) – one switch per pair is DF (des forwarder) for the BD

Traffic received for multicast forwarding not from 'my' vPC VTEP address is:

- a) Forwarded to all non VPC (orphan) ports in the OIFL
- b) Forwarded to vPC ports in the OIFL based on hash selection (symmetrical hash determines which switch forwards each frame, a single copy is sent to the attached host)

COOP – Council of oracle Protocol Spine-Proxy

COOP Software Architecture – High Level



- COOP is a distributed runtime repository for the fabric
- All the records (ex : end point, mcast group membership,...) are maintained on the Oracles
- Oracles form a council
- Citizens publish records into the council of oracles
- Oracles sync the records between themselves in the council
- Citizens receive any notifications for published records
- Only new learned or aged local EP are send by citizen to oracle
- Citizen == Leaf and Spine == Oracle

COOP (= End Point Learning on Spine)

How Spines collect all EPs data from each Leaf ?

1. Leaf learns EP (either MAC or/and IP) as **local**
2. Leaf reports it to Spine via COOP process
3. Spine stores these in COOP DB

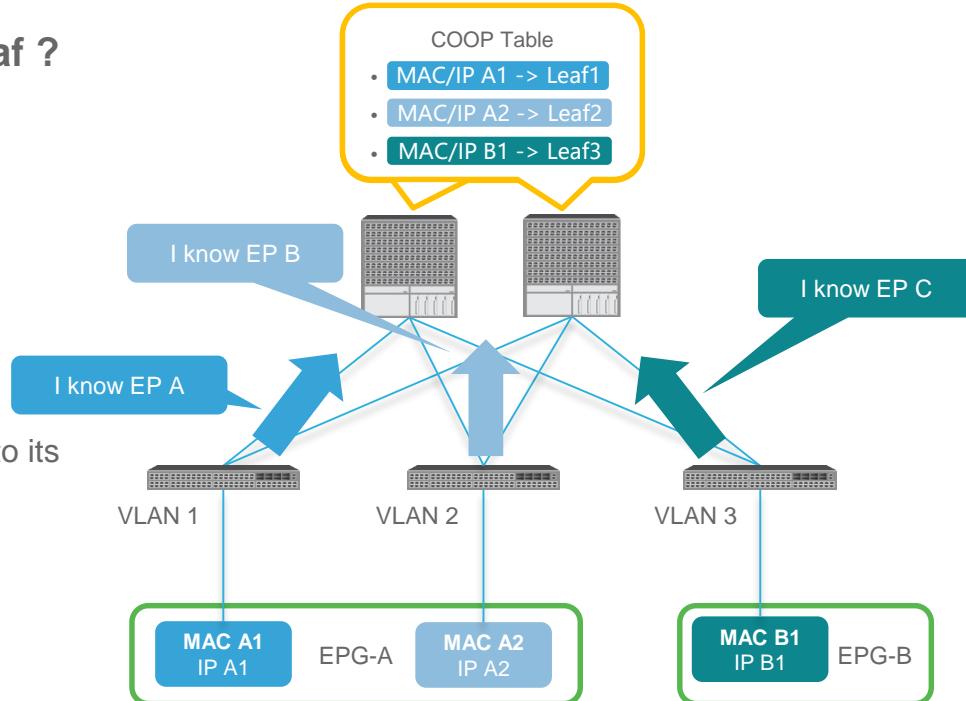
What is the purpose of COOP?

When Leaf doesn't know dst EP, Leaf forwards packet to Spine in order to let Spine handle according to its COOP DB.

This behavior is called **Spine-Proxy**.

Note :

- Normally Spine doesn't propagate COOP DB to each Leaf. It just receives and stores.
- Remote EndPoints are stored on each Leaf nodes as cache. This is not reported to Spine COOP.





How to check COOP DB on Spine

```
fab5-spine2# show coop internal info repo ep key 15826915 0000.5555.1111 | egrep 'vnid|mac|id|Real'
```

EP bd vnid : 15826915
EP mac : 00:00:55:55:11:11
Vrf vnid : 2293760

EP data

BD VNID

Epg vnid : 0
Ep vpc-id : 0
Ep vpc virtual switch-id : 0.0.0.
publisher id : 11.0.200.92
Real IPv4 EP : 192.168.5.111

TEP address of LEAF

EP data

```
fab5-spine2# show coop internal info ip-db key 2293760 192.168.5.111
```

IP address : 192.168.5.111
Vrf : 2293760
Flags : 0
EP bd vnid : 15826915
EP mac : 00:00:55:55:11:11
Publisher Id : 11.0.200.92
---- snip ----

TEP address of LEAF

EP data

VRF VNID

Example Coop entry

```
bdsol-aci32-spine1# show coop internal info repo ep |  
egrep -20 "172.16.10.1"
```

```
Current published TEP : 10.0.88.90  
Backup Path:  
BackupTunnel nh : 0.0.0.0  
Current Backup (publisher_id): 0.0.0.0  
Anycast_flags : 0  
Current citizen (publisher_id): 10.0.88.90  
Previous citizen : 10.0.88.90  
Prev to Previous citizen : 10.0.88.90  
. .  
IPv4 Repo Hdr last pub timestamp : 11 12 2018  
10:31:10 473620552  
IPv4 Repo Hdr last dampen timestamp : 01 01 1970  
00:00:00 0  
IPv4 Repo Hdr dampen penalty : 0  
IPv4 Repo Hdr flags : IN_OBJ UPD_OBJ EXPORT  
Real IPv4 EP : 172.16.10.1  
Synthetic Flags IPv4 EP : 0x25  
EVPN Seq no : 0  
Remote publish timestamp: 01 01 1970 00:00:00 0  
Current publisher_id: 10.0.88.90  
BackupTunnel nh : 0.0.0.0  
MAC Tunnel : 10.0.88.90  
IPv4 Tunnel : 10.0.88.90  
IPv6 Tunnel : 10.0.88.90
```

```
bdsol-aci32-spine1# show coop internal info ip-  
db key 2654211 172.16.10.1
```

```
IP address : 172.16.10.1  
Vrf : 2654211  
Flags : 0  
EP bd vnid : 15106002  
EP mac : 00:50:56:A4:00:17  
Publisher Id : 10.0.88.90  
Record timestamp : 11 12 2018 10:31:10 473465969  
Publish timestamp : 11 12 2018 10:31:10  
473620552  
Seq No: 0  
Remote publish timestamp: 01 01 1970 00:00:00 0  
URIB Tunnel Info  
Num tunnels : 1  
Tunnel address : 10.0.88.90  
Tunnel ref count : 1
```

Coop log

Citizen

```
pod2-leaf1# show coop internal trace-detail-uc | egrep "10.11.1.102"
1) 2016 Oct  4 11:48:35.056710 TID 01:coop_citizen_publish_ep:955: ADD EP msg <16023499,
00:2A:6A:B1:91:7C> #IPs: 2 <3047424, 10.11.1.102> with trans_id: 243388 rec_ts: 1475581398:770859790
pub_ts: 1475581398:771923263
10) 2016 Oct  4 11:34:42.970314 TID 01:coop_citizen_publish_ep:955: ADD EP msg <16023499,
00:2A:6A:B1:91:7C> #IPs: 2 <3047424, 10.11.1.102> with trans_id: 243385 rec_ts: 1475580566:684896885
pub_ts: 1475580566:685521638
```

oracle

```
pod2-spine1# show coop internal trace-detail-uc | egrep 10.11.1.102
2) 2016 Oct  4 11:53:27.990320 TID 11:coop_program_synthip:1493: log_collect_coop optype=ADD;eptype=ip;
ip=10.11.1.102; nh=10.0.168.95;flags=0x65;<bd_vnid=16023499;mac=00:2A:6A:B1:91:7C>;
22) 2016 Oct  4 11:48:35.058296 TID 16:coop_program_synthip:1493: log_collect_coop optype=ADD;eptype=ip;
ip=10.11.1.102; nh=10.0.168.95;flags=0x65;<bd_vnid=16023499;mac=00:2A:6A:B1:91:7C>;
```

COOP log in linux file system (as of 3.1)

Now coop log are in filesystem as well

```
bdsol-aci32-spine1# ls -al coop_trace.txt  
-rw-rw-rw- 1 root root 53325824 Sep 26 11:02 coop_trace.txt  
bdsol-aci32-spine1#
```

```
bdsol-aci32-spine1# egrep "eptype=ip.*10.100.0.2" coop_trace.txt  
[2018 Sep 26 02:45:15.945429479:10707311:U:coop_program_synthip:2081] TID  
25:[DBG_COOP_TRACE_DETAIL_UC_LOG_TRACE]:log_collect_coop optype=ADD;eptype=ip; ip=10.100.0.2;  
nh=10.0.80.94;flags=0x5;<vrif_vnid=3112960;ip=10.100.0.2>;  
[2018 Sep 26 03:45:16.002153863:10737862:U:coop_program_synthip:2081] TID  
13:[DBG_COOP_TRACE_DETAIL_UC_LOG_TRACE]:log_collect_coop optype=ADD;eptype=ip; ip=10.100.0.2;  
nh=10.0.80.94;flags=0x5;<vrif_vnid=3112960;ip=10.100.0.2>;  
[2018 Sep 26 04:45:16.057634435:10768189:U:coop_program_synthip:2081] TID  
31:[DBG_COOP_TRACE_DETAIL_UC_LOG_TRACE]:log_collect_coop optype=ADD;eptype=ip; ip=10.100.0.2;  
nh=10.0.80.94;flags=0x5;<vrif_vnid=3112960;ip=10.100.0.2>;
```

COOP summary for end point

- Just a sharded DB on spine
- Leaf responsible of informing COOP of any New locally learned end point
- Leaf responsible of clearing in COOP any Aged/cleared locally learned end point
- Leaf Takes no action for remote end point learned or cleared
- COOP takes care of Bounced EP (see later in epm part)

Any active endpoint anywhere in the fabric should be in COOP DB

COOP what else does it do ?

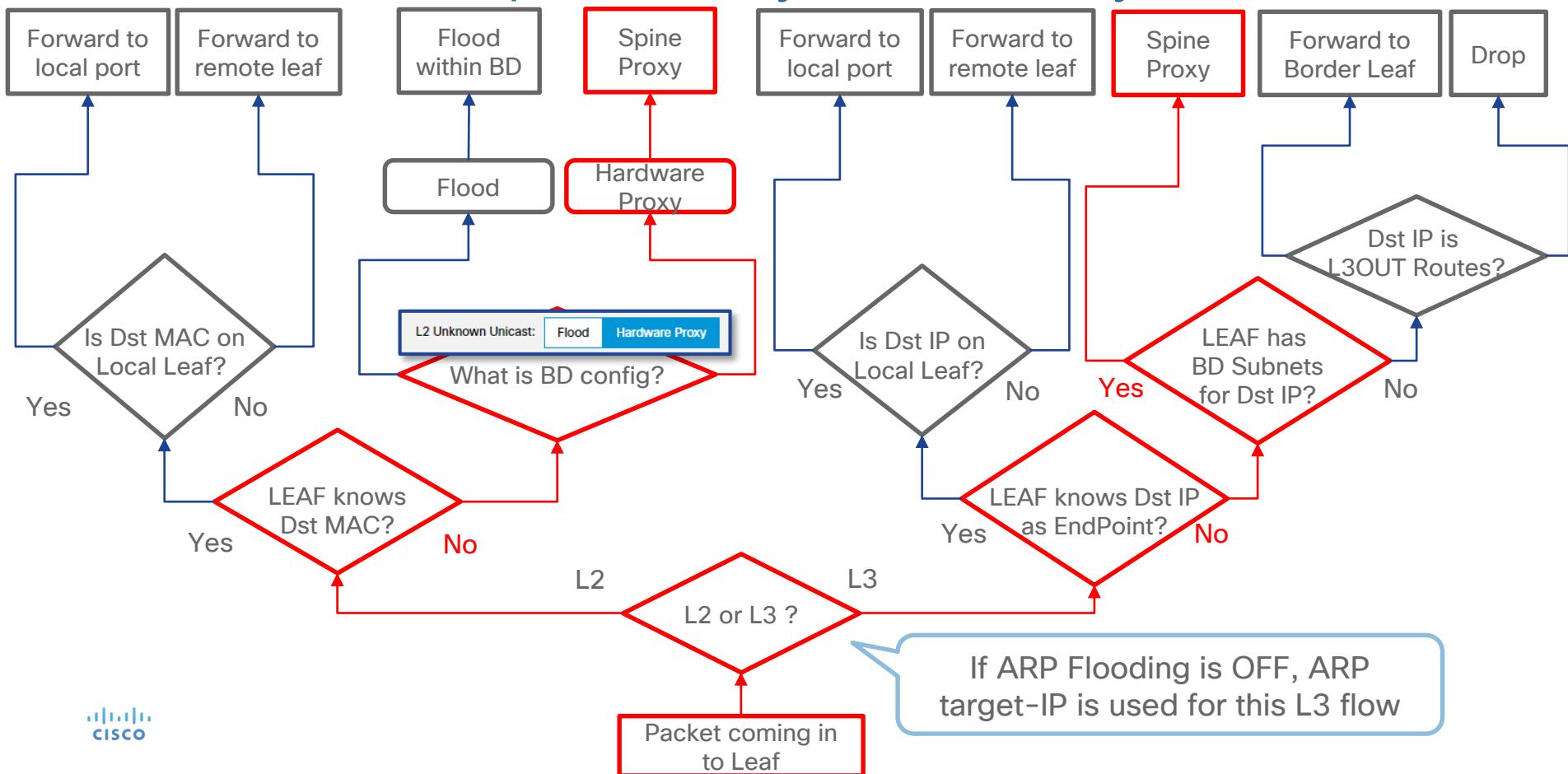
- Maintain vpc member DB (to redirect traffic to anycast or PTEP depending of active VPC member)
- COOP maintains IGMP group interest across the fabric
- ...

The Spine Proxy Function

Proxy what is it ?

- When we do not know we asks spine (Proxy) !
- That means encaps the original packet and send it to the spine.
 - Outer Target IP will be one of the 3 spine loopback depending we want a L2 or L3 lookup in COOP DB

When do we Spine Proxy ? Summary





How to check Spine-Proxy TEP

```
leaf1# show ip route vrf TK:VRF1
```

```
192.168.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive
  *via 10.0.16.64%overlay-1, [1/0], 00:21:39, static
```

BD Subnet (Pervasive Route)

next-hop should be
SPINE-PROXY

```
leaf1# show isis dsteps vrf overlay-1 | grep PROXY
```

10.0.16.65	SPINE	N/A	PHYSICAL, PROXY-ACAST-MAC
10.0.16.64	SPINE	N/A	PHYSICAL, PROXY-ACAST-V4
10.0.16.67	SPINE	N/A	PHYSICAL, PROXY-ACAST-V6

next-hop of Pervasive Route
is IPv4 Spine Proxy TEP

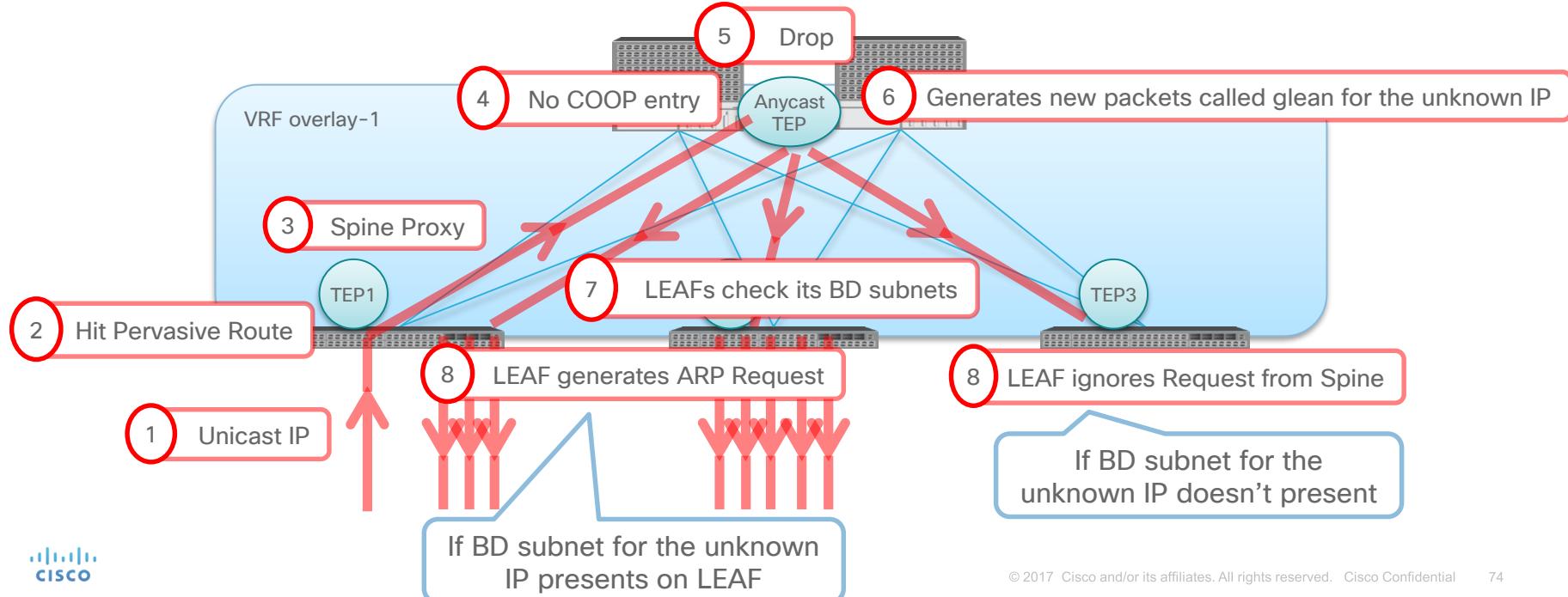
Three types of Spine Proxy TEP

- Proxy-Acast-MAC
 - ✓ Spine-Proxy for L2 traffic (L2 Unknown Unicast mode “Hardware Proxy”)
- Proxy-Acast-V4
 - ✓ Spine-Proxy for IPv4 traffic (includes ARP Request with ARP Flooding mode “OFF”)
- Proxy-Acast-V6
 - ✓ Spine-Proxy for IPv6 traffic

ARP Glean (Silent Host Tracking)

What if even SPINE COOP doesn't know the destination when proxy'ed?

- ✓ L2 Traffic : Drop
- ✓ L3 Traffic : ARP Glean



BD settings

Pervasive Gateway(BD SVI)

The screenshot shows the Cisco Application Centric Infrastructure (ACI) User Interface. On the left, the navigation pane for Tenant TK includes sections for Quick Start, Tenant TK (with Application Profiles and Networking), Bridge Domains (with BD1 selected), Subnets (highlighted with a red box), ND Proxy Subnets, and BD2, BD3, BD_SG_PBR1. The main panel displays 'Subnet - 192.168.0.254/24' properties. The 'Properties' section includes fields for IP Address (192.168.0.254/24), Description (optional), Treat as virtual IP address (unchecked), Make this IP address primary (unchecked), Scope (Private to VRF checked, Advertised Externally and Shared between VRFs unchecked), Subnet Control (checkbox checked, radio button unchecked), and L3 Out for Route Profile (Select a value dropdown). Below these are 'Route Profile' and 'Route Profile' dropdowns.

```
leaf1# show ip route vrf TK:VRF1
```

192.168.0.0/24, ubest/mbest: 1/0, attached, direct, **pervasive**
*via 10.0.184.64%overlay-1, [1/0], 04:32:16, static

192.168.0.254/32, ubest/mbest: 1/0, attached
*via 192.168.0.254, vlan10, [1/0], 04:32:16, local, local

BD SVI with PI-VLAN

Pervasive route

Pervasive SVI

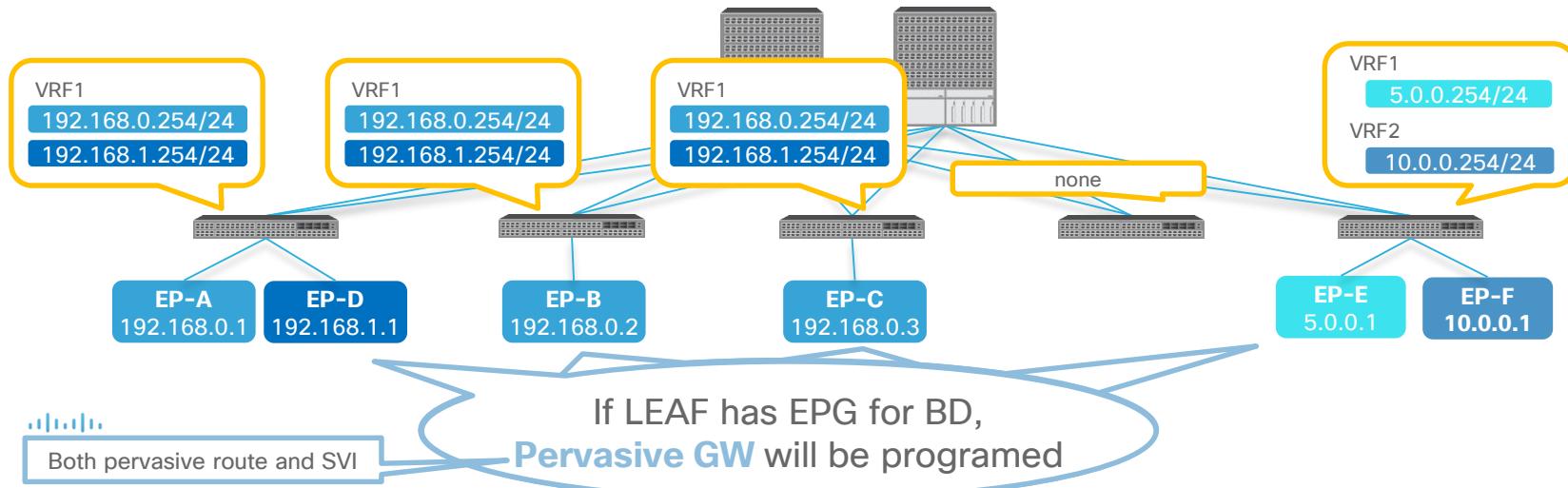
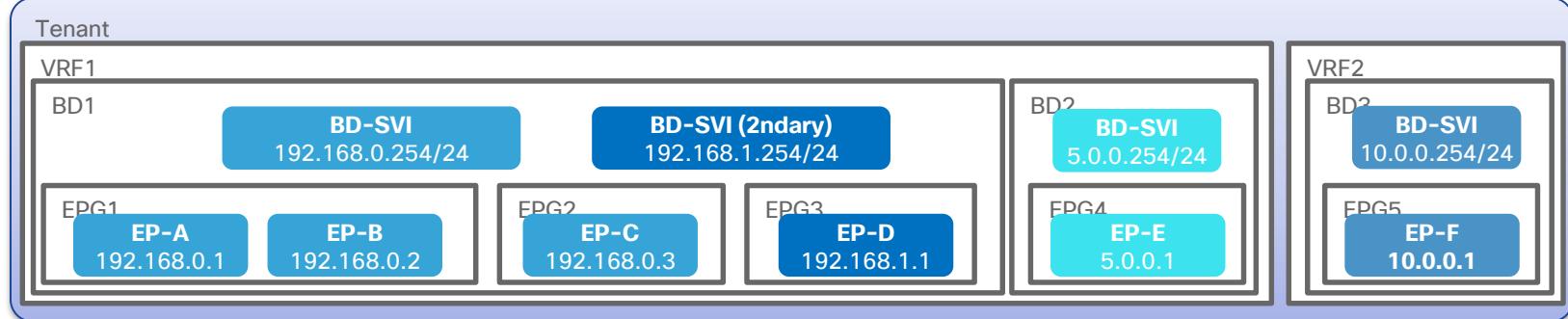
What is pervasive GW for?

- To be a default GW for EPs in the Fabric
 - All EPs can have consistent gateway IP address one hop away
- To represent subnets(IP ranges) for a BD
 - ACI knows which BD may have potential hidden/silent EPs

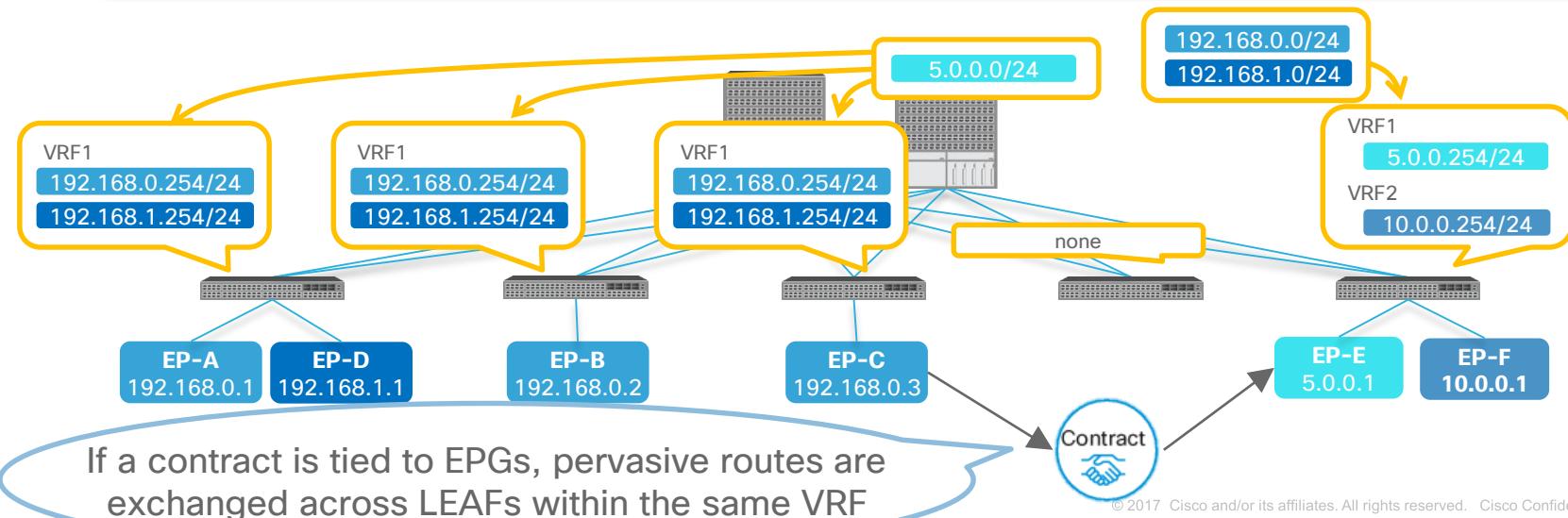
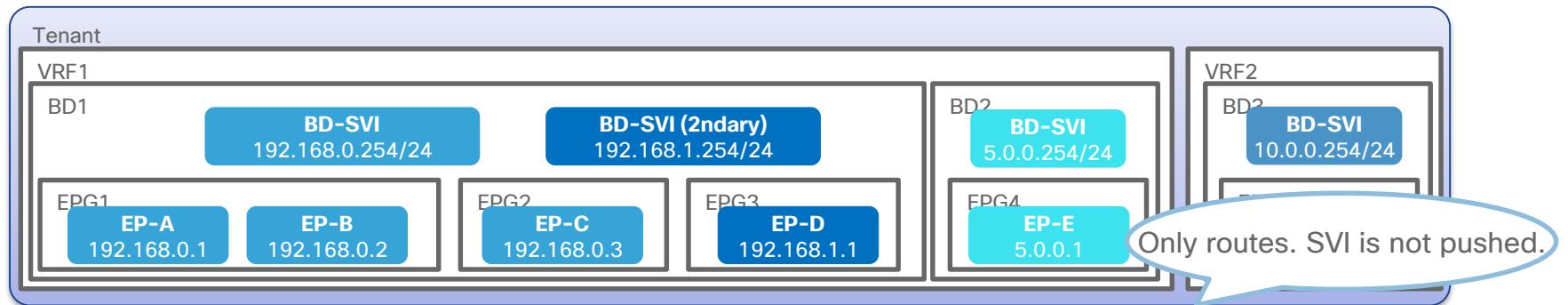
How is pervasive GW deployed?

- Installed as an SVI on LEAFs
 - PI-VLAN for BD is used to represent a pervasive GW SVI
 - A pervasive SVI has secondary IP when multiple pervasive GWs are configured on the same BD
 - User can choose a primary address

Pervasive Gateway(BD SVI) example



Pervasive Gateway(BD SVI) example



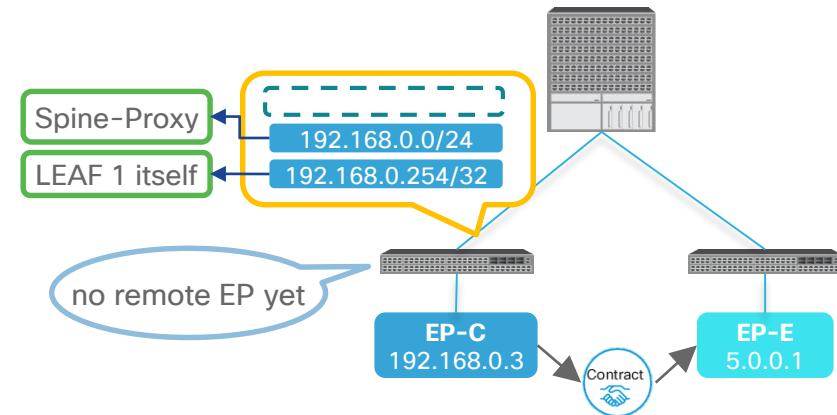
Pervasive Gateway(BD SVI) cont.

Why does ACI push pervasive routes to other LEAFs after a contract?

- Pervasive routes are required for Spine-Proxy



what if no pervasive route, no remote EP?

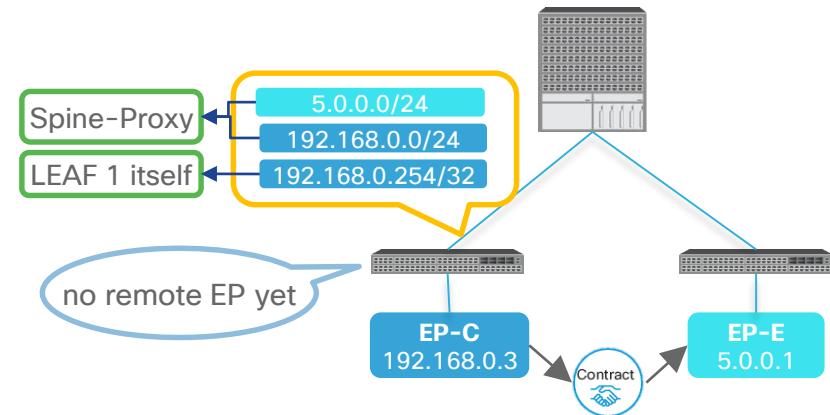


No Spine-Proxy for 5.0.0.1

It may be either dropped or forwarded to L3OUT if a default route exists



with pervasive route and no remote EP?



Spine-Proxy for 5.0.0.1

With the contract, ACI knows the LEAF needs to reach out to 5.0.0.0/24

Pervasive Gateway



Without contract

```
L101# show ip route vrf TK:VRF1
```

```
192.168.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 04:32:16, static  
192.168.0.254/32, ubest/mbest: 1/0, attached  
*via 192.168.0.254, vlan10, [1/0], 04:32:16, local, local
```

```
L103# show ip route vrf TK:VRF1
```

```
5.0.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 00:00:06, static  
5.0.0.254/32, ubest/mbest: 1/0, attached  
*via 192.168.2.254, vlan13, [1/0], 00:00:06, local, local
```

Exchange pervasive route

With contract

```
L101# show ip route vrf TK:VRF1
```

```
192.168.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 04:32:27, static  
192.168.0.254/32, ubest/mbest: 1/0, attached  
*via 192.168.0.254, vlan10, [1/0], 04:32:27, local, local  
5.0.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 00:00:02, static
```

```
L103# show ip route vrf TK:VRF1
```

```
192.168.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 00:00:10, static  
5.0.0.0/24, ubest/mbest: 1/0, attached, direct, pervasive  
*via 10.0.184.64%overlay-1, [1/0], 00:00:32, static  
5.0.0.254/32, ubest/mbest: 1/0, attached  
*via 192.168.2.254, vlan13, [1/0], 00:00:32, local, local
```

- Pervasive routes are pushed to other LEAFs with contracts



ACI BD Forwarding Option

The screenshot shows the Cisco ACI Bridge Domain configuration interface for Tenant TK. The left sidebar lists various tenant and networking configurations. The main window displays the properties for Bridge Domain BD1. A red box highlights the "L3 Configurations" tab, which is selected. A blue callout bubble points to a list of forwarding options:

- Unicast Routing
- L2 Unknown Unicast
- L3 Unknown Multicast Flooding
- Multi Destination Flooding
- ARP Flooding

Below the "L3 Configurations" tab, several forwarding options are listed with their current values:

- L2 Unknown Unicast: Flood (highlighted by a red box), Hardware Proxy
- L3 Unknown Multicast Flooding: Flood (highlighted by a red box), Optimized Flood
- Multi Destination Flooding: Flood in BD (highlighted by a red box), Drop, Flood in Encapsulation

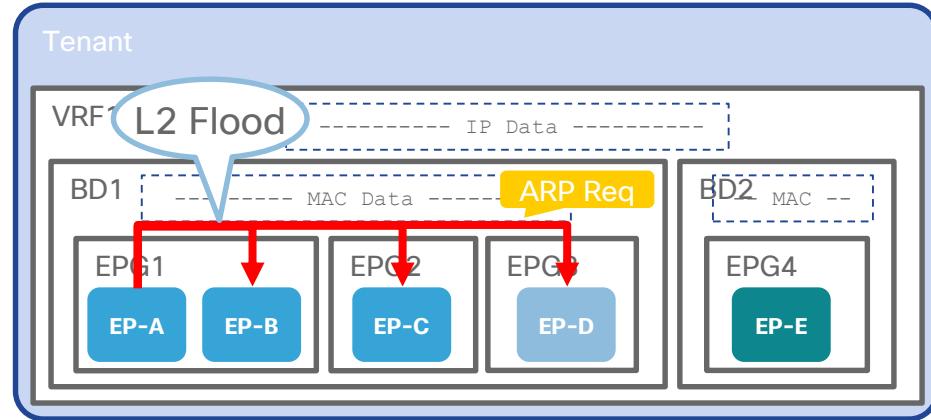
Other settings shown include PIM (unchecked), IGMP Policy (select an option), ARP Flooding (checked), Endpoint Dataplane Learning (checked), Limit IP Learning To Subnet (checked), End Point Retention Policy (select a value), and IGMP Snoop Policy (select a value). A red box highlights the "Unicast Routing" checkbox in the "Properties" section, which is checked. The operational value for Unicast Routing is set to true. The custom MAC address is 00:22:BD:F8:19:FF, and the virtual MAC address is Not Configured.

※ Please check a whitepaper “ACI Fabric EP Learning” for EP learning options
<https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-739989.html>

ARP Flooding

ARP Flooding: On

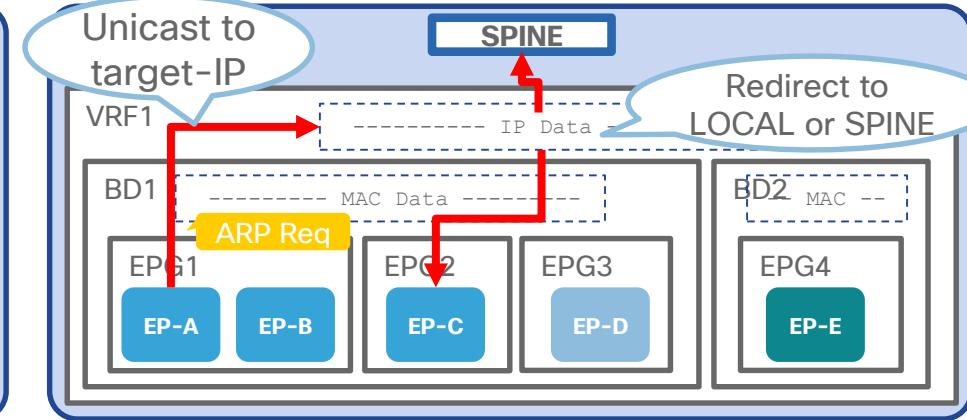
- ARP Flood On



Always **flood** ARP Request **within the same BD**

- Flood as broadcast if DST-MAC is FFFF.FFFF.FFFF
- Flood to other Leaf switches through Spine
- EP IP Data is not used for forwarding but still Sender-IP is learned if Unicast Routing is enabled.
- Good option when BD is supposed to be pure L2 without Unicast Routing like legacy VLAN

- ARP Flood Off (= Spine-Proxy)



ARP Request is handled as **L3 Unicast** with Target-IP

- If IP is **learned** on ingress Leaf,
 - Ingress Leaf forwards ARP Req **directly to dest**
- If IP is **not learned** on ingress Leaf,
 - Ingress Leaf forwards ARP Req **to Spine** (Spine-Proxy)
Spine will forward it to Leaf on which DstIP resides
- If IP is **not learned** even on Spine,
 - Drop and **ARP Glean** (only within BD)

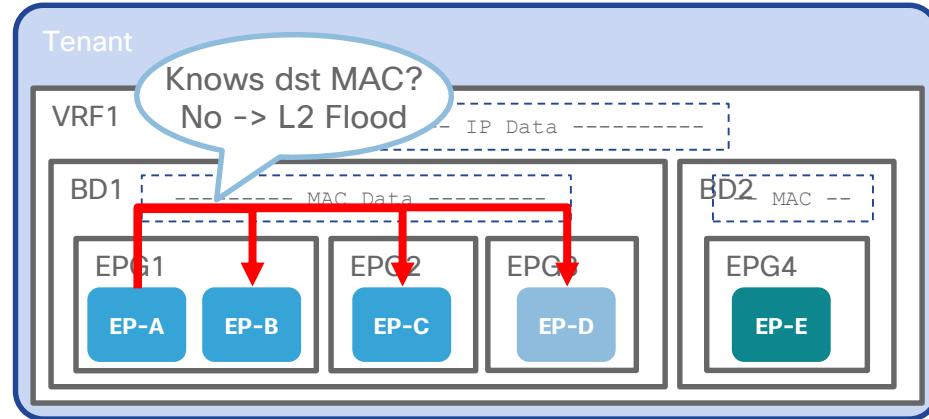


* ARP is not filtered by a contract by default

* if dst mac is not bcast, ARP request is bridged based on dst mac regardless of ARP Flooding mode

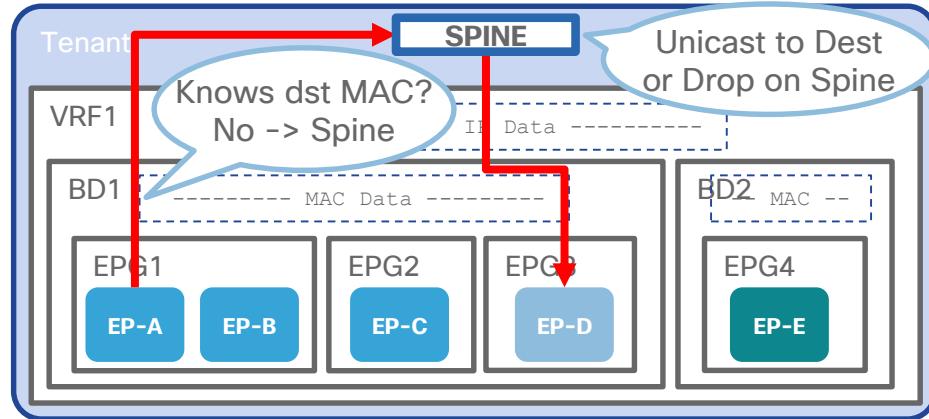
L2 Unknown Unicast

- Flood



L2 Unknown Unicast: Flood Hardware Proxy

- Hardware Proxy (= Spine-Proxy)



Always **flood** L2 Unknown Unicast **within the same BD**

- Flood as well as legacy VLAN.
- Flood happens locally and on other Leaf switches.
- Good option when BD is supposed to be pure L2 without Unicast Routing as in legacy VLAN
- Good option when there are silent L2 hosts

L2 Unknown Unicast is sent to Spine-Proxy

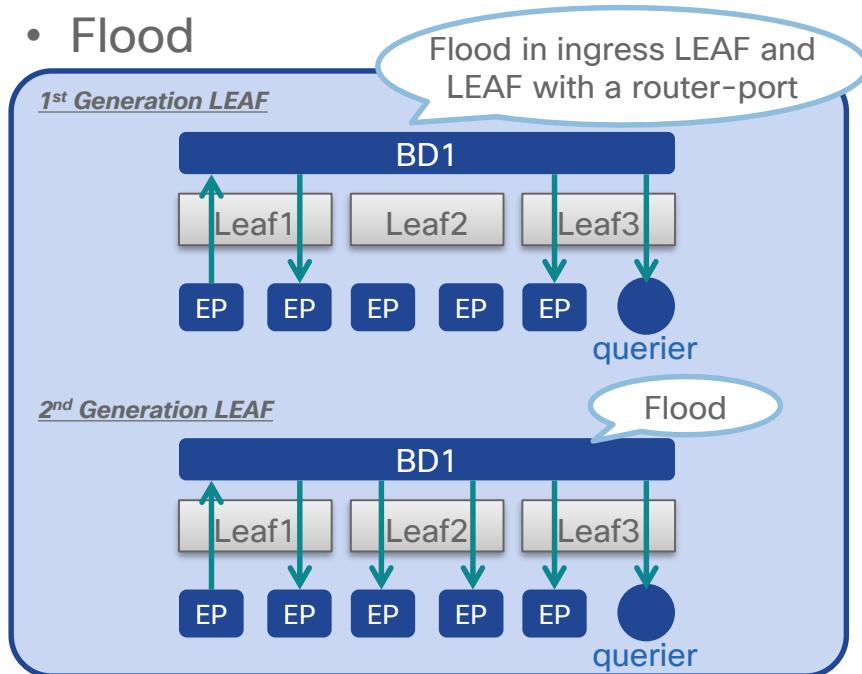
- If DST-MAC is **learned** on Spine,
 - Spine forwards it **directly to dest Leaf**
- If DST-MAC is **not learned** even on Spine
 - Drop

※EP IP Data is not used even if Leaf knows DST-IP. L2 Unknown is still L2 traffic.

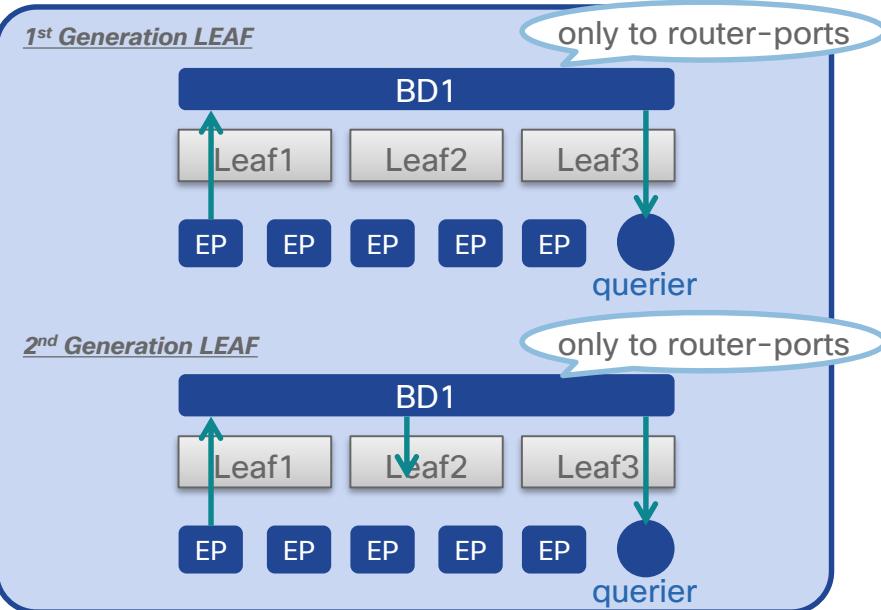
L3 Unknown Multicast Flooding

L3 Unknown Multicast Flooding: Flood Optimized Flood

- Flood



- OMF (Optimized Multicast Flood)



L3 Unknown Multicast = IP multicast group unknown to LEAF IGMP snooping

➤ Controls flooding *unknown* IGMP snooping groups

End point learning,move
and aging high level

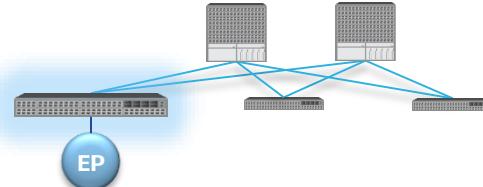
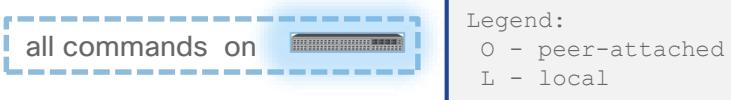
End Point Types in ACI

Physical Local Endpoint (PL)

- An endpoint attached to this LEAF

```
fab1-leaf1# show endpoint ip 192.168.0.51
```

19	vlan-5	0000.5555.1111	L	eth1/1
TK:VRF1	vlan-5	192.168.0.51	L	eth1/1

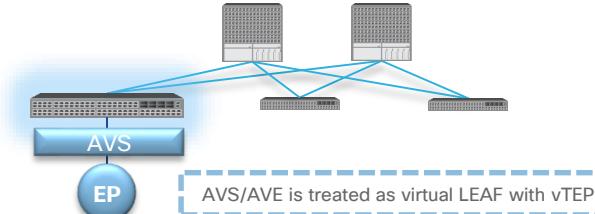


Virtual Local Endpoint (VL)

- An endpoint on AVS/AVE (or any vxlan attached) attached to this LEAF

```
fab1-leaf1# show endpoint ip 192.168.66.2
```

14	vxlan-8388608	0050.5680.34eb	L	tunnel10
TK:VRF1	vxlan-8388608	192.168.66.2	L	tunnel10



Remote Endpoint (Xr)

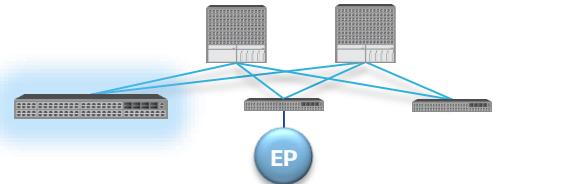
- An endpoint on a remote LEAF

```
fab1-leaf1# show endpoint mac 0000.5555.2222
```

17/TK:VRF1	vxlan-15826915	0000.5555.2222	tunnel8
------------	----------------	----------------	---------

Access Encap VLAN (VxLAN)

BD VNID (not Access Encap VLAN)



```
fab1-leaf1# show endpoint ip 192.168.0.52
```

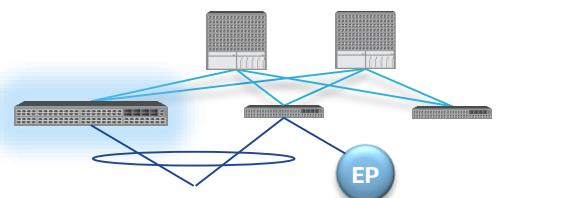
TK:VRF1	192.168.0.52	tunnel8
---------	--------------	---------

On-Peer Endpoint

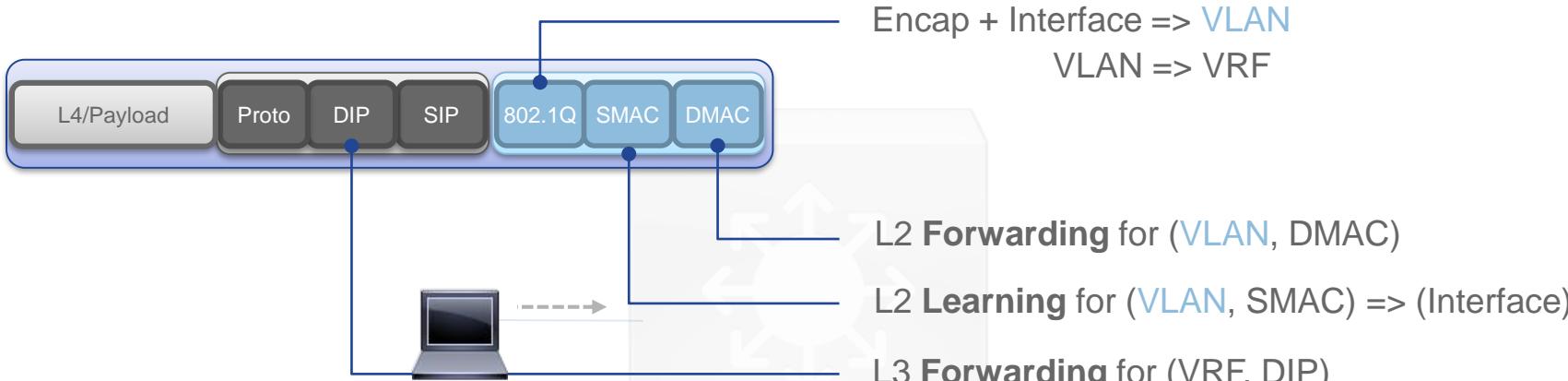
- An endpoint connected to an orphan port on vPC peer

```
fab1-leaf1# show endpoint ip 192.168.0.52
```

19	vlan-5	0000.5555.2222	O	tunnel8
TK:VRF1	vlan-5	192.168.0.52	O	tunnel8



Classical Learning



L2 Forwarding:

- (**VLAN**, DMAC) Miss => Flood
- (**VLAN**, DMAC) Gateway MAC => Route
- (**VLAN**, DMAC) Hit => Destination Port

L3 Forwarding (Longest Prefix Match)

- (**VRF**, DIP) Miss => Drop
- (**VRF**, DIP) Hit=> Adjacency

config on destination port + VLAN
determines egress encap
(tagged or untagged)

Might be Glean or packet rewrite (SMAC, DMAC,
VLAN, etc...), may include destination port in adjacency
or require second L2 lookup on new DMAC

Classical Learning

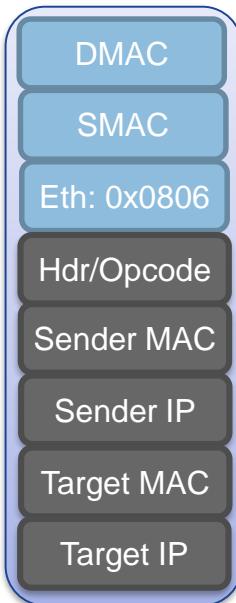
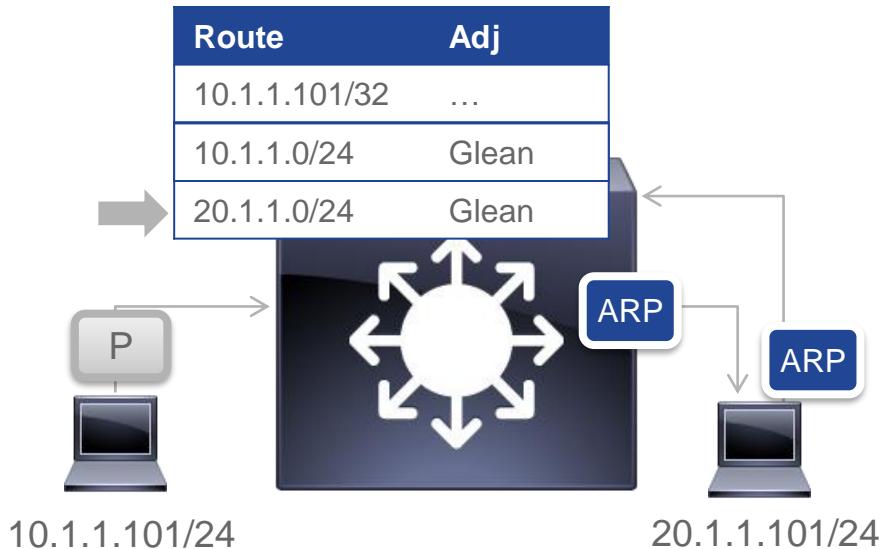
ARP Packet

LPM Routes

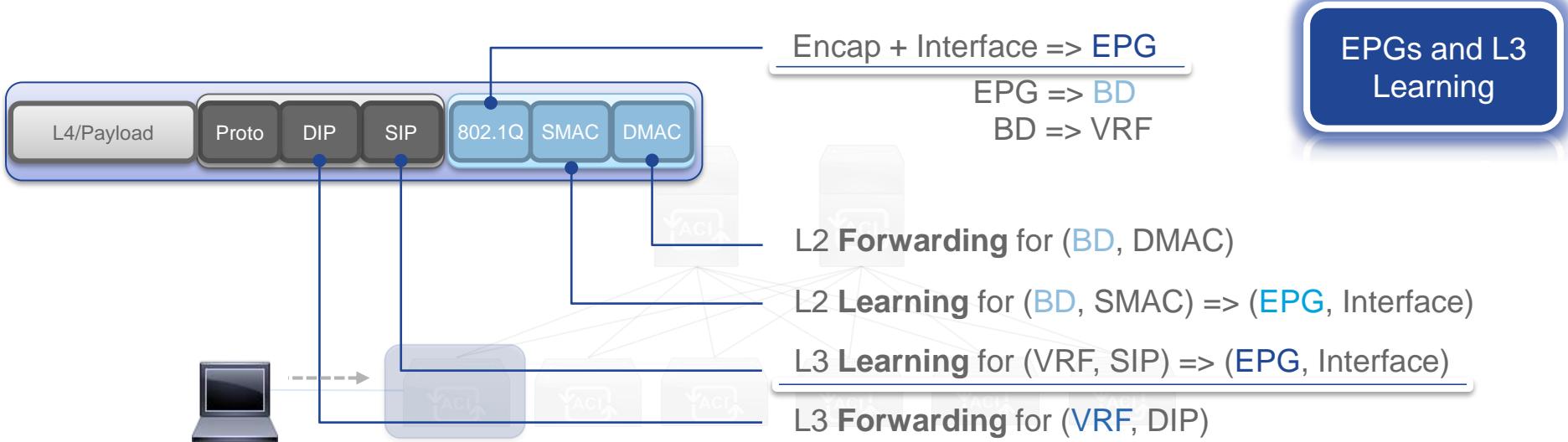
- Connected/direct routes manually configured
- Static/dynamic routing protocols to learn prefixes

Host Routes (*IP Endpoints*)

- **Glean** adjacency for connected routes to punt frame and generate ARP request
- **ARP/ND** used to create MAC to IP binding and install host route into routing table



ACI Learning (Physical Local - PL)



L2 Forwarding:

- (BD, DMAC) Miss => (Flood/**Proxy**+Drop)
- (BD, DMAC) Gateway MAC => Route
- (BD, DMAC) Hit => Adjacency

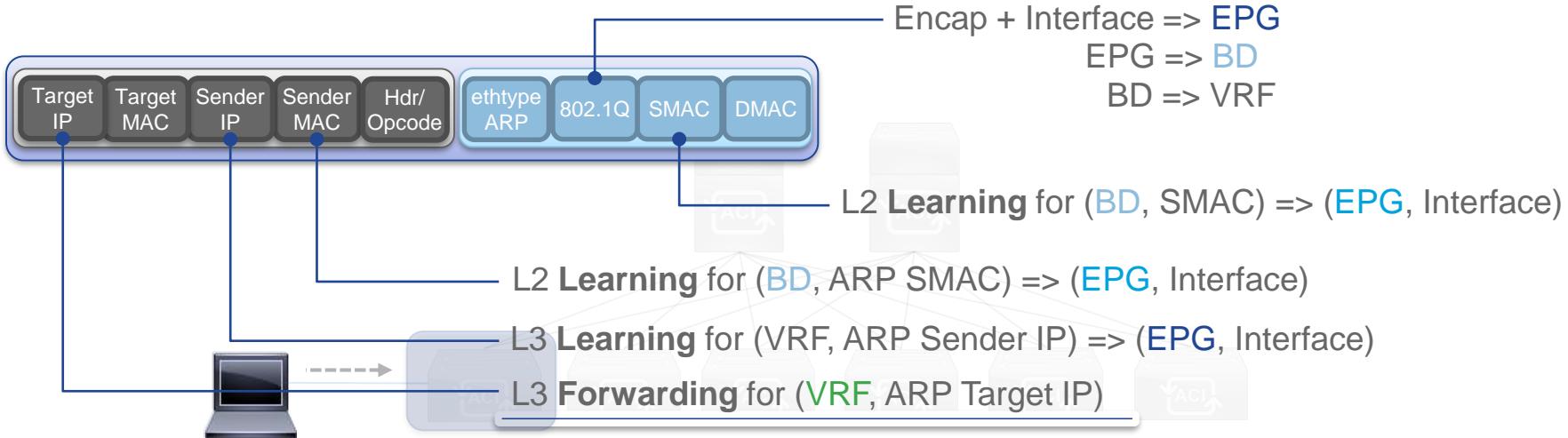
L3 Forwarding (Longest Prefix Match)

- (VRF, DIP) Miss => Drop
- Proxy/Glean for BD subnets
- (VRF, DIP) Hit=> Adjacency

Adjacency contains dst EPG, encap information, dst VTEP or port, etc...
More in upcoming slides

ACI Learning (ARP)

Optimize Forwarding
(ARP Flooding disabled)



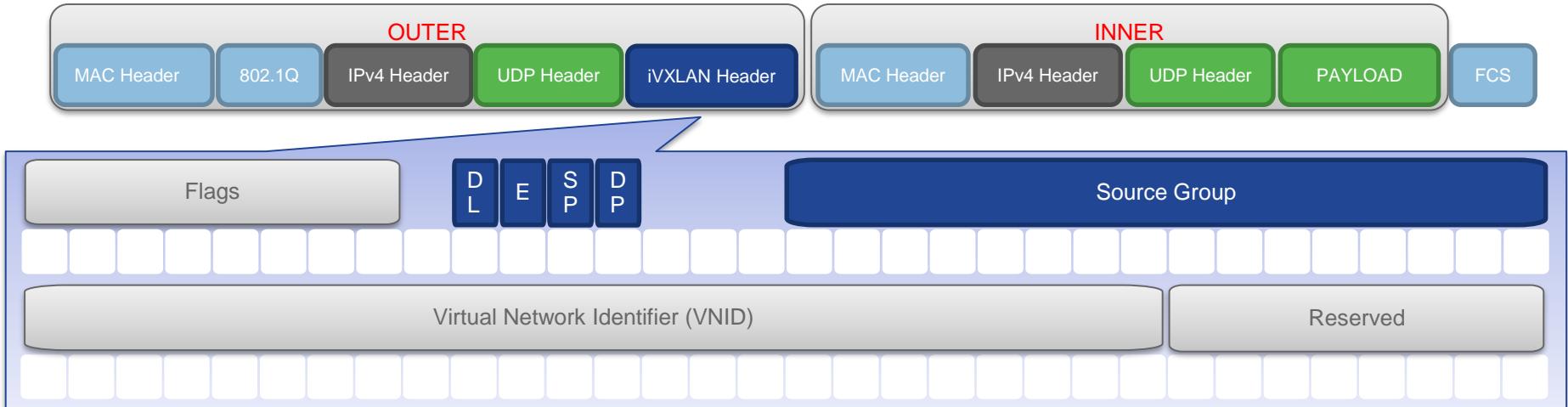
ARP L3 Forwarding

(VRF, ARP Target IP) Miss => Proxy
(VRF, ARP Target IP) Hit=> Adjacency

L3 forwarding based on ARP target IP field
with miss sent to spine proxy ☺

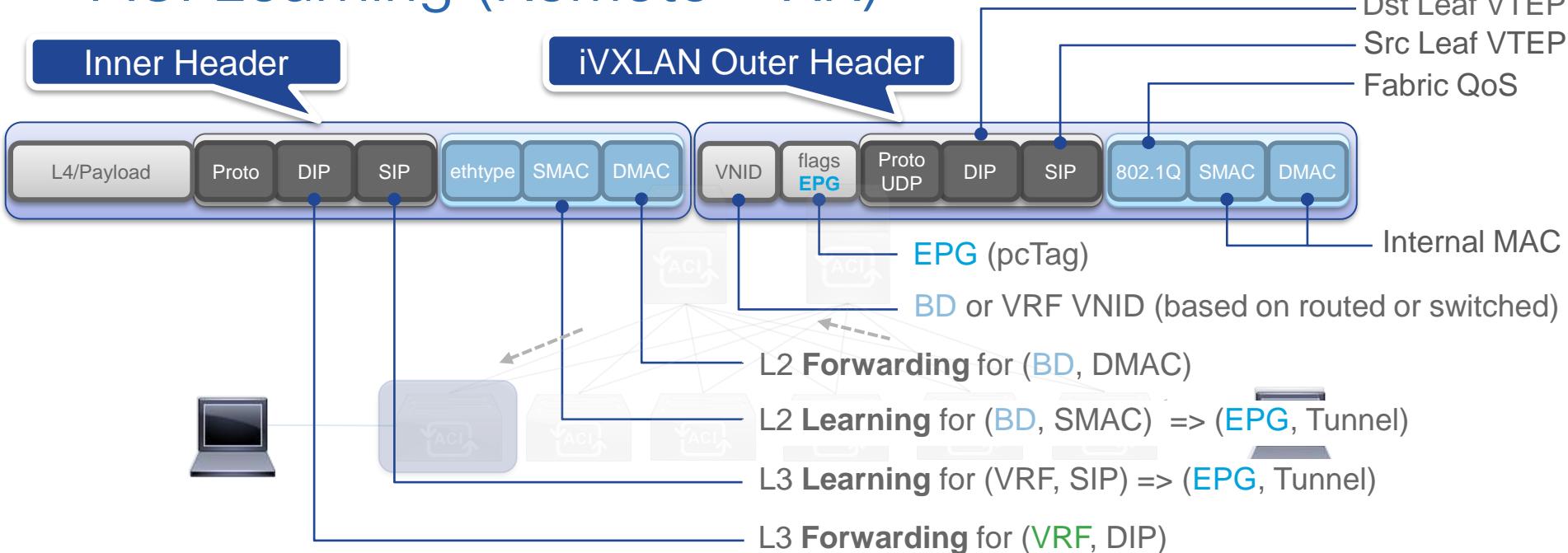


iVXLAN Header



Abbr.	Name	Description
DL	Do not learn	Informs remote leaf that it should not perform dataplane learning from this frame
E	Exception	Set when frame has gone through proxy path
SP	Source-policy-applied	Policy has already been applied to this frame
DP	Destination-policy-applied	- (DP and SP are always set together)
sclass/pcTag	Source group (policy-control tag)	16-bit policy control tag representing the EPG that sourced the frame

ACI Learning (Remote - XR)



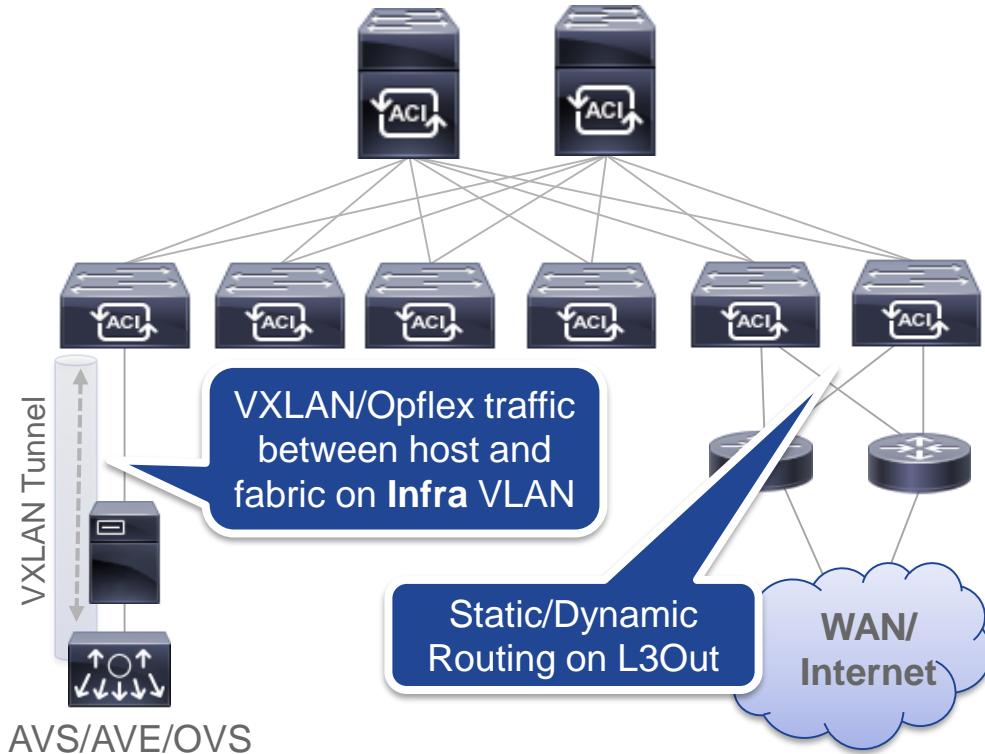
ACI Learning

Learning Exceptions

- No IP EP learning if routing is disabled on the BD
- No IP EP learning on **external** BD's (Layer-3 Outside interfaces)
- No IP EP learning on **Infra** VLAN
- No IP learning of shared service prefixes outside of our VRF

LPM Routes (Same as Classical)

- Pervasive SVI Routes (BD Subnets)
- Static and dynamic routing protocols on L3Out

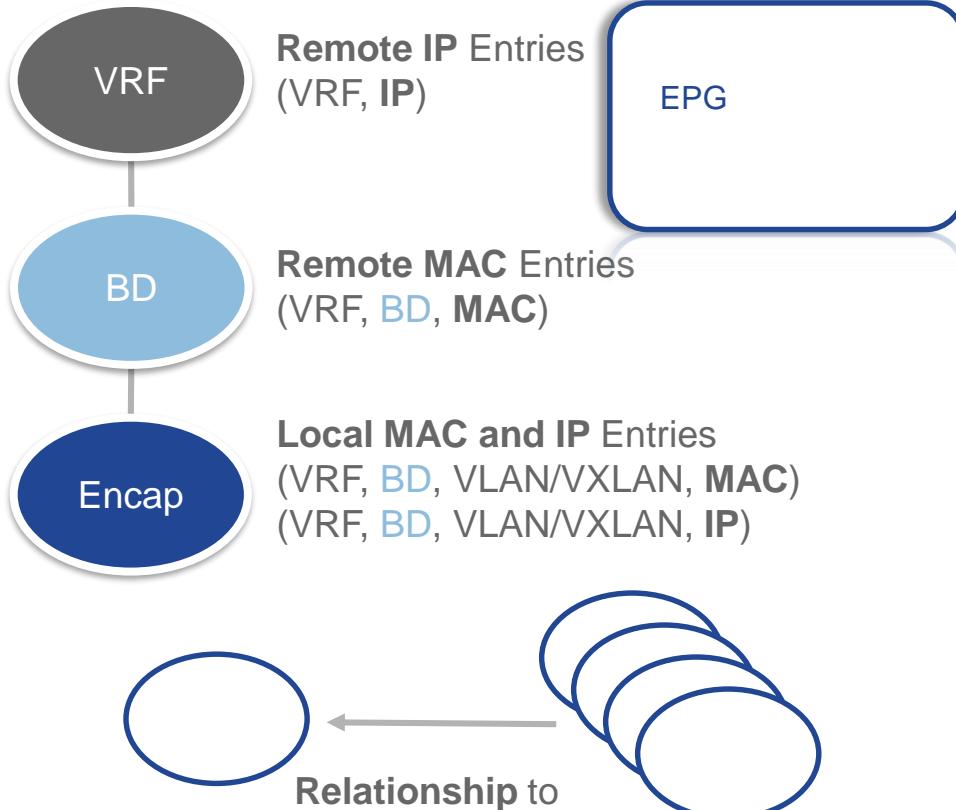


ACI Learning

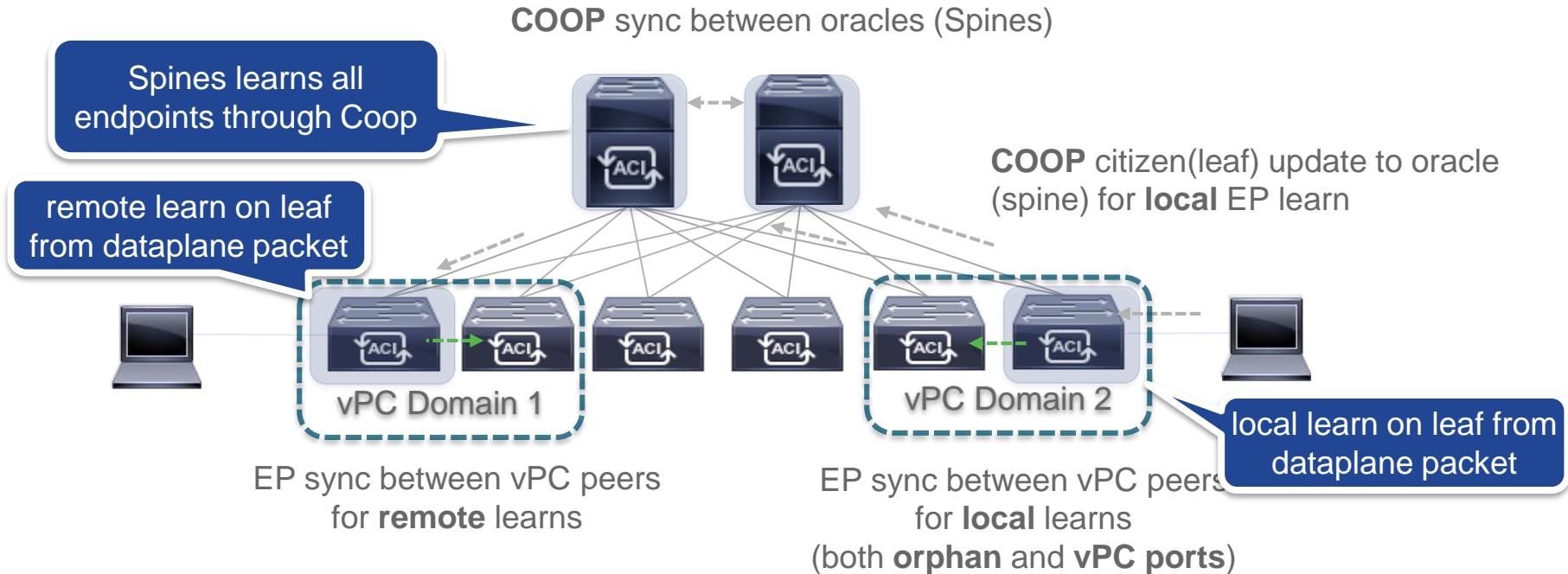


Frame	Forwarding Operation	Learn
Non-IP/IP	Bridged	MAC
ARP	-	MAC (sender-HW), IP (sender-IP)
IPv4	Unicast Routed	MAC, IP
IPv6	Unicast Routed	MAC, IP
IPv6	Neighbor Discovery	MAC, IP

Leaf Endpoint Database



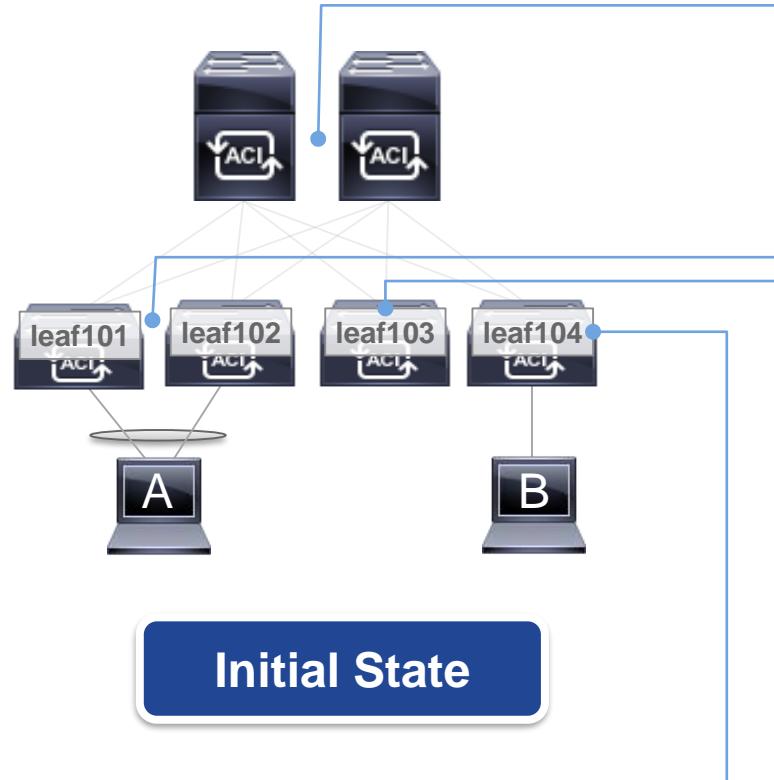
ACI Learning (COOP and EP Sync)



ACI Learning: Review

- **MAC** learning for all frames
- **IP** learning for routed packets and ARP packets
- No **IP** learning on frames received on L3Out or Infra vlan
- **All local endpoint learns** are published to coop
 - spine has full knowledge of all fabric endpoints
 - Proxy forwarding for any fabric endpoint allowing for zero-penalty impact for remote endpoint miss

Moves and Bounce



Spines

Addr	Interface	Detail
A	tun1001	leaf101/102 vTEP
B	tun4	leaf104 TEP

Leaf101/102

Addr	Interface	Detail
A	vpc1	local vpc
B	tun4	XR -> leaf104

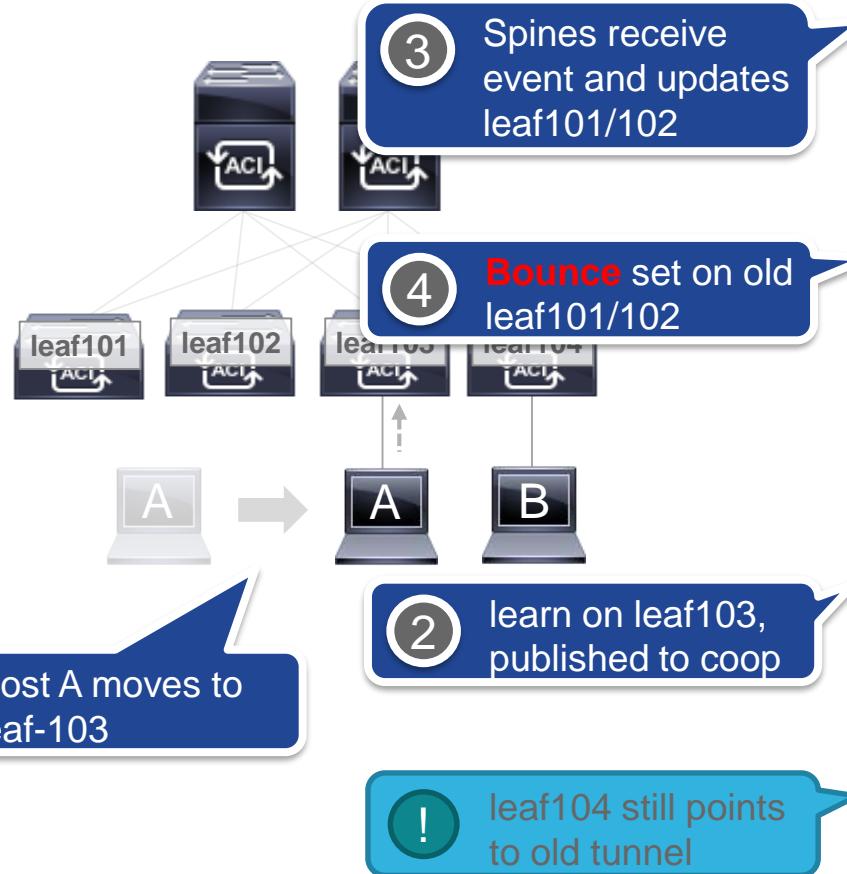
Leaf 103

Addr	Interface	Detail
-	-	-
-	-	-

Leaf 104

Addr	Interface	Detail
A	tun1001	XR -> leaf101/102 VIP
B	eth1/1	local learn

Moves and Bounce



Spines

Addr	Interface	Detail
A	tun3	leaf103 TEP
B	tun4	leaf104 TEP

Leaf101/102

Addr	Interface	Detail
A	tun3, bounce	XR -> leaf103 with bounce bit set
B	tun4	XR -> leaf104

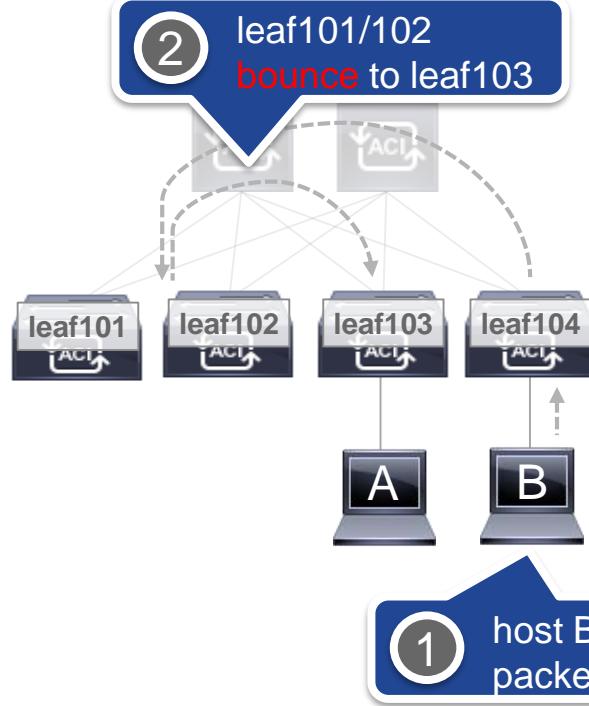
Leaf 103

Addr	Interface	Detail
A	eth1/1	local learn from 1 st packet
-	-	-

Leaf 104

Addr	Interface	Detail
A	tun1001	XR -> leaf101/102 VIP
B	eth1/1	local learn

Moves and Bounce



Spines

Addr	Interface	Detail
A	tun3	leaf103 TEP
B	tun4	leaf104 TEP

Leaf101/102

Addr	Interface	Detail
A	tun3, bounce	XR -> leaf103 with bounce bit set
B	tun4	XR -> leaf104

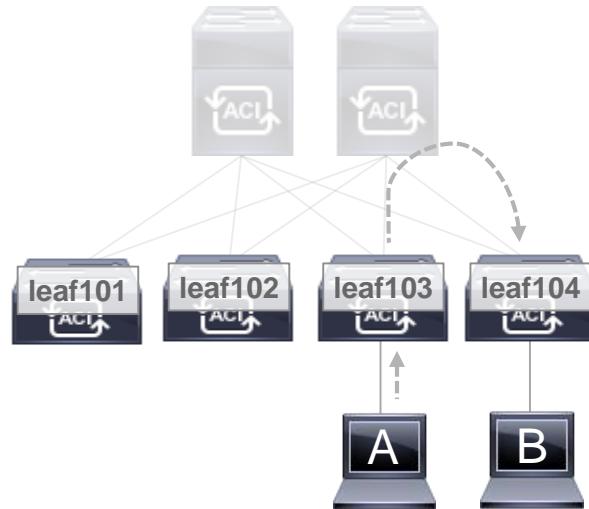
Leaf 103

Addr	Interface	Detail
A	eth1/1	local learn
B	tun4	XR -> leaf104

Leaf 104

Addr	Interface	Detail
A	tun1001	XR -> leaf101/102 VIP
B	eth1/1	local learn

Moves and Bounce



4

host A sends packet to host B

5

leaf104 updates XR to leaf103

Spines

Addr	Interface	Detail
A	tun3	leaf103 TEP
B	tun4	leaf104 TEP

Leaf101/102

Addr	Interface	Detail
A	tun3, bounce	XR -> leaf103 with bounce bit set
B	tun4	XR -> leaf104

Leaf 103

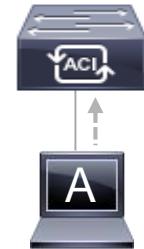
Addr	Interface	Detail
A	eth1/1	local learn
B	tun4	XR -> leaf104

Leaf 104

Addr	Interface	Detail
A	tun3	XR -> leaf103 TEP
B	eth1/1	local learn

Aging

Addr	Time-left	Reset-count	Hit
A	900 second	225	Yes

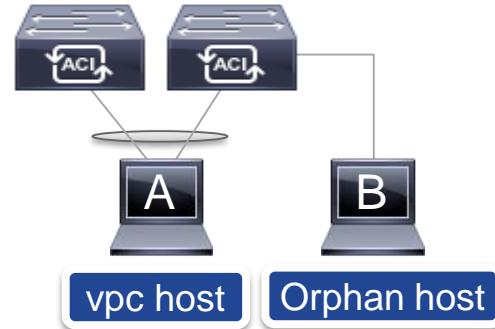


- Hardware maintains hit-bit for each entry which is set whenever a frame is received from corresponding source address
- If packet is *not seen* within timeout, then entry is aged and removed from hardware
- Else if leaf receives a frame and hit-bit is set, then software resets timer and hit bit and entry is not aged out.
- For **local IP** endpoints, at **75% of endpoint timer**, then host tracking sends 3x ARP/ND to verify if endpoint is still present
 - ARP/ND reply resets timer for both **IP** and **MAC**
 - Support for silent hosts
 - No response and endpoint will eventually age-out

No regular ARP/ND required to verify IP is still present if traffic is regularly received!

VPC Aging

Addr	Hit	Flags
A	No	local, vpc-attached
B	No	peer-attached



Addr	Hit	Flags
A	No	local, vpc-attached
B	No	local

- For vpc, both leaves in the vpc domain have to age out the entry before it is removed. This applies to remote and local entries
- For orphan ports, as soon as the local leaf ages it out it is deleted from both switches.

VPC Aging

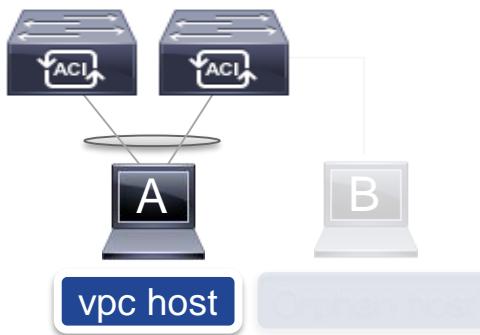
2

Peer-aged flag set indicating that peer has aged the entry. Will be deleted once local leaf ages out it as well.

1

When vpc endpoint is aged, set local-aged flag and send update to peer

Addr	Hit	Flags
A	No	local, vpc-attached peer-aged
B	No	peer-attached



Addr	Hit	Flags
A	No	local, vpc-attached local-aged
B	No	local

VPC Aging

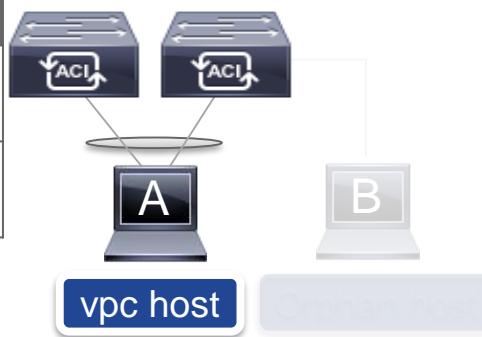
3

Endpoint is locally-aged, send update to peer. Since both local-aged and peer-aged is set, delete entry

4

Receive peer-aged from peer. Since both local-aged and peer-aged is set, delete entry

Addr	Hit	Flags
A	No	local, peer-attached peer-aged, local-aged
B	No	peer-attached



Addr	Hit	Flags
A	No	local, peer-attached local-aged, peer-aged
B	No	local

VPC Aging

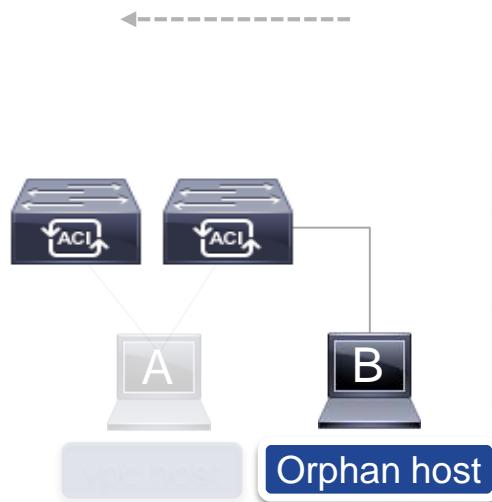
2

Orphan port deleted as soon as peer ages it out

1

When orphan port is locally-aged, simply delete and send update to peer

Addr	Hit	Flags
A	No	local, vpc-attached
B	No	peer-attached



Addr	Hit	Flags
A	No	local, vpc-attached
B	No	local local-aged

Timers – Endpoint Retention Policy

Timer	Default	Applied at BD	Applied at VRF
Local	900 sec	Mac and IP	-
Bounce	630 sec	Mac	IP
Remote	300 sec	Mac	IP
Move	256/sec	-	-
Hold	300 sec	-	-

XR MACs are always learned at BD level

XR IP's are always learned at VRF level

- If moves/sec exceed rate then ***learning is disabled*** on BD for the hold time as a protection mechanism for software components (epm/ePMC/coop)



Timers – Endpoint Retention Policy

System **Tenants** Fabric Virtual Networking L4-L7 Services Admin Operations Apps Integrations

ALL TENANTS | Add Tenant | Tenant Search: name or descr | common | ag | mgmt | scale | infra

ag

> Quick Start

ag

> Application Profiles

> Networking

Bridge Domains

- bd1
 - DHCP Relay Labels
 - Subnets
 - 10.1.1.1/24
 - 10.1.1.2/24
 - 10.1.1.3/24
 - ND Proxy Subnets
- bd2
- bd3
- bd4
- VRFs
- External Bridged Networks
- External Routed Networks
- Dot1Q Tunnels
- Contracts
- Policies
- Services

Bridge Domain - bd1

Properties

Type: fc regular

Advertise Host Routes:

Enable Legacy Mode:

Legacy Mode: No

VLAN:

VRF: v1

Resolved VRF: ag/v1

L2 Unknown Unicast: Flood Hardware Proxy

L3 Unknown Multicast Flooding: Flood Optimized Flood

IPv6 L3 Unknown Multicast: Flood Optimized Flood

Multi Destination Flooding: Flood in BD Drop Flood in Encapsulation

PIM:

IGMP Policy: select an option

ARP Flooding:

IP Data-plane Learning: no yes

Limit IP Learning To Subnet:

Endpoint Retention Policy: select a value

This policy only applies to local L2, L3, and remote L3 entries

IGMP Snoop Policy: select a value

MLD Snoop Policy: select a value

Summary **Policy** Operational Stats Health Faults History

General L3 Configurations Advanced/Troubleshooting

Create End Point Retention Policy

Name: custom-timers

Description: optional

Hold Interval (sec): 300

Bounce Entry Aging Interval (sec): 630

Local Endpoint Aging Interval (sec): 900

Remote Endpoint Aging Interval (sec): 300

Move Frequency (per sec): 256

Custom Aging Timers at BD level

Cancel Submit

Show Usage Reset Submit



Timers – Endpoint Retention Policy

System **Tenants** Fabric Virtual Networking L4-L7 Services Admin Operations Apps Integrations

ALL TENANTS | Add Tenant | Tenant Search: name or descr | common | **ag** | mgmt | scale | infra

ag

> Quick Start

ag

> Application Profiles

> Networking

> Bridge Domains

> bd1

> bd2

> bd3

> bd4

> VRFs

> v1

> Multicast

> EPG Collection for VRF

> v2

> v3

> External Bridged Networks

> External Routed Networks

> Dot1Q Tunnels

> Contracts

> Policies

> Services

VRF - v1

Properties

OSPF Timers: select a value

OSPF Context Per Address Family:

OSPFA Timers

No items have been found.
Select Actions to create a new item.

Endpoint Retention Policy: custom-timers

This policy only applies to remote L3 endpoints.

Monitoring Policy: select a value

EIGRP Context Per Address Family:

EIGRP Timers

No items have been found.
Select Actions to create a new item.

Create SNMP Context:

Create Route Target Profile:

DNS labels: enter names separated by comma

Route Tag Policy: select a value

IP Data-plane Learning: Disabled Enabled

Enable GOLF-OPFLEX MODE:

Summary **Policy** Operational Stats Health Faults History

Show Usage Reset Submit

Custom Aging Timers at VRF level

Endpoint learning configuration: IP routing

- If packet is bridged the mapping database learns: BD, MAC
- If packet is **routed** the mapping database learns both (BD, MAC) and (VRF, IP)
- **If you want IPs to be learned you need IP ROUTING ENABLED**

The image shows a screenshot of a Cisco endpoint learning configuration interface. At the top, there is a checkbox labeled "Unicast Routing:" which is checked and highlighted with a red border. Below this, there are two fields: "Custom MAC Address: 00:22:BD:F8:19:FF" and "Virtual MAC Address: Not Configured". Underneath these fields, the word "Subnets:" is followed by a table. The table has four columns: "Gateway Address", "Scope", "Primary IP Address", and "Virtual IP". A header row is present above the table. Below the table, a message states "No items have been found. Select Actions to create a new item." The Cisco logo is visible in the bottom left corner.

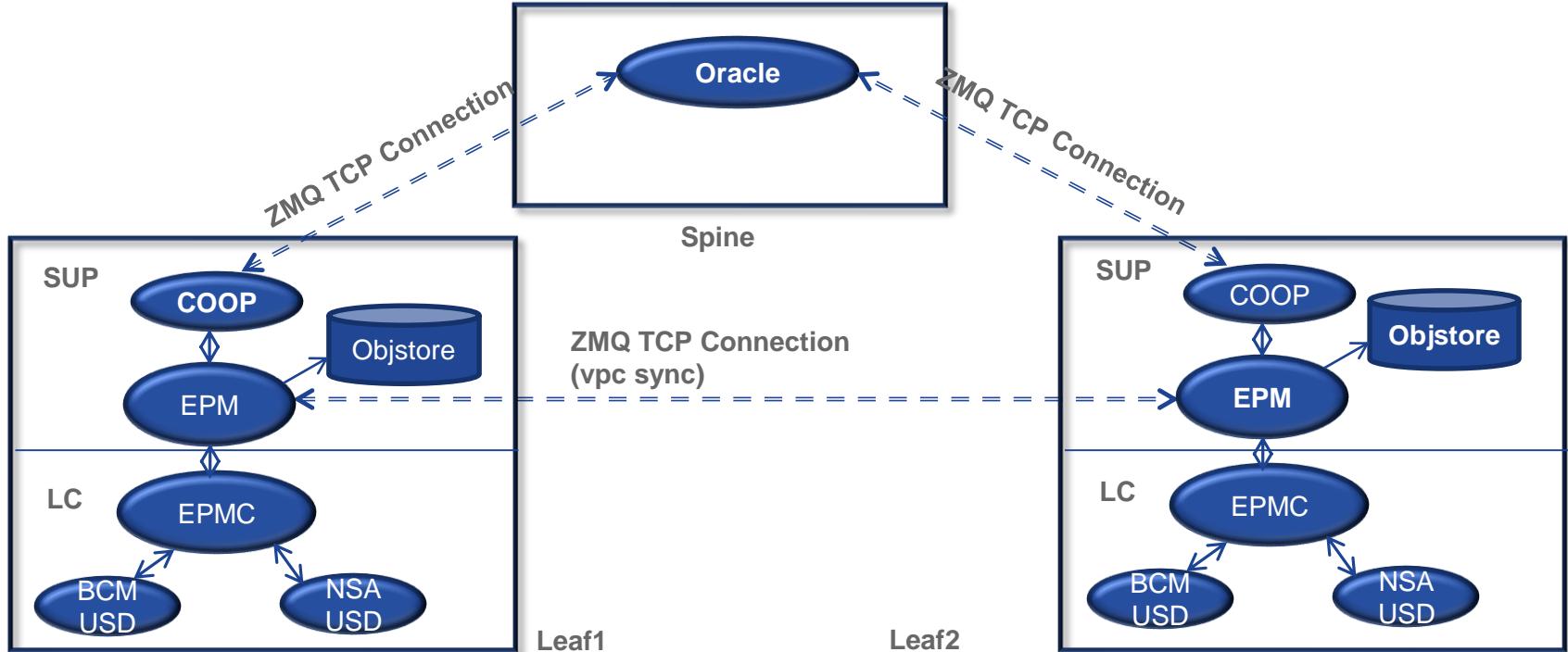
Gateway Address	Scope	Primary IP Address	Virtual IP
No items have been found. Select Actions to create a new item.			

End Point manager EPM – EPMC Process

Introducing EPM - End Point Manager

- A process Running on each leaf and is responsible of learning of end point information (IP , MAC or both)
Won't see it in routing table though
- Builds a ‘host routing table’ (~ /32 host routes or mac-address-table or ARP entry)
- Learns based :
 - All Local End point (EP) based on traffic received on downlink
 - Remote End point (EP) based on traffic received on fabric uplink (spine port)
 - Sync EP with vpc Peer if needed.
- Programs ASIC
- Informs the MO tree about learned info to update APIC about EP location
- Informs COOP about any local learning

EPM Software Architecture (pair of Leaf part of a vpc Domain shown here)



EPM – End point manager

EPM

- Handles Endpoint learning
- Publishes Endpoints in ObjectStore/mitfs
- Publishes Endpoints to Spine via COOP
- Syncs Endpoints between vPC peers

EPMC – End Point manager Client

EPMC

- Handles programming Endpoints in hardware
- Handles learning notifications from hardware
- Manage aging of endpoints

The type of learning of the endpoint can be derived from the EPG Operational view

EPG - app

Client End-Points							Configured Access Policies	Contracts
End Point	MAC	IP	Learning Source	Hosting Server	Reporting Controller Name	Interface	Encap	
VM-7	00:50:56:... ---	---	vmm	192.168.11.70	vcenter-br...	Node-101/eth1/15 (vmm)	vlan-35	
VM-8	00:50:56:... ---	---	vmm	192.168.11.71	vcenter-br...	Node-102/eth1/16 (vmm)	vlan-35	

VMM alone has no meaning from the dataplane perspective,
i.e. there may be zero entries in the endpoint tables even if
the VM is known due to the VMM

Flags of the endpoint can be of the following types

- **vmm**: from vCenter or SCVMM and so on, this is not indication of an entry in the dataplane (both real dataplane and ARP)
- **vmm, learn**: this means that both the Virtual Machine Manager and the dataplane (both real dataplane and ARP) provided this entry information
- **learn**: from ARP spoofing and/or dataplane forwarding
- **static**: manually entered
- **static,learn**: manually entered plus the entry is learned in the dataplane

End Point Learning and ARP

Review ARP packet format

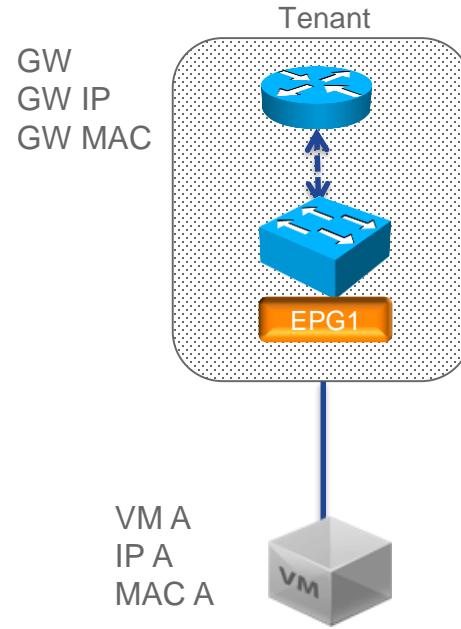
Internet Protocol (IPv4) over Ethernet ARP packet

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

- Operation :
 - 1 – Request
 - 2 – Reply
- SHA – THA – Mac address
- SPA – TPA – IP addresses

Typical ARP resolution

Packet flow when a VM needs to resolve
Its ARP gateway



Broadcast ARP req
from A to its Gateway

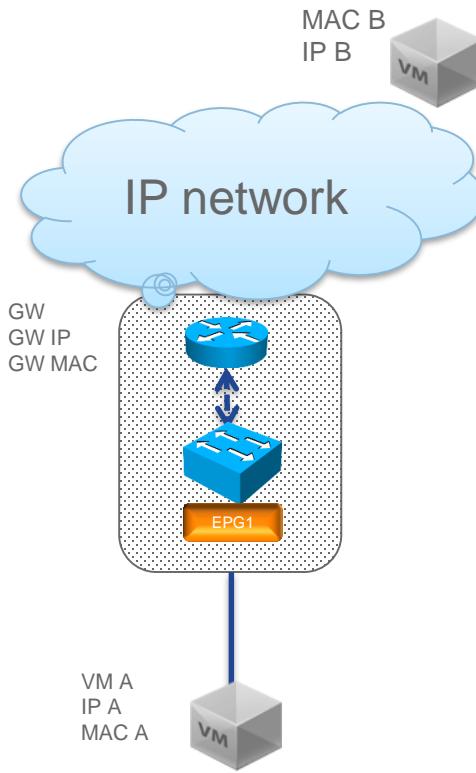
Unicast reply from Gw
to A

Dst MAC	Src MAC	Etype	OP	S HA	S IP	T HA	T IP
ffff.ffff.ffff	VM A MAC	0806	Req	VM A MAC	IP A	0000.0000.0000	GW IP
VM A MAC	GW MAC	0806	Reply	GW MAC	GW IP	VM A MAC	IP A

What is proxy ARP

Wikipedia :

Proxy ARP is a technique by which a proxy device on a given network answers the [ARP](#) queries for an [IP address](#) that is not on that network. The proxy is aware of the location of the traffic's destination, and offers its own [MAC address](#) as the (ostensibly final) destination

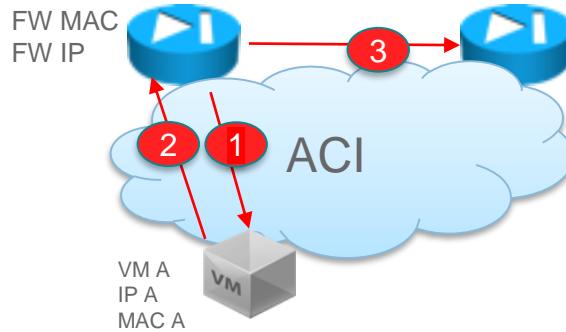


	Dst MAC	Src MAC	Etype	OP	SHA	S IP	T HA	TIP
Broadcast ARP req from A to resolve B	ffff.ffff.ffff	MAC A	0806	Req	MAC A	IP A	0000.0000.0000	IP B
Unicast reply from Gw to A with STA IP B	MAC A	GW MAC	0806	Reply	GW MAC	IP B	MAC A	IP A

Other “weird” ARP Behavior

Similar to Proxy ARP some devices may decided to send ARP for other devices than themselves (To sync arp table in cluster for example). Very often the Src MAC may differ from sender HA

Example : By default, all ARP Reply packets received by the Active Firewall cluster member are forwarded to other peer members. These ARP packets are re-broadcasted or multicasted from the Active Member.



	Dst MAC	Src MAC	Etype	OP	SHA	SIP	T HA	TIP
1	Broadcast ARP req from Fw to resolve A	ffff.ffff.ffff	FW MAC	0806	Req	FW MAC	IP FW	0000.0000.0000
2	Unicast reply from A to FW with STA IP A	FW MAC	MAC A	0806	Reply	MAC A	IP A	FA MAC
3	FW sync ARP -	ffff.ffff.ffff	FW MAC	0806	??	MAC A	IP A	FW MAC

Here 2 and 3 packets have same SHA and SIP But come on diff port, if we learn From payload arp, we would see A flapping on two ports

ACI learning behavior from ARP packet

Three parameters are important

- Unicast routing flag of the BD

- Leaf generation Gen-1 <> Gen2-3

- Local leaf <> remote leaf learning

Gen-2 Learning behavior from ARP

(tested with 3.2(4)d) on EX leaves

- Unicast routing disable
 - Do not learn anything locally , neither remotely
- Unicast routing enable
 - Learns locally Src MAC with Sender IP if Target IP is not BD GW
 - Learns locally Sender Mac with Sender IP if Target IP is BD GW
 - Learns remotely sender IP always

Gen-1 Learning behavior from ARP

(tested with 3.2(4)d) on gen-1 leaves

- Unicast routing disable
 - Remote learns Sender MAC alone
 - Local learns sender MAC with Src IP
- Unicast routing enable – remote learn always
 - Src IP to tunnel interface
 - Sender mac to Tunnel IF the Src IP was not learned already
 - DO NOT LEARN sender mac to tunnel if the src IP was Already learned for a previous reason
- Unicast routing enable – we always learn locally sender Mac and src IP together

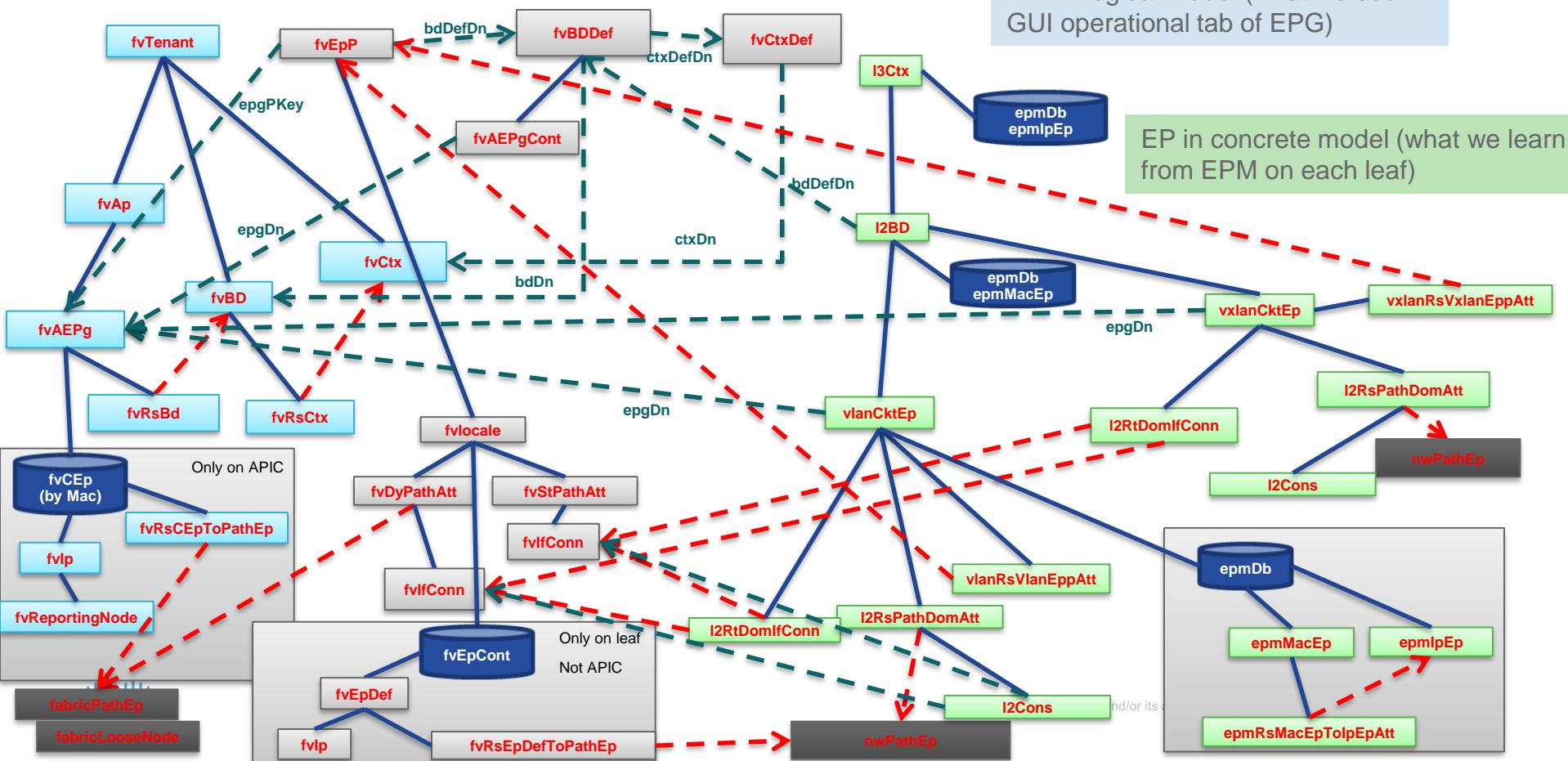
End Point manager EPM to Object Store

Ctx – BD – EPG and EP repo

Logical

Resolved

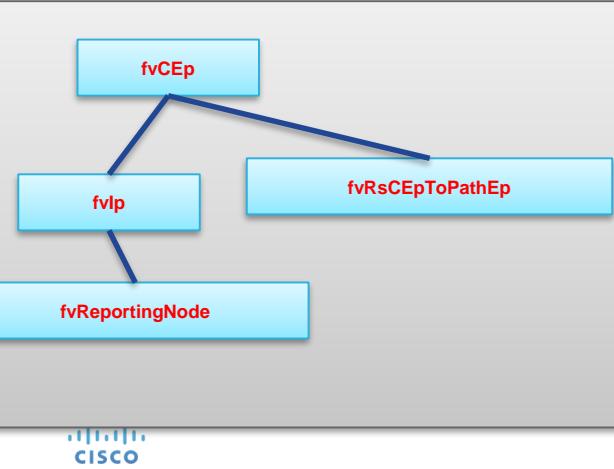
Concrete

End Point DB
In each model

End point in GUI and logical model

what the user will check ...

In tenant – app – EPG
Operational view
And click on the EP



CLIENT END POINT - 10.11.1.101/00:2A:6A:B1:91:7C

POLICY HISTORY

PROPERTIES

MAC: 00:2A:6A:B1:91:7C
IP: 10.11.1.101
Compute Life-Cycle: learned
Encap: vlan-2101
Multicast Address: N/A
VM Name:
VM Operational State:
VM GUID:
Hypervisor Name:
Hypervisor Operational State:
Hypervisor GUID:
vNIC Name:
vNIC IP Address:
vNIC MAC:
vNIC Operational State:

IP Addresses: — IP ADDRESS
10.11.1.101

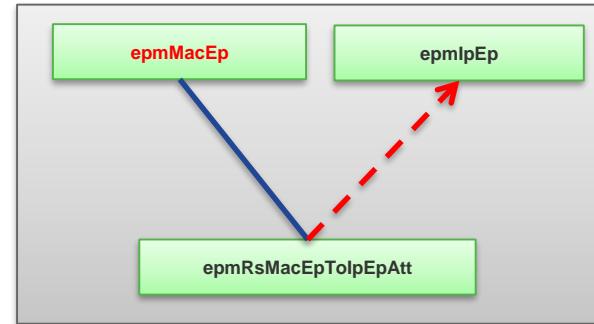
Interfaces: — PATH
Node-101/eth1/33

PAGE 1 OF 1 OBJECTS PER PAGE: 15 DISPLAYING OBJECTS 1-1 OF 1

SUBMIT CLOSE

This screenshot shows the "Properties" tab for a client endpoint in a network management interface. The endpoint's MAC address is 00:2A:6A:B1:91:7C and its IP address is 10.11.1.101. The "Compute Life-Cycle" is listed as "learned". The "Encap" value is "vlan-2101". The "Multicast Address" is marked as "N/A". Below these, fields for "VM Name", "VM Operational State", "VM GUID", "Hypervisor Name", "Hypervisor Operational State", "Hypervisor GUID", "vNIC Name", "vNIC IP Address", "vNIC MAC", and "vNIC Operational State" are present but empty. A table titled "IP Addresses" lists the single IP address 10.11.1.101. Another table titled "Interfaces" lists a single interface path: Node-101/eth1/33. Navigation controls at the bottom indicate this is page 1 of 1, with 15 objects per page, and 1 object displayed. Buttons for "SUBMIT" and "CLOSE" are visible at the bottom right.

Using Query from APIC to find location of EP per Mac



- Example 1 – 2 EP in same L2 EPG (mac learned)

```
admin@pod2-apic1:~> moquery -c epmMacEp | egrep "dn.*00:50:56:8A:15:70"

dn      : topology/pod-1/node-104/sys/ctx-[vxlan-2097153]/bd-[vxlan-15335345]/db-ep/mac-00:50:56:8A:15:70
dn      : topology/pod-1/node-103/sys/ctx-[vxlan-2097153]/bd-[vxlan-15335345]/db-ep/mac-00:50:56:8A:15:70
dn      : topology/pod-1/node-102/sys/ctx-[vxlan-2097153]/bd-[vxlan-15335345]/vlan-[vlan-1140]/db-ep/mac-00:50:56:8A:15:70
dn      : topology/pod-1/node-101/sys/ctx-[vxlan-2097153]/bd-[vxlan-15335345]/vlan-[vlan-1140]/db-ep/mac-00:50:56:8A:15:70
```

Leaf id where
EPM knows it

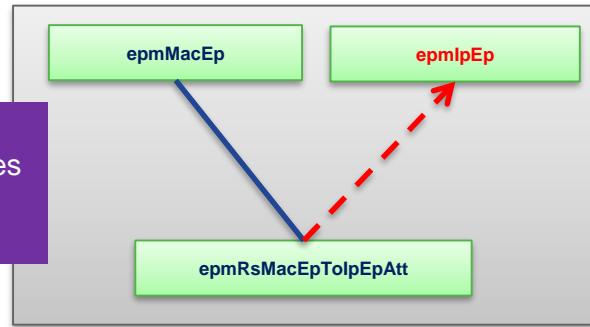
Context id and
Bd id where the EP
Are (all same BD/CTX here)

Access encaps
(vlan-1140)
Only on leaf where
The EP is local

Analysis of the DN

Conclusion : that Mac is local in vlan 1140 local on leaf 101 and 102
103 and 104 leaf knows it as remote Mac (across fabric)

Using Query from APIC to find location of EP per IP



On leaf 103-104, only context is present
(no BD), BD may not even be on those leaves
That EP was routed on ingress leaf
These are the remote leaves.

- Example 2 – 2 EP in different EPG (IP learned)

```
admin@pod2-apic1:~> moquery -c epmIpEp | egrep "dn.*10.200.1.10"
dn      : topology/pod-1/node-104/sys/ctx-[vxlan-2097153]/db-ep/ip-[10.200.1.10]
dn      : topology/pod-1/node-103/sys/ctx-[vxlan-2097153]/db-ep/ip-[10.200.1.10]
dn      : topology/pod-1/node-102/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-ep/ip-[10.200.1.10]
dn      : topology/pod-1/node-101/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-ep/ip-[10.200.1.10]
```

Leaf id where
EPM knows it

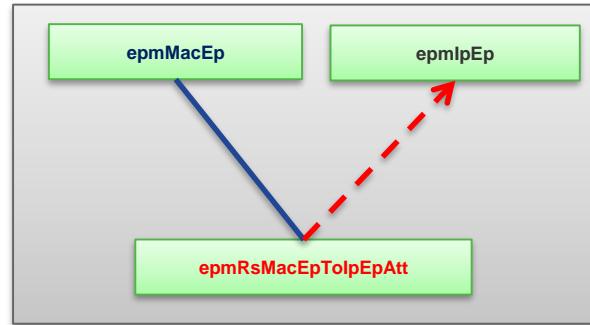
Context id and
Bd id where the EP
Are (all same BD/CTX here)
Only on leaf 101-102

Access encaps
(vlan-1140)
Only on leaf where
The EP is local

Analysis of the DN

Conclusion : that Mac is only known in vlan 1140 local on leaf 101 and 102
Remote leaves learned IP only in the context (vrf)

Using Query from APIC to find location where and EP has IP and Mac bound together



- Example 2 – 2 EP in different EPG (IP learned)

```
admin@pod2-apic1:~> moquery -c epmRsMacEpToIpEpAtt | egrep "dn.*10.200.1.10"
```

```
dn      : topology/pod-1/node-102/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-ep/mac-  
00:50:56:8A:15:70/rsmacEpToIpEpAtt-[topology/pod-1/node-102/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-  
ep/ip-[10.200.1.10]]
```

```
dn      : topology/pod-1/node-101/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-ep/mac-  
00:50:56:8A:15:70/rsmacEpToIpEpAtt-[topology/pod-1/node-101/sys/ctx-[vxlan-2097153]/bd-[vxlan-16383903]/vlan-[vlan-1140]/db-  
ep/ip-[10.200.1.10]]
```

Leaf id where
We have relation IP to mac
Only local leaf here

Mac and IP are part of the dn

Analysis of the DN

Conclusion : that Mac is only known in vlan 1140 local on leaf 101 and 102
Remote leaves learned IP only in the context (vrf)

APIC – show endpoint

```
apic1# show endpoint ip 10.200.2.105
```

Legends:

(P) :Primary VLAN
(S) :Secondary VLAN

Dynamic Endpoints:

Tenant : DC
Application : App
AEPg : EPG2

End Point MAC	IP Address	Node	Interface	Encap	Multicast Address
00:00:00:01:10:02	10.200.2.105	101	eth1/3	vlan-1002	not-applicable

Total Dynamic Endpoints: 1

Total Static Endpoints: 0

EP tracker in GUI

The screenshot shows the Cisco Application Policy Infrastructure Controller (APIC) GUI. At the top, there is a navigation bar with tabs: System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, and Integrations. The Operations tab is selected, and its sub-tabs are: Visibility & Troubleshooting, Capacity Dashboard, EP Tracker (which is highlighted with a red box), and Visualization.

Below the navigation bar, the main content area is titled "EP Tracker". Under this title, there is a section for "End Point Search" with a search input field containing placeholder text: "type or search by MAC, IPv4 or IPv6 address, or VM name e.g., 00:50:56:B0:2E:6C, 10.0.0.1 or 2002:50:22:0:50::1, or comp-vm1". To the right of the search input is a "Search" button.

Further down, there is a section titled "State Transitions" with a table header and some pagination controls. The table header includes columns: Date, IP, MAC, EPG, VRF, Action, Node, Interface, and Encap. Below the table, it says "No Objects Found".

End Point Search

10.200.1.101

Search

Learned At

Tenant

Application

EPG

IP

Pod:1, Leaf:101, Port:eth1/3

DC

App

EPG1

10.200.1.101

Pod:1, Leaf:101, Port:eth1/3

Checking EPM(C)

Leaf - Checking EPM entry which command ?

`Show endpoint [det]`

Basic shows encaps vlan and interface

one (or two) line per EP -> good for grepping

`Show system internal epm endpoint [ip|mac] [x.x.x.x|xxxx.xxxx.xxxx]`

Useful most of the time

shows more detail, internal vlan, ...

`vsh_lc -c' Show system internal epmc endpoint [ip|mac] [x.x.x.x|xxxx.xxxx.xxxx]'`

Most detailed

closest to the hardware

shows learn source, aging information,..

Show endpoint

```
bdsol-aci32-leaf1# show endpoint ip 10.200.1.98 detail
```

Legend:

O - peer-attached	H - vtep	a - locally-aged	S - static
V - vpc-attached	p - peer-aged	L - local	M - span
s - static-arp	B - bounce		

VLAN/ Domain	Encap VLAN	MAC Address IP Address	MAC Info/ IP Info	Interface	Endpoint Group Info
12 DC:DC	vlan-2034 vlan-2034	0050.56a4.0aa1 L 10.200.1.98 L		eth1/9 eth1/9	DC:App:EPG1

Dissecting EPM command

Key of the endpoint: 3 options:

1. MAC only – local and remote EP in layer 2 mode
2. Mac + x IP (0<x< 1024) - only local EP in EPG with unicast routing enable
3. IP only – only remote EP in BD with unicast routing enable

VRF VNID – unique id for a vrf – used in evxlan header for routed packet
Also primary key for security tcam

```
pod2-leaf1# show system internal epm endpoint mac 0050.568a.21ed
```

```
MAC : 0050.568a.21ed :: Num IPs : 1
IP# 0 : 1.1.1.15 :: IP# 0 flags :
Vlan id : 120 :: Vlan vnid : 8904 :: VRF name : DC:span
BD vnid : 16678779 :: VRF vnid : 2949120
Phy If : 0xa02e000 :: Tunnel If : 0
Interface : Ethernet1/47
Flags : 0x80004c04 :: sclass : 16386 :: Ref count : 5
EP Create Timestamp : 09/22/2016 09:07:39.810800
EP Update Timestamp : 09/23/2016 09:59:21.308830
EP Flags : local|IP|MAC|sclass|timer|
::::
```

Vlan Id – PI internal vlan where the EP is learned with its VNID
typically the FD_VLAN (internal value for encapsulation vlan in epg)

BD_VNID = VNID for BD parent of the EPG
Used in eVxlan header for L2 packet in the fabric

If information, learned interface source, either a tunnel (from fabric
from AVS) or a phy interface for locally vlan learned EP

Sclass – the unique EPG identifier in the vrf (aka same sclass can be
reused by different epg if they are in different vrf
Sclass below 16k are Global scope (typically used for Inter-VRF)

Dissecting EPMC command – What more do we have ?

Learn Source can be :

NS : learned by ASIC northstar

EPM : learned from EPM – likely a epsync from vpc peer

Aging line –

Time Type – HT (Host Tracker) , Age, Bounce or Hold

Timeout-left – count down timer till 0 or refreshed

Hit-bit – flag set when we see hit bit in the entry set in hw asic , reset when timer is refreshed

Reset-count - how many time that entry was refreshed (without being deleted)

```
pod2-leaf1# vsh_lc -c 'show system internal epmc endpoint mac 0050.568a.5429'
MAC : 0050.568a.5429 :: Num IPs : 0
Vlan id : 135 :: Vlan vnid : 9306112 :: BD vnid : 16383903
Encap vlan : VXLAN/9306112
VRF name : DC:DC :: VRF vnid : 2097153
phy if : 0 :: tunnel if : 0x18010007 :: Interface : Tunnel7
Flags : 0x80004815
Ref count : 4 :: sclass : 32770
Timestamp : 01/22/1970 22:27:20.960287
last mv timestamp 01/01/1970 00:00:00.000000 :: ep move count : 0
last loop_detection_ts 01/01/1970 00:00:00.000000
previous if : 0 :: loop detection count : 0
Learn Src: EPM
EP Flags : local|vPC|locally-aged|MAC|sclass|timer|
Aging: Timer-type : HT :: Timeout-left : 619 :: Hit-bit : No :: Timer-reset count : 0

PD handles:
Bcm 12 hit-bit : No
[L2]: Asic : NS :: ADJ : 0x13 :: LST SA : 0xd55 :: LST DA : 0xd55 :: GST ING : 0x8ee :: BCM : No
<detail> SDB Data:
    mod 4 :: port 7
```

Hw info pointer to various hardware table in broadcom or northstar

VM moved from one Leaf to another leaf (original leaf shown) – EPMC bounce Entry

BEFORE

```
module-1# show system internal epmc endpoint ip 2.2.2.200
  MAC : 0050.56bf.1082 :: Num IPs : 1
  IP# 0 : 2.2.2.200 :::: IP# 0 flags :
  Vlan id : 15 :::: Vlan vnid : 8292 :::: BD vnid : 16318375
  Encap vlan : 802.1Q/2900
  VRF name : aobst:Network1 :::: VRF vnid : 2097153
  phy if : 0x1a003000 :::: tunnel if : 0 :::: Interface :
Ethernet1/4
  Flags : 0x80004c04
  Ref count : 5 :::: sclass : 16389
  Timestamp : 01/01/1970 10:42:25.275539
  last mv timestamp 01/01/1970 10:00:00.000000 :::: ep move
count : 0
  last loop_detection_ts 01/01/1970 10:00:00.000000
  previous if : 0 :::: loop detection count : 0
  EP Flags : local,IP,MAC,class-set,timer,
  Aging:Timer-type : Host-tracker timeout :::: Timeout-left :
72 :::: Hit-bit : Yes :::: Timer-reset count : 3
...
...
```

AFTER

```
module-1# show system internal epmc endpoint ip 2.2.2.200
  MAC : 0000.0000.0000 :: Num IPs : 1
  IP# 0 : 2.2.2.200 :::: IP# 0 flags :
  Vlan id : 0 :::: Vlan vnid : 0 :::: BD vnid : 0
  VRF name : aobst:Network1 :::: VRF vnid : 2097153
  phy if : 0 :::: tunnel if : 0x18010005 :::: Interface :
Tunnel5
  Flags : 0x80004408
  Ref count : 3 :::: sclass : 16389
  Timestamp : 01/01/1970 10:44:59.504552
  last mv timestamp 01/01/1970 10:00:00.000000 :::: ep move
count : 0
  last loop_detection_ts 01/01/1970 10:00:00.000000
  previous if : 0x18010005 :::: loop detection count : 0
  EP Flags : bounce, IP,class-
set,timer,
  Aging:Timer-type : Bounce timeout :::: Timeout-left : 604
  :::: Hit-bit : No :::: Timer-reset count : 0
```

Vm was moved from local port 1/4 to a remote leaf (port didn't went down)
Entry was updated by COOP to be Bounced to redirect to remote leaf (tunnel 5)
For bounce timer duration

Epm move and summary

```
LEAF# vsh_lc -c "show system internal epmc counters move"
```

MOVE counters

```
L2L upd : 809
L2P upd : 0
L2R upd : 2
P2L upd : 49
P2P upd : 0
P2R upd : 0
R2L upd : 2
R2P upd : 0
R2R upd : 1259
EP non-static to static upd : 0
EP static to non-static upd : 0
PL to VL move : 0
VL to PL move : 0
Cache EP L2R/L2P move : 0
Cache EP P2L/R2L move : 6
EP Force del : 407066
Bounce upd : 208
Sclass upd : 5404
IP only moves : 33707
number of mem port learns overridden : 0
IP R2L moves : 1
BD learn disable due to move : 56
BD learn disable due to loop : 0
BD learn disable : 56
BD learn enable : 55
```

```
Leaf# show system internal epm endpoint all summary
```

EPM Endpoint Summary

Total number of local endpoints	: 1323
Total number of local MACs	: 1323
Total number of local IPv4 addresses	: 803
Total number of local IPv6 addresses	: 0
Total number of non-vPC endpoints	: 90
Total number of vPC endpoints	: 1233
Total number of PL endpoints	: 737
Total number of VL endpoints	: 586
Total number of non-vPC on-peer endpoints	: 110
Total number of remote endpoints	: 2966
Total number of remote MACs	: 2512
Total number of remote IPv4 addresses	: 454
Total number of remote IPv6 addresses	: 0
Total number of VTEPs	: 22
Total number of loopback endpoints	: 7
Total number of SVI endpoints	: 74
Total number of static endpoints	: 4
Total number of config endpoints	: 103
Total number of cached endpoints	: 0
Total number of MACs	: 3945
Total number of IPs	: 1361

End point Troubleshooting methodology

- Start by **moquery** or **show endpoint** on the APIC (epmMacEp, epmIpEp) to find if and where the end point is known (do it few times to see if it is stable)
 - Alternative (better) use EP tracker (embedded or enh EP tracker App)
- Jump to the leaf, **use « show system internal epm endpoint [ip | mac] »** few times to see if it is known and stable
- Get epm(c) traces and look for any of the symptoms in the few previous slides.
- Do not forget to save all epm files and eventually tech-support

EPM log Troubleshooting – Before 4.0

EPM and EPMC trace – Before 4.0

EPM and EPMC (epm Client) keeps very complete log of all EP activity.

In a production fabric those logs may wrap very quickly (we keep around 1 gig of trace for each in rotating log)

Location of those logs on each leaf :

`/var/sysmgr/tmp_logs/epm*`

`/var/log/dme/oldlog/epm*`

Best practice is to copy and store those file as quick as possible after an event you want to analyse

```
mkdir /tmp/epm-log-20150908
```

```
cp /var/log/dme/oldlog/epm* /tmp/epm-log-20150908
```

Big part of troubleshooting is looking at those logs

You can get more verbose log (but filling faster) by enabling :

```
pod2-leaf1# debug system internal epm trace detail
setting debug level to EPM_DEBUG_LEVEL_TRACE_DETAIL
pod2-leaf1# vsh_lc -c 'debug system internal epmc trace detail'
vsh_lc -c "debug system internal epmc trace detail"
setind debug level to EPMC_DEBUG_LEVEL_TRACE_DETAIL
```

EPMC trace – new EP learn (ns learn)

```
[2015 Sep  9 11:03:24.935395795:8200:epmc_handle_ns_learn_notif:4177:t] Got ns learn for 1 entries
Notify: valid=1 key-type=L3 V4 oidx=0 seg_vrf=0x12
          addr64_0:1=0x1010164:0x0 mac=0x5056bf03c2 mode=0 <<< source IP and mac detected by NS
          smod/sport=0x0/0x18 vp=0x0 port=0x0
          sclass=0x8006 nd_ow=0 learn_from=0
          prio=0 chg=0 spare=0 source=1 location=0
          rsvd_0/1/2/3=0/0/0/0
```

Epmc signature match :
Epmc_handle_ns_learn_notif
Means we got notif from Hw

```
[2015 Sep  9 11:03:24.935422602:8202:epmc_copy_learn_notif_to_ep_req:3537:t] Pl host
EPM EP req :: EP_OP=ADD VLAN 16 MAC:=0050.56bf.03c2 #IPs=1
IP[0] = 1.1.1.100, IP flags :
EPG/BD/VRF=8358/16318375/2097153 IfIndex=0x1a003000 TunIfIndex=0x0 SCLASS=32774
Interface name : Ethernet1/4
EP Flags: local,IP,MAC,class-set,timer,
Timestamp: 01/02/1970 08:15:00.396871
```

Epmc signature match :
Epmc_copy_learn_notif_to_ep_req
Means we update ASIC table

EPMC trace – EPMC to EPM

```
[2015 Sep  9 11:03:24.935544529:8232:epmc_prog_ep_create:2313:t] New EP:  
  
EPM EP entry :: VLAN 16 MAC:=0050.56bf.03c2 #IPs=1  
    IP[0] = 1.1.1.100, IP flags :  
    EPG/BD/VRF=8358/16318375/2097153 IfIndex=0x1a003000 TunIfIndex=0x0 Ref=5 SCLASS=32774  
    Interface name : Ethernet1/4  
EP Flags: local,IP,MAC,class-set,timer,  
Timestamp: 01/02/1970 08:15:00.396871  
Aging:  
    Timer-type: Host-tracker timeout, Timeout-left: 120  
[L2]: Asic : NS :::: ADJ : 0xf :::: BCM : Yes  
[L3-0]: Asic : NS :::: ADJ : 0xf :::: BCM : Yes
```

epmc_prog_ep_create - MAC+IP

"New EP:" - Add Endpoint (with Mac Details) to EndPoint database

This get pushed to EPM

Note this only occurs for a "new" endpoint, a refreshed endpoint will not see this step

EPM trace – getting notif from EPMC

```
[2015 Dec 10 01:17:44.194168309:170477:epm_handle_epmc_ep_req_rsp_msg:4952:t] Processing 1 EPs
[2015 Dec 10 01:17:44.194177478:170478:epm_debug_dump_epm_ep_req:347:t]
EP req - SAP 376
EP_OP ADD MAC: = aaaa.0101.0101, #IPs 1
IP : 1.1.101.101
VLAN : 24 EPG vnid : 10092 BD vnid : 16547723 VRF vnid : 3014656
ifidx : 0x1a007000 TEP ip/tun if idx : 0 sclass : 16388
Flags : local,IP,MAC,sclass,timer,
EP Req TS : 01/03/1970 10:59:15.757284
[2015 Dec 10 01:17:44.194514113:170483:epm_ep_notify_coop:1416:t] rec_ts: 1449728276:350604947
[2015 Dec 10 01:17:44.195008853:170484:epm_add_or_mod_mac_ep_obj:1764:t] Add of MAC obj with Dn sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep/mac-AA:AA:01:01:01:01 under sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep
[2015 Dec 10 01:17:44.195086729:170485:epm_add_or_mod_ip_ep_obj:1590:t] Add of IpEp obj with Dn sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep/ip-[1.1.101.101]
[2015 Dec 10 01:17:44.195158280:170486:epm_add_or_mod_ip_ep_obj:1604:t] Mac-to-IP relation obj added to objstore with Dn sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep/mac-AA:AA:01:01:01:01:rsmacEpToIpEpAtt-[sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep/ip-[1.1.101.101]]
[2015 Dec 10 01:17:44.195163570:170488:epm_debug_dump_epm_ep:385:t]
EP entry
MAC: = aaaa.0101.0101, #IPs 1
IP : 1.1.101.101
VLAN : 24 EPG vnid : 10092 BD vnid : 16547723 VRF vnid : 3014656
ifidx : 0x1a007000 tun if idx : 0 vtep tun if idx : 0
sclass : 16388 ref cnt : 5
Flags : local,IP,MAC,sclass,timer,
Create TS : 12/10/2015 01:17:56.349284
Upd TS : 12/10/2015 01:17:56.349284

[2015 Dec 10 01:17:44.195181109:170493:epm_send_ep_req_msg:1059:t] Sending 1 EPs to MTS_SAP_COOP
[2015 Dec 10 01:17:44.195258104:170494:epm_mcecm_tl_debug_log_func:63:t] Sent 532 bytes

[2015 Dec 10 01:17:44.195260584:170495:epm_walk_dirty_vlan_list:3521:t] Updating vlan 24 EP db with version 140
[2015 Dec 10 01:17:44.195323926:170496:epm_add_ckt_objs:2839:t] Modified EpDb obj sys/ctx-[vxlan-3014656]/bd-[vxlan-16547723]/vlan-[vlan-101]/db-ep
```

Epmc_handle_epmc_ep_req: process new learn from Epmc

Epm_ep_notify_coop: update coop with new EP

Epm_add_or_mod_mac_ep_obj: EPM publish to object store

ACI forwarding – Summary Recap

Forwarding behavior Layer 2 only

ARP resolution – Layer 2 BD – Unicast routing disable

How do we resolved ARP from one EP to another EP assuming both are in the same EPG and the BD has no unicast routing enable

- Hardware-proxy or Flood do not make difference
- **Broadcast ARP** request are always **flooded** (in BD aka **GIPo Multicast** group of BD)
 - Even if ARP flooding shows disable (indeed with arp flooding disable we should use target IP to try proxy but we do not store ip with unicast routing disable)
- **Unicast ARP** request are **unicast** to remote Leaf TEP (if Dmac of the ARP is known, otherwise flooded in flood mode or send to Proxy in Hw-proxy mode)

Layer 2 unicast same EPG EP known on ingress and egress Leaf

Leaf1/2 knows VM a as local and VM B as remote

Leaf 4 knows VM B as local and VM A as remote

A → B : leaf1/2 encaps with

src = VTEP VPC

Dst = PTEP Leaf4

VNID = BD VNID

Routing in fabric (ECMP)

Decaps in Leaf 4

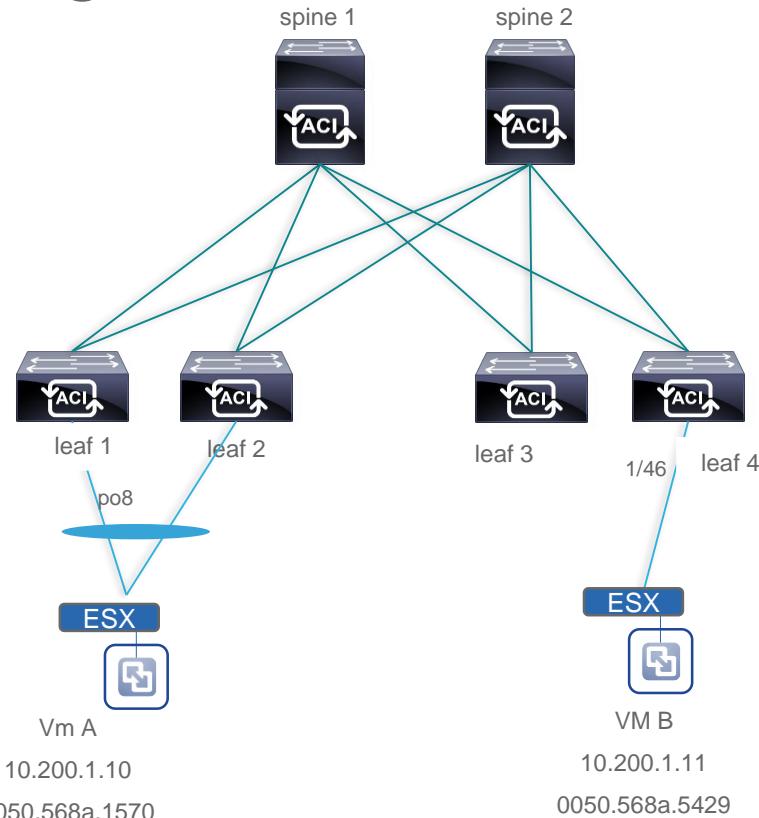
Leaf 4 does mac lookup and send out to the correct port

B → A : leaf 4 encaps with

Src = leaf 4 PTEP

Dst = vpc VTEP of leaf1/2

VNID of BD



Example 1 : 2 VM in same EPG

BD is L2 (no unicast routing)

Traffic is flowing between A and B – Leaf 1 Output

LEAF 1

```
=====
pod2-leaf1# show system internal epm endpoint mac 00:50:56:8A:15:70
MAC : 0050.568a.1570 :::: Num IPs : 0
Vlan id : 59 :::: Vlan vnid : 9032 :::: VRF name : DC:DC
BD vnid : 15335345 :::: VRF vnid : 2097153
Phy If : 0x16000007 :::: Tunnel If : 0
Interface : port-channel18
Flags : 0x80004805 :::: sclass : 32770 :::: Ref count : 4
EP Create Timestamp : 12/22/2015 11:45:13.440144
EP Update Timestamp : 12/22/2015 12:13:31.091614
EP Flags : local,vPC,MAC,class-set,timer,
:::
```

```
pod2-leaf1# show system internal epm endpoint mac 00:50:56:8A:54:29
MAC : 0050.568a.5429 :::: Num IPs : 0
Vlan id : 57 :::: Vlan vnid : 15335345 :::: VRF name : DC:DC
BD vnid : 15335345 :::: VRF vnid : 2097153
Phy If : 0 :::: Tunnel If : 0x18010006
Interface : Tunnel16
```

```
Flags : 0x80004820 :::: sclass : 32770 :::: Ref count : 4
EP Create Timestamp : 12/22/2015 12:01:35.978385
EP Update Timestamp : 12/22/2015 12:17:15.519222
EP Flags : peer-aged,MAC,class-set,timer,
:::
```

```
pod2-leaf1# show interface tunnel 6
Tunnel6 is up
```

```
MTU 9000 bytes, BW 0 Kbit
Transport protocol is in VRF "overlay-1"
Tunnel protocol/transport is ivxlan
Tunnel source 10.0.168.95/32 (lo0)
Tunnel destination 10.0.168.90
```

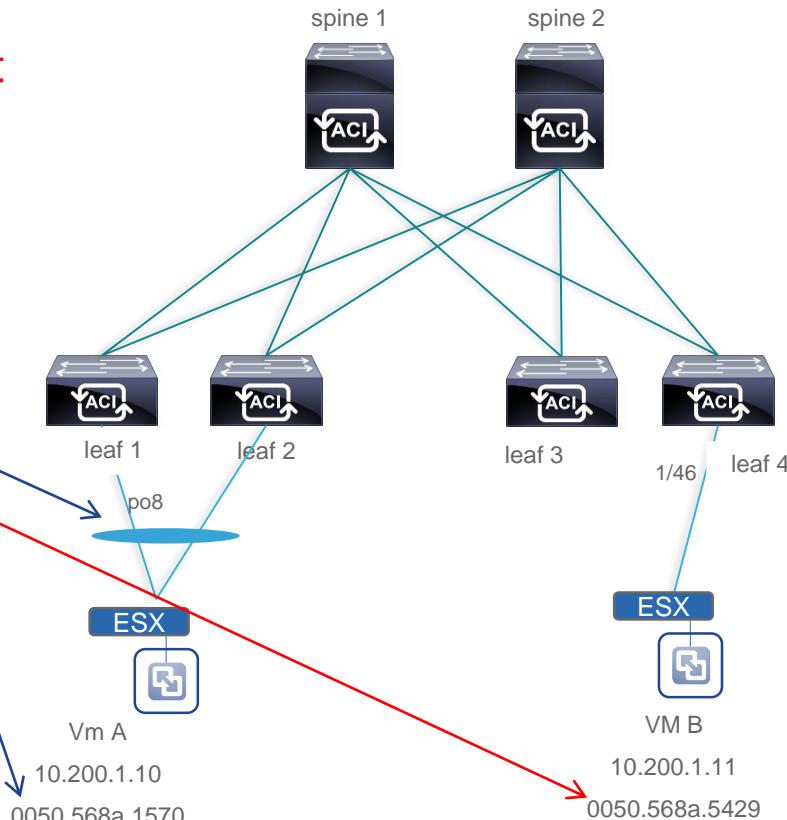
```
...
pod2-leaf1# show ip route vrf overlay-1 10.0.168.90
```

```
10.0.168.90/32, ubest/mbest: 2/0
*via 10.0.168.92, eth1/49.1, [115/3], 6d23h, isis-isis_infra, L1
*via 10.0.168.94, eth1/50.2, [115/3], 6d23h, isis-isis_infra, L1
```

Local MAC is learned
On Po8

Remote Mac is known on Tun 6

Tunnel 6 is ivxlan tunnel to 10.0.169.95
ISIS routes it to the spine



10.0.168.90 is indeed top address of leaf 4

```
pod2-leaf4# acidiag fnvread | egrep "10.0.168.90"
```

104	pod2-leaf4	SAL1818RP59	10.0.168.90/32	leaf	active	0
-----	------------	-------------	----------------	------	--------	---

Example 1 : 2 VM in same EPG

BD is L2 (no unicast routing)

Traffic is flowing between A and B - leaf 4 output

```
LEAF 4
=====
pod2-leaf4# show system internal epm endpoint mac 00:50:56:8A:15:70
MAC : 0050.568a.1570 :: Num IPs : 0
Vlan id : 37 :: Vlan vnid : 15335345 :: VRF name : DC:DC
BD vnid : 15335345 :: VRF vnid : 2097153
Phy If : 0 :: Tunnel If : 0x18010002
Interface : Tunnel2
Flags : 0x80004800 :: sclass : 32770 :: Ref count : 4
EP Create Timestamp : 12/22/2015 11:45:13.438616
EP Update Timestamp : 12/22/2015 12:38:39.007656
EP Flags : MAC,class-set,timer,
```

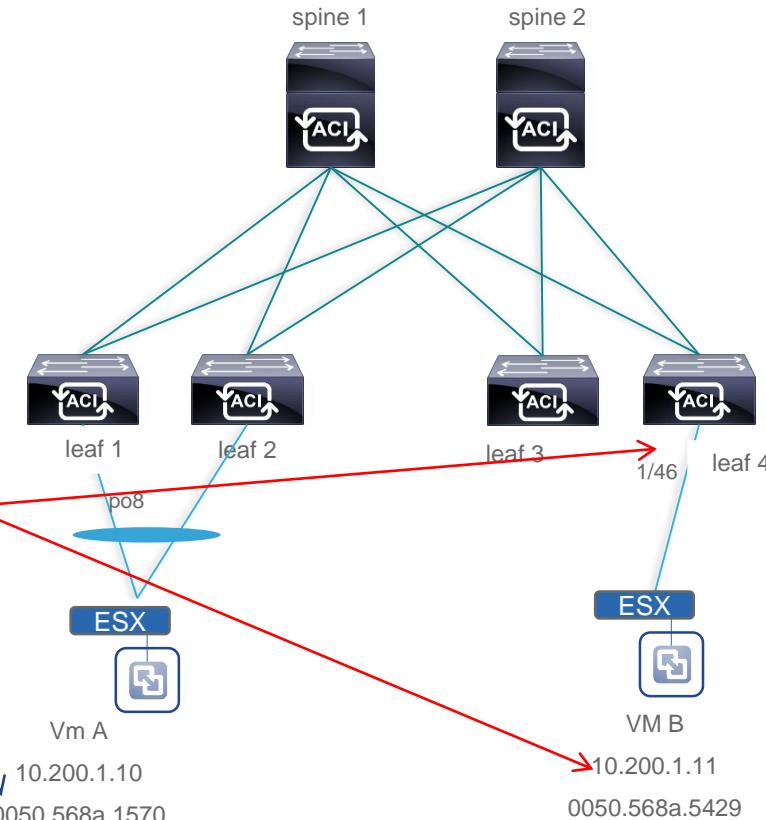
```
pod2-leaf4# show system internal epm endpoint mac 00:50:56:8A:54:29
MAC : 0050.568a.5429 :::: Num IPs : 0
Vlan id : 38 :::: Vlan vnid : 9032 :::: VRF name : DC:DC
BD vnid : 15335345 :::: VRF vnid : 2097153
Phy If : 0x1a02d000 :::: Tunnel If : 0
Interface : Ethernet1/46
Flags : 0x80004804 :::: sclass : 32770 :::: Ref count : 4
EP Create Timestamp : 12/22/2015 11:51:20.530855
EP Update Timestamp : 12/22/2015 12:41:11.608699
EP Flags : local,MAC,class-set,timer,
```

local Mac is known

```
pod2-leaf4#  
pod2-leaf4# show interface tunnel 2  
Tunnel2 is up  
    MTU 9000 bytes, BW 0 Kbit  
    Transport protocol is in VRF "overlay-1"  
    Tunnel protocol/transport is ixvxlan  
    Tunnel source 10.0.168.90/32 (lo0)  
    Tunnel destination 10.0.168.97
```

Tunnel 2 is ivxlan tunnel to 10.0.168.97
ISIS routes it to the spine

```
10.0.168.97 is indeed vpc VIP address of leaf 1-2 pair  
pod2-leaf1# show system internal epm vpc | egrep "10.0.168.97"  
vPC VIP : 10.0.168.97
```



Layer 2 unicast same EPG EP unknown on ingress and egress Leaf

Leaf1/2 do not know VM V as remote
Leaf 4 do not know VM B as local and VM A
Spine do not have VM B in their Repo (DB)

VM A have ARP resolved for VM B

It depends from BD unknown unicast settings

1/ BD in Hw proxy

A→B : leaf1/2 encaps with
src = VTEP VPC

Dst = anycast MAC of spine

VNID = BD VNID

Spine does not know it and **drops frame**

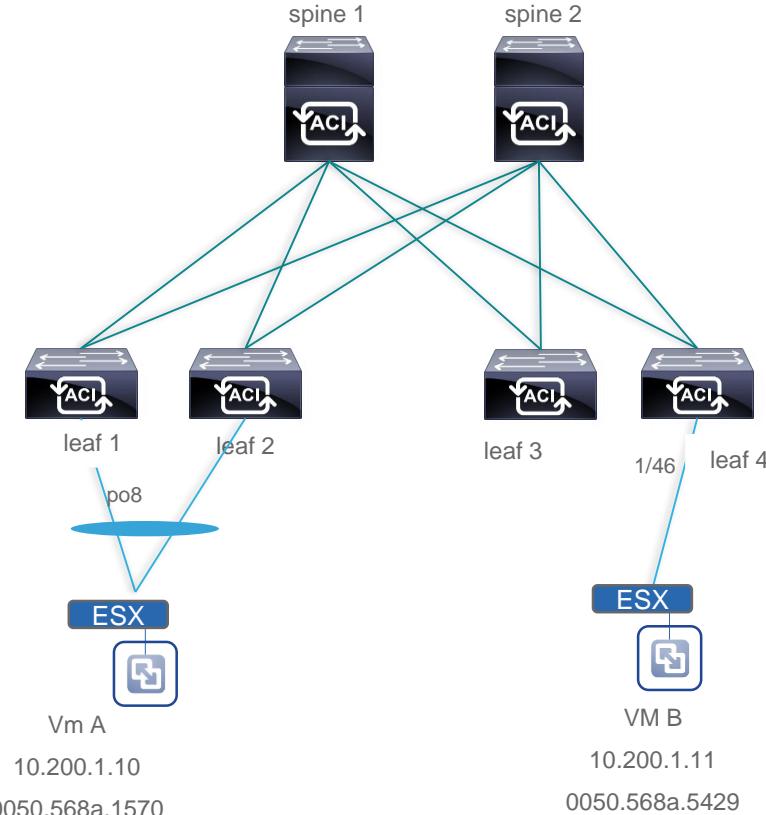
2/ BD in flood mode

A→B : Leaf 1 or 2 encaps with

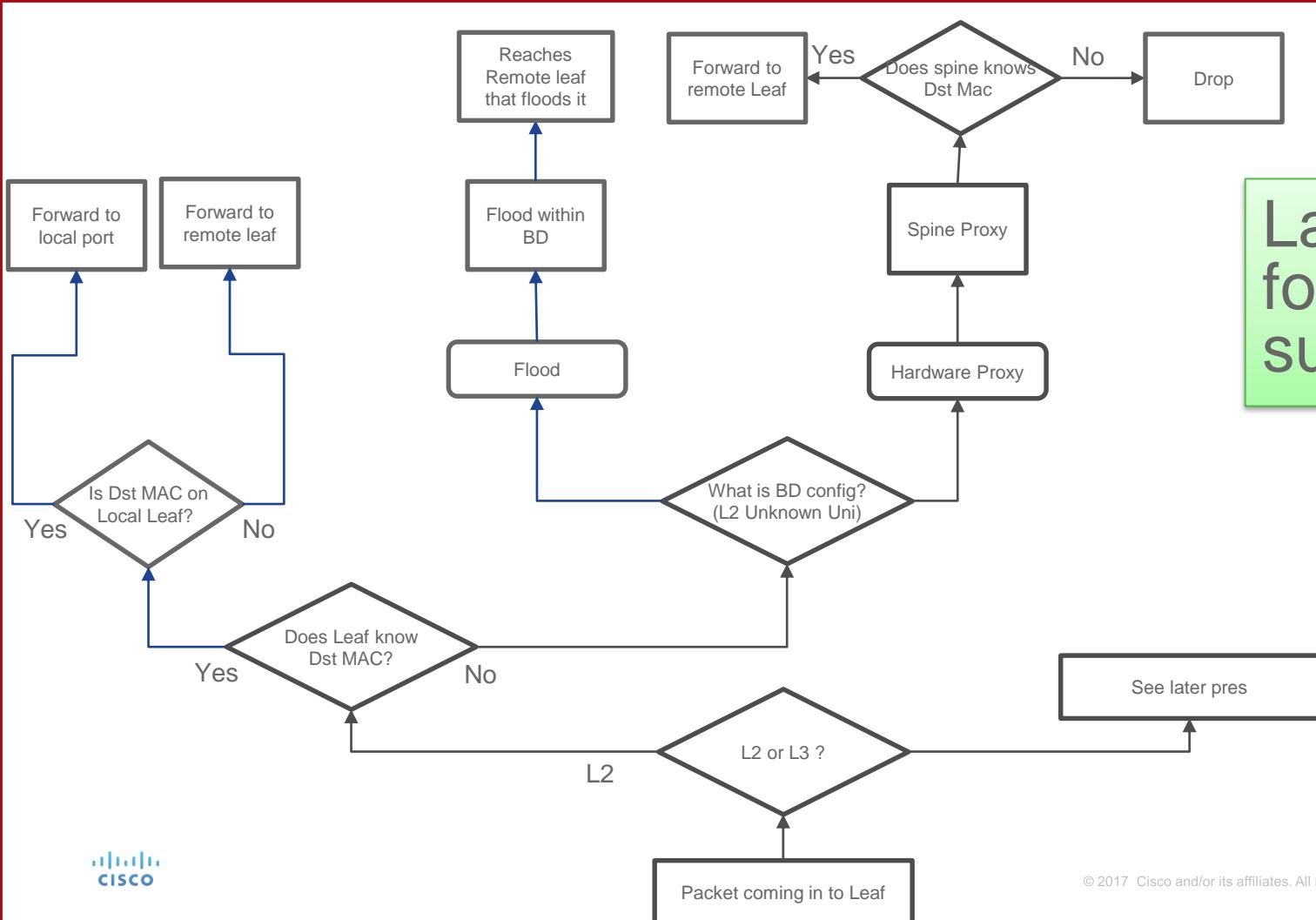
Src = vtep VPC

Dst = BD GIPo

Frame is flooded in the epg



Layer 2 forwarding summary



Forwarding behavior Layer 3

ARP resolution – BD with unicast routing enable and subnet under BD

Pervasive gateway (usually default gateway ARP resolution):

- Pervasive gateway and subnet should be visible on each leaf with endpoint attached
- ARP will be replied by the directly connected leaf (or one of the two in a vpc pair)

Note : no ARP table is maintained as such for EP part of the fabric, you Should check EPM entry instead

ARP pervasive gateway debugging

Tcpdump on kpm_inb
Note that for ARP only ARP Rx
By CPU will be seen there

```
pod2-leaf2# tcpdump -xxvvi kpm_inb arp
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes

14:34:03.289865 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 10.200.1.1 tell 10.200.1.16, length
46
    0x0000: ffff ffff ffff 0050 568a 5429 0806 0001
    0x0010: 0800 0604 0001 0050 568a 5429 0ac8 0110
    0x0020: 0000 0000 0000 0ac8 0101 0000 0000 0000
    0x0030: 0000 0000 0000 0000 0000 0000 0000 0000
```

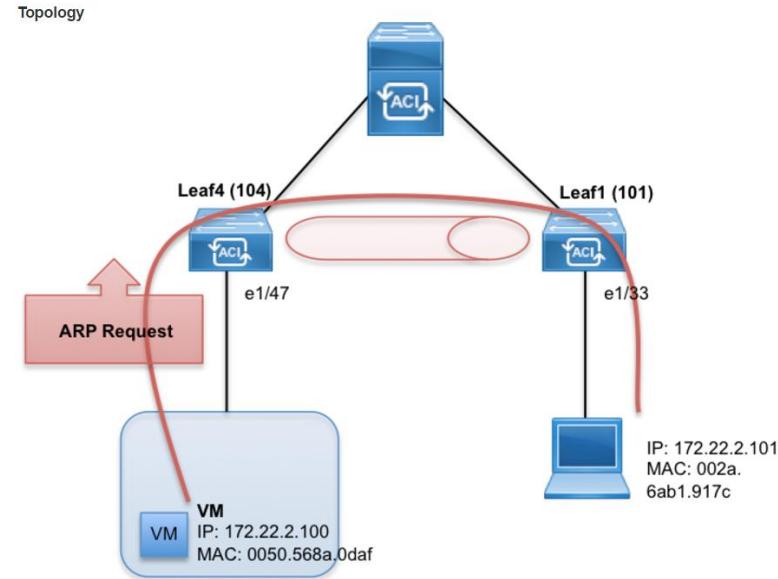
Arp process traces

```
Pod2-leaf2# show ip arp internal eve ev | egrep -B 1 "10.200.1.16"
10) Event:E_DEBUG_DSF, length:181, at 290447 usecs after Fri Sep 23 14:34:03 2016
    [116] TID 9842:arp_process_receive_packet_msg:7186: log_collect_arp_pkt; sip = 10.200.1.16; dip =
10.200.1.1;interface = Vlan159; phy_interface = Tunnel13;Info = Received arp request
11) Event:E_DEBUG_DSF, length:145, at 290271 usecs after Fri Sep 23 14:34:03 2016
    [116] TID 9842:arp_update_epm_payload:7447: Updating epm ifidx: 1801000d vlan: 162 ip: 10.200.1.16,
ifMode: 128is_garp: 0, mac: 0 80 86 138 84 41
12) Event:E_DEBUG_DSF, length:159, at 290241 usecs after Fri Sep 23 14:34:03 2016
    [116] TID 9842:arp_process_receive_packet_msg:7100: log_collect_arp_pkt; sip = 10.200.1.16; dip =
10.200.1.1;interface = Vlan159; Info = DIP local on interface.
13) Event:E_DEBUG_DSF, length:156, at 290237 usecs after Fri Sep 23 14:34:03 2016
    [116] TID 9842:arp_process_receive_packet_msg:6943: log_collect_arp_pkt; sip = 10.200.1.16; dip =
10.200.1.1; interface = Vlan159;info = Garp Check adj:(nil)
--
```

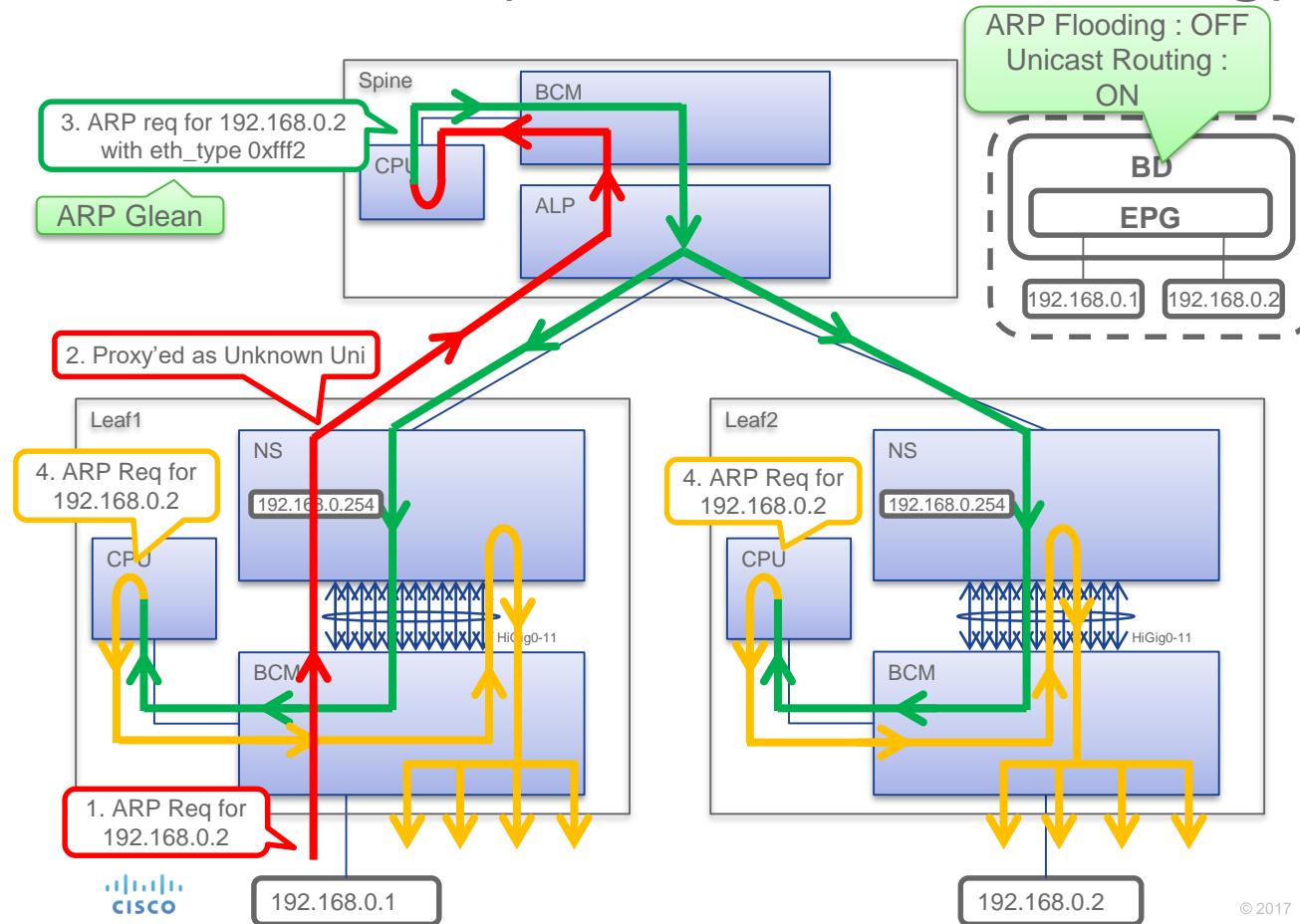
ARP resolution across the fabric in BD with unicast routing.

3 scenarios:

1. if ARP flooding enable (hw-proxy or flood): we always flood all Broadcast ARP request to all path in the BD across all leaves (like L2 case)
 - (Target IP info not used for Fwd, but sender IP learned by EPM)
2. If ARP flooding disable and ingress leaf EPM knows Dest EP (per target IP address) → unicast ARP req (even if Bcast) to egress Leaf TEP. Egress leaf will flood to all local ports in the BD
3. If ARP flooding disable and ingress leaf EPM do not know remote IP -> unicast to anycast spine proxy IP address
 - a) If Spine knows EP in Coop DB → unicast to TEP on egress leaf and egress leaf floods to BD (like scenario 2)
 - b) (arp glean) if Spine does not know the EP in Coop DB ---> drops arp request but generate a new ARP request with sender info being ACI BD pervasive address, this new arp is send to all leaves in the BD and is flooded to all path across all leaves in the BD. Subsequent ARP request for this EP will only go the leaf where the EP is connected.



ARP Glean (Silent Host Tracking)



1. ARP Req is sent out from source Host
2. Target-IP LookUP on NS
=> miss
=> NS forwards it to Spine
3. Spine checks COOP
=> miss
=> ARP Glean are sent out to all Leaf
4. Each Leaf receives it on its CPU
 - I. Check if Target-IP belongs to one of the subnets in the same VRF on the Leaf
 - II. If yes, generate ARP Req for Target-IP
 - III. Floods it out to all ports within BD with that subnet

Note on ARP glean

ARP glean happens when :

- For Routed Packet (IP packets) , when COOP DB on spine do not know remote IP
- For bridged packet only :
 - When we receive ARP packet (only ARP) target to an IP we do not know in COOP DB in our BD subnet
 - ARP glean for bridged packet only occurs if Unicast routing is set and if the target IP is part of our BD subnet

ARP glean do not happen when :

- ARP glean do not happen for bridged IP packet when COOP DB Do not know destination IP

Scenario where traffic black hole can occur (silent host scenario)

- If unicast routing is disable for any bridged packet when Dest MAC is unknown in COOP and we run Hw-Proxy in BD
- In unicast routing is enable and we are in Hw-Proxy mode and
 - We receive ARP packet for a target IP which is not part of our BD subnet
 - **We receive a bridged IP packet (not ARP) for which the target IP is not known in COOP**

This last scenario normally only occurs when we have bridged packet in a routed BD and End Point do have ARP resolved for the destination and as such send IP packet directly to an unknown silent Dest IP. This could happen If we get a STP TCN that flush EP after the ARP was resolved by a VM for example

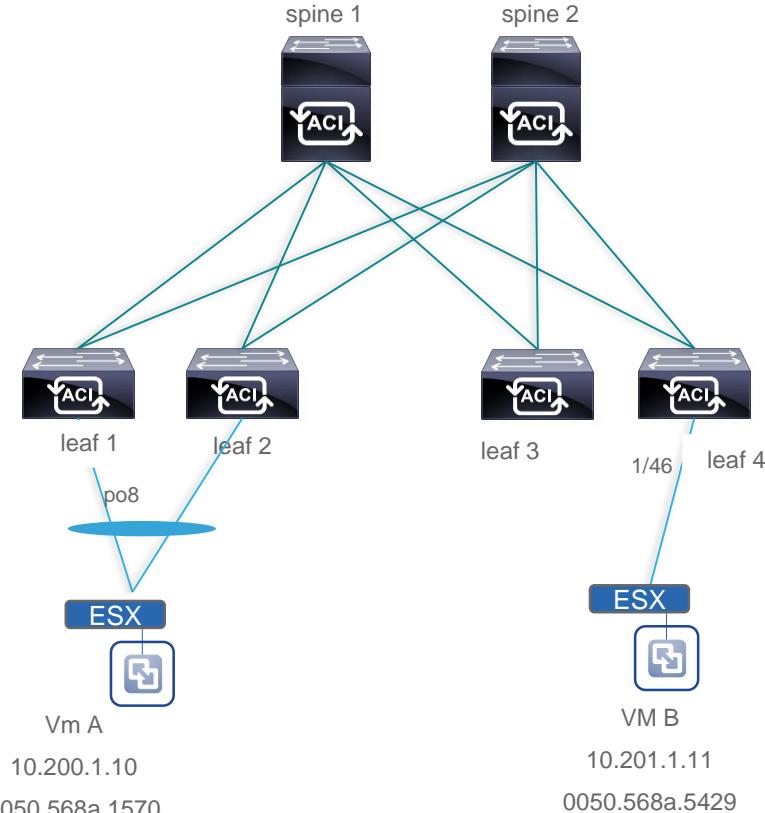
Layer 3 unicast different EPG EP known on ingress and egress Leaf

Leaf1/2 knows VM a as local and VM B as remote
Leaf 4 knows VM B as local and VM A as remote

A → B : leaf1/2 makes the routing (and apply policy as dst epg is known) and then encaps with
src = VTEP VPC
Dst = PTEP Leaf4
VNID = CTX VNID

Routing in fabric (ECMP)
Decaps in Leaf 4
Leaf 4 does IP and mac lookup and send out to the correct port

B → A : leaf 4 encaps with
Src = leaf 4 PTEP
Dst = vpc VTEP of leaf1/2
VNID of CTX



Example 2 : 2 VM in different EPG/BD

Both BD are L3(unicast routing)

Traffic is flowing between A and B – Leaf 1 output

```
LEAF 1
=====
pod2-leaf1# show system internal epm endpoint ip 10.200.1.10
MAC : 0050.568a.1570 :: Num IPs : 1
IP# 0 : 10.200.1.10 :: IP# 0 flags :
Vlan id : 19 :: Vlan vnid : 9032 :: VRF name : DC:DC
BD vnid : 16383903 :: VRF vnid : 2097153
Phy If : 0x16000007 :: Tunnel If : 0
Interface : port-channel18
Flags : 0x80004c05 :: sclass : 32770 :: Ref count : 5
EP Create Timestamp : 12/22/2015 17:26:26.912442
EP Update Timestamp : 12/23/2015 13:28:31.943672
EP Flags : local,vPC,IP,MAC,class-set,timer,
:::
```

```
pod2-leaf1# show system internal epm endpoint ip 10.201.1.11
MAC : 0000.0000.0000 :: Num IPs : 1
IP# 0 : 10.201.1.11 :: IP# 0 flags :
Vlan id : 0 :: Vlan vnid : 0 :: VRF name : DC:DC
BD vnid : 0 :: VRF vnid : 2097153
Phy If : 0 :: Tunnel If : 0x18010006
Interface : Tunnel6
Flags : 0x80004420 :: sclass : 49153 :: Ref count : 3
EP Create Timestamp : 12/23/2015 12:59:47.292658
EP Update Timestamp : 12/23/2015 13:27:48.051138
EP Flags : peer-aged,IP,class-set,timer,
```

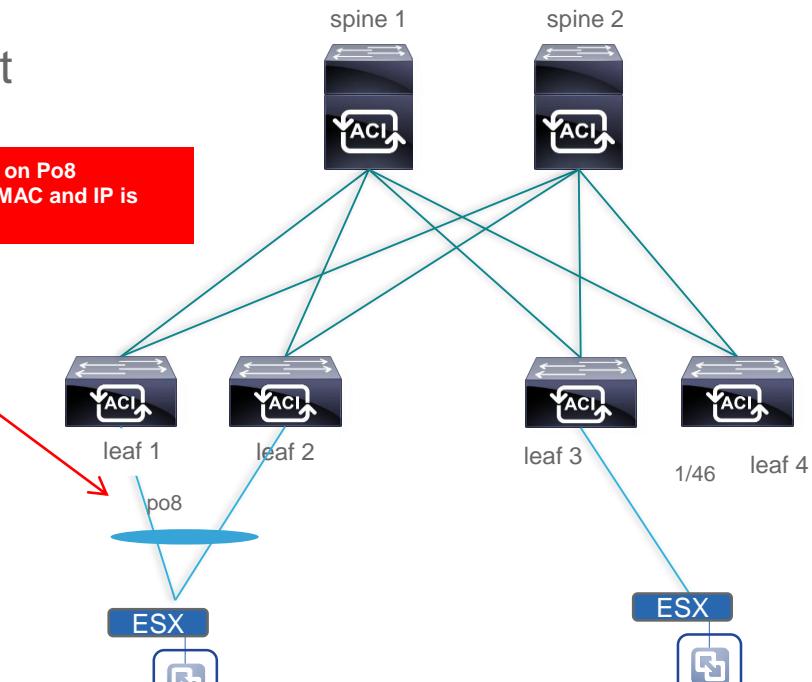
```
pod2-leaf1# show interface tunnel 6
Tunnel6 is up
```

```
MTU 9000 bytes, BW 0 Kbit
Transport protocol is in VRF "overlay-1"
Tunnel protocol/transport is ivxlan
Tunnel source 10.0.168.95/32 (lo0)
Tunnel destination 10.0.168.95
...
```

Local IP is known on Po8
EPM entry key is MAC and IP is bound to it

Remote IP is learned
On Tunnel 6
Note MAC is not there ...

Tunnel 6 is ivxlan tunnel to 10.0.168.95



10.200.1.10
0050.568a.1570

10.201.1.11
0050.568a.5429

10.0.168.90 is indeed top address of leaf 4
pod2-leaf4# acidiag fnvread | egrep "10.0.168.95"
104 pod2-leaf3 SAL1818RP59 10.0.168.95/32 leaf active 0

Example 2 : 2 VM in different EPG/BD

Both BD are L3(unicast routing)

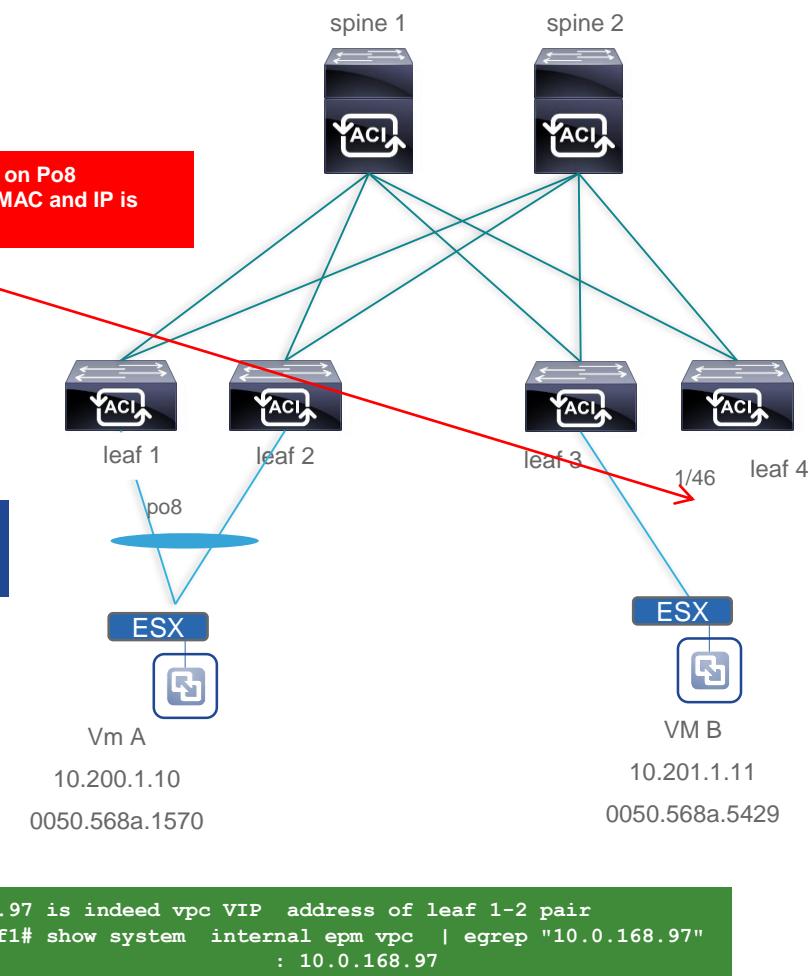
Traffic is flowing between A and B – leaf 3 output

```
LEAF 3
=====
pod2-leaf4# show system internal epm endpoint ip 10.201.1.11
MAC : 0050.568a.5429 :: Num IPs : 1
IP# 0 : 10.201.1.11 :: IP# 0 flags :
Vlan id : 21 :: Vlan vnid : 8967 :: VRF name : DC:DC
BD vnid : 16285611 :: VRF vnid : 2097153
Phy If : 0xa02d000 :: Tunnel If : 0
Interface : Ethernet1/46
Flags : 0x80005c04 :: sclass : 49153 :: Ref count : 5
EP Create Timestamp : 12/23/2015 10:10:38.127495
EP Update Timestamp : 12/23/2015 13:28:39.830406
EP Flags : local,IP,MAC,host-tracked,class-set,timer,
:::
pod2-leaf4# show system internal epm endpoint ip 10.200.1.10
MAC : 0000.0000.0000 :: Num IPs : 1
IP# 0 : 10.200.1.10 :: IP# 0 flags :
Vlan id : 0 :: Vlan vnid : 0 :: VRF name : DC:DC
BD vnid : 0 :: VRF vnid : 2097153
Phy If : 0 :: Tunnel If : 0x18010002
Interface : Tunnel2
Flags : 0x80004420 :: sclass : 32770 :: Ref count : 3
EP Create Timestamp : 12/23/2015 12:59:47.290969
EP Update Timestamp : 12/23/2015 13:28:39.829616
EP Flags : peer-aged,IP,class-set,timer,
:::
pod2-leaf4# show interface tunnel 2
Tunnel2 is up
MTU 9000 bytes, BW 0 Kbit
Transport protocol is in VRF "overlay-1"
Tunnel protocol/transport is ivxlan
Tunnel source 10.0.168.90/32 (lo0)
Tunnel destination 10.0.168.97
```

Local IP is known on Po8
EPM entry key is MAC and IP is bound to it

Remote IP is learned
On Tunnel 2
Note : MAC is not there ...

Tunnel 2 is ivxlan tunnel to 10.0.168.97



Layer 3 unicast different EPG EP unknown on ingress leaf

Leaf1/2 does not know VM

Leaf 4 knows VM B as local and VM A as remote

VM A have an ARP entry for VM B

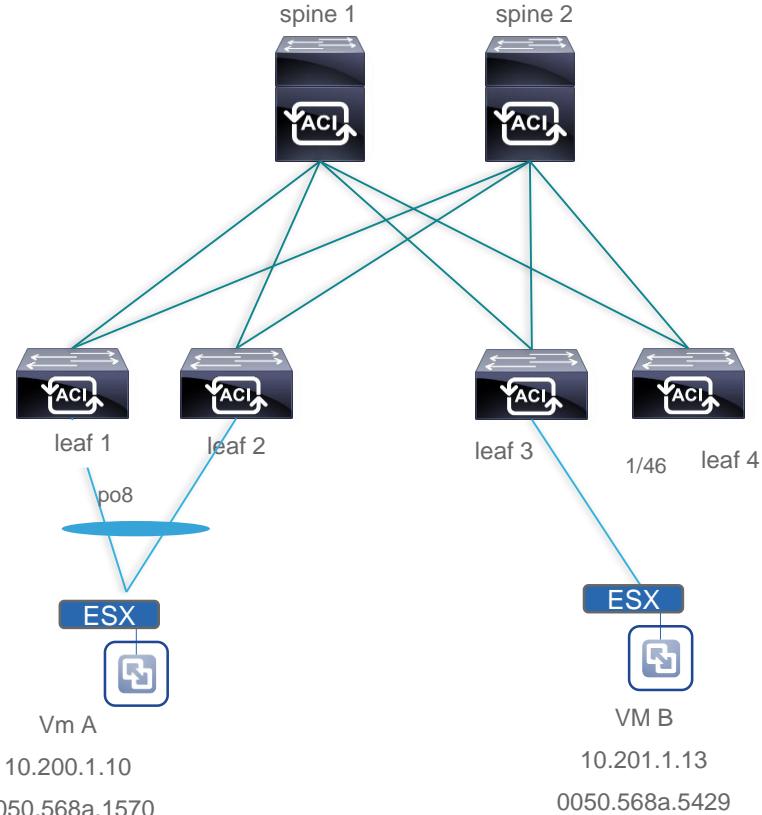
Packet is to be routed (dst mac is Our fabric mac)

Ingress leaf will use routing table as no /32 epm entry is there. This will redirect to anycast-v4 on spine

If Spine knows Remote EP in COOP it will then unicast (inline rw of outer header) to Destination leaf 3 PTEP

If spine DOES NOT know Remote EP in COOP, it will attempt to resolve arp with ARP Glean.

Note that BD mode for unk ucast do not matter (hw-proxy or flood)



Layer 3 unicast same EPG EP unknown on ingress leaf –

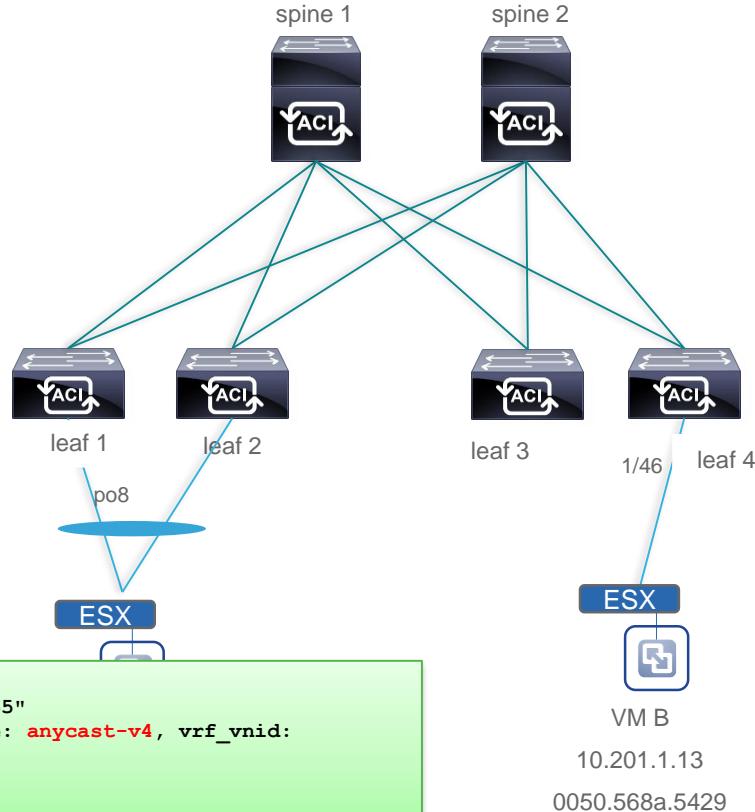
We will send from ingress leaf to anycast spine proxy-v4
Whatever the BD mode is (here is is routed frame)

```
pod2-leaf1# show system internal epm endpoint ip 10.201.1.13

pod2-leaf1#

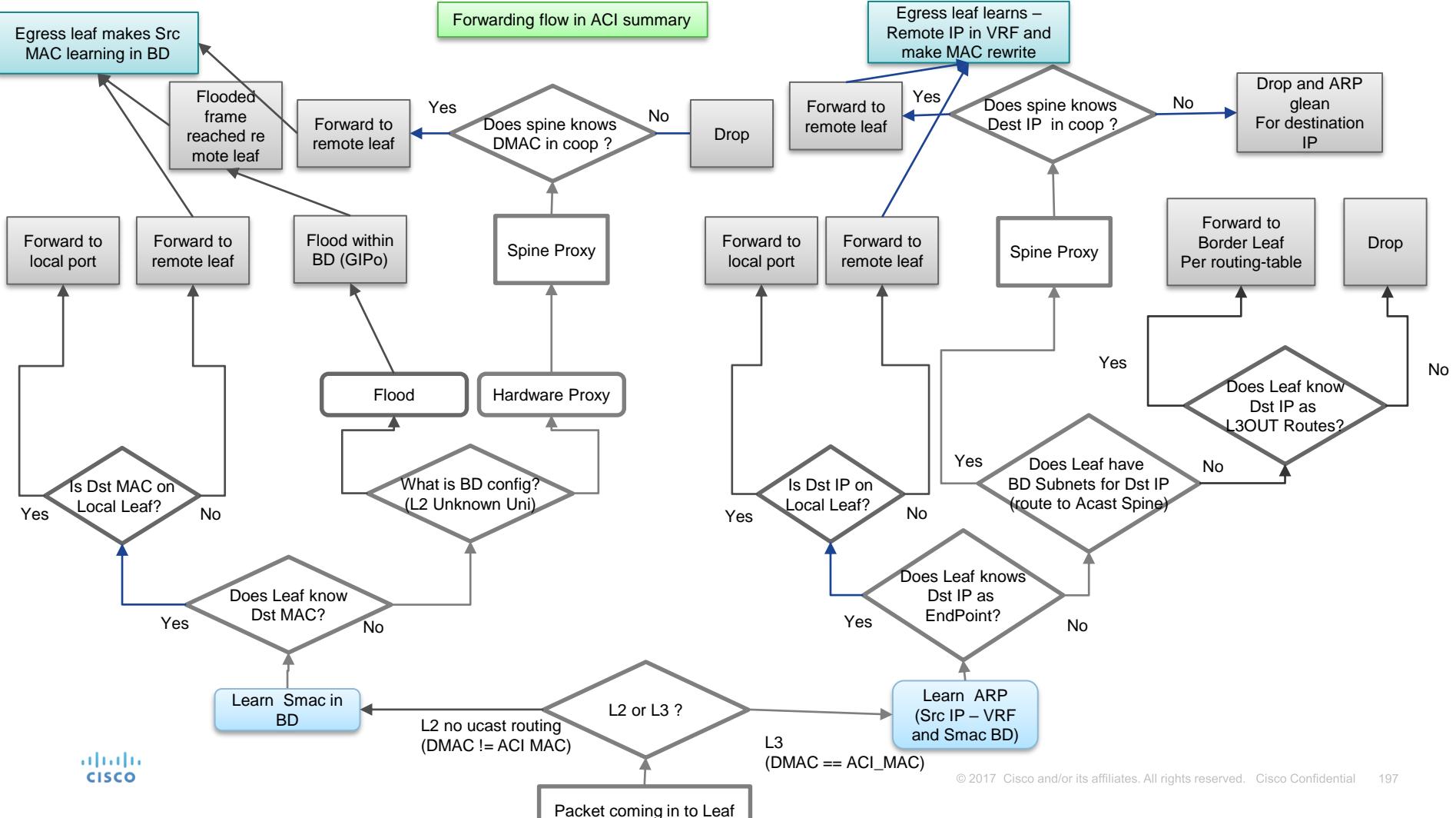
pod2-leaf1# show ip route vrf DC:DC 10.201.1.13
IP Route Table for VRF "DC:DC"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

10.201.1.0/24, ubest/mbest: 1/0, attached, direct, pervasive
  *via 10.0.232.65%overlay-1, [1/0], 04w28d, static
    recursive next hop: 10.0.232.65/32%overlay-1
```



```
pod2-spine1# show ip interface vrf overlay-1 | egrep -B 1 "10.0.232.65"
loopback3, Interface status: protocol-up/link-up/admin-up, iod: 85, mode: anycast-v4, vrf_vnid:
16777199
  IP address: 10.0.232.65, IP subnet: 10.0.232.65/32
pod2-spine1#
```

CheatSheet



End Point Control Feature

What can modify EP and learning behavior

Unicast routing per BD

- **Unicast routing disable on BD:**
 - Subnet under BD not pushed to leaf (useless), no routing, no IP learning, only MAC learning per BD
- **Unicast routing enable on BD with no BD subnet:**
 - Even with no BD subnet configured, we may still learn IP information from ARP packets. But no routing will occur (As no SVI)
 - Not recommended
- **Unicast routing enable on BD and BD subnet**
 - Routing occurs in ACI
 - Learning from ARP and src IP / src MAC by default

Limit IP learning to subnet

Alleviate Rogue Source IP learning issue

- Use in BD – Enforce subnet check for IP learning
- Will disable Src IP learning (from IP or from ARP glean) in case the source IP is not part of BD subnet
- Only works on ingress leaf (aka packet from front panel)
- **Do only prevent epm learning BUT DO NOT prevent Forwarding**
- Remote learning might still be done

Example where it is not enough to fully prevent rogue IP:

- Packet from rogue IP enters Leaf 1
- Limit ip learning to subnet prevent Leaf 1 to learn the rogue IP
- If packet is a BUM it is still send to the fabric (GIPo flooding or anycast to spine)
- Remote Leaf will learn that IP with a tunnel pointing to Leaf 1
- If that IP is a real IP in the VRF:
 - If real location is leaf 1 → that's fine
 - If real location is not leaf 1 → flap on remote leaf between leaf 1 and real leaf where EP is

Limit IP Learning to Subnet

Available in 1.1(1)

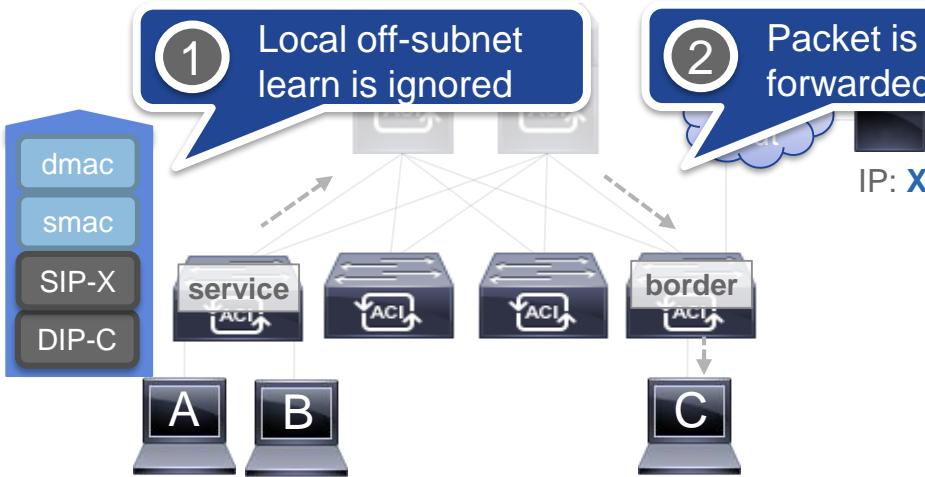
Tenant -> Networking -> Bridge Domain

The screenshot shows the Cisco Application Centric Infrastructure (ACI) User Interface. The top navigation bar includes tabs for System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, and Integrations. The Tenants tab is selected. Below the navigation is a search bar for 'Tenant Search: name or descr' and a filter bar with categories: common, ag, mgmt, scale, and infra. The left sidebar lists tenants: ag, bd1, bd2, bd3, bd4, VRFs, External Bridged Networks, External Routed Networks, Dot1Q Tunnels, Contracts, Policies, and Services. Under tenant 'ag', 'Bridge Domains' is expanded, showing 'bd1' which is also selected in the sidebar. The main content area displays the 'Bridge Domain - bd1' configuration. It has tabs for Summary, Policy, Operational, Stats, Health, Faults, and History. The 'Policy' tab is selected, showing sub-tabs for General, L3 Configurations, and Advanced/Troubleshooting. Under 'General', there are several policy settings: L2 Unknown Unicast (Flood, Hardware Proxy), L3 Unknown Multicast Flooding (Flood, Optimized Flood), IPv6 L3 Unknown Multicast (Flood, Optimized Flood), Multi Destination Flooding (Flood in BD, Drop, Flood in Encapsulation), PIM (checkbox), IGMP Policy (select an option dropdown), ARP Flooding (checkbox), and IP Data-plane Learning (no, yes). The 'IP Data-plane Learning' setting is highlighted with a blue border. Below these is a section for 'Limit IP Learning To Subnet' with a checked checkbox. Other policy settings include Endpoint Retention Policy (select a value dropdown), IGMP Snoop Policy (select a value dropdown), and MLD Snoop Policy (select a value dropdown). A note states: 'This policy only applies to local L2, L3, and remote L3 entries'.

- Default setting for new BDs created in 2.3(1e) and 3.0(1k) and above.

Issue #3

Fix: Limit IP Learning to Subnet (Partial Fix)



Limit IP learning to subnet prevents off-subnet learn on local leaf but border leaf cannot apply off-subnet logic on XR frame since BD information is not present in packet, only VRF VNID in iVXLAN header

Interface	Detail
A tun1	XR -
B tun1	XR -
C eth1/1	local learn
X tun1	XR -> Service Leaf

Service Leaf (SL)

Addr	Interface	Detail
A	eth1/1	local learn
B	eth1/2	local learn
C	tun6	XR -> Border Leaf

Prefix check EPMC traces and BD vlan settings

```
[2016 Jan 20 16:46:02.196277946:3577388355:epmc_validate_ep_req:3167:t] PFX_CHK: Checking 13-lrn validity  
for ip_pfx 172.19.112.17  
[2016 Jan 20 16:46:02.196279254:3577388356:epmc_is_local_13_lrn_valid:1034:t] PFX_CHK: matching ip_addr  
172.19.112.17 in ip_pfx_list of bd_vlan 47  
[2016 Jan 20 16:46:02.196281327:3577388357:epmc_validate_ep_req:3175:E] PFX_CHK: 13-lrn invalid for  
local-ep ip_addr 172.19.112.17
```

```
module-1# show system internal epmc vlan 12 detail

VLAN 12
VLAN type : Tenant BD
hw id : 5 :: scope : 7 :: sclass : 16386
access enc : (INVALID, 0)
fabric enc : (VXLAN, 14942176)
BD vlan id : 12 :: BD vnid : 14942176 :: VRF vnid : 3014656
EP retention policy valid : Yes
Local EP timeout : 900 :: Remote EP timeout : 300
EP bounce timeout : 630 :: EP hold timeout : 300 :: EP move frequency : 256
fwd_mode : route,bridge :: fwd_ctrl : mdst-flood,arp-flood,ip-lrn-pfx-check :: bridge_mode: mac :::
unk_mac_ustcast: flood
Endpoint count : 3
Learning disabled :no
No of ep moves 0 since 00:00:00.000000
BD Subnet ip_pfx-1 : 10.200.2.254/24
MAC CKT HT is empty
::::
```

Enforce Subnet Check

Available in 2.2(2)
and 3.0(2)

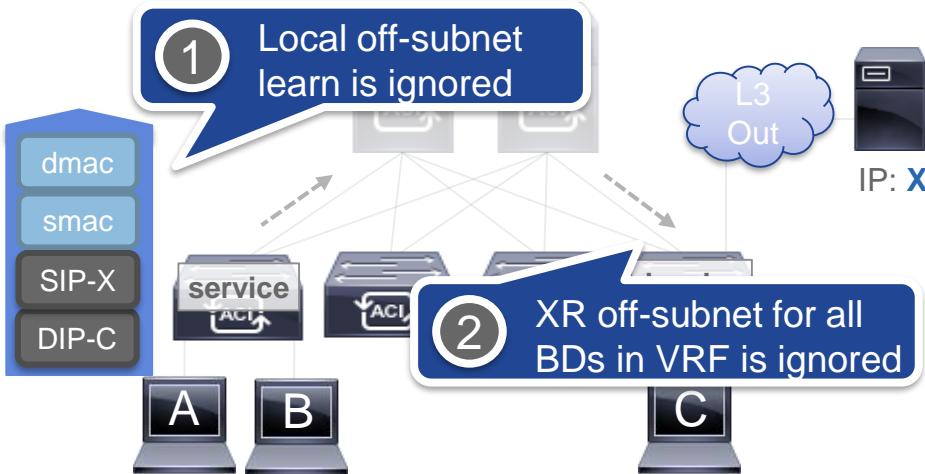
System -> System Settings -> Fabric Wide Settings

The screenshot shows the Cisco Fabric Manager interface with the following navigation path: System -> System Settings -> Fabric Wide Settings. The main content area displays the 'Fabric Wide Setting Policy' page. On the left sidebar, under 'System Settings', the 'Fabric Wide Setting' option is selected. The main panel shows several policy options with checkboxes:

- Disable Remote EP Learning: To disable remote endpoint learning in VRFs containing external bridged/routed domains.
- Enforce Subnet Check:** To disable IP address learning on the outside of subnets configured in a VRF, for all VRFs. (This option is highlighted with a blue border.)
- Reallocate Gipo: Relocate some non-stretched BD gipos to make room for stretched BDs.
- Enforce Domain Validation: Validation check if a static path is added but no domain is associated to an EPG.
- Opflex Client Authentication: To enforce Opflex client certificate authentication for GOLF and Linux.



Enforce Subnet Check



- This feature is available only for **Gen2** switches and above
- This **implicitly** enables local subnet check whether it is enabled or not enabled on the BD (i.e., **Limit Ip Learning to Subnet on the BD is no longer required**).
- For remote learns, the **IP** is only learned if the IP belongs to at least BD in the VRF.

Difference for local learning

Limit IP learning to subnet (BD)

- Limit only Src IP learning
- Mac still learns
- Works in Software
- Can be tuned per BD
- Not needed if enforce subnet check is globally set

Enforce subnet check (global)

- Prevent both MAC and IP learning if IP is outside of BD subnet
- Works in Hw
- Cannot be tuned by Per BD it is a global settings

Disable Remote IP learning

GUI below is from 3.1
Section of System/System Settings was added in GUI in 3.0

The screenshot shows the Cisco Application Policy Infrastructure Controller (APIC) GUI. The top navigation bar includes the Cisco logo, APIC, and tabs for System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, and Apps. Below the navigation is a secondary menu with QuickStart, Dashboard, Controllers, System Settings (which is highlighted in blue), Faults, Config Zones, Events, Audit Log, and Active Sessions.

The main content area has two sections:

- System Settings**: On the left, a sidebar lists several settings: Quota, APIC Connectivity Preferences, BD Enforced Exception List, Control Plane MTU, Endpoint Controls, **Fabric Wide Setting** (which is selected and highlighted in blue), System Global GIPo, BGP Route Reflector, COOP Group, Load Balancer, and Precision Time Protocol.
- Fabric Wide Setting Policy**: This section contains a "Properties" table with the following rows:
 - Disable Remote EP Learning: To disable remote endpoint learning in VRFs containing external bridged/routed domains
 - Enforce Subnet Check: To disable IP address learning on the outside of subnets configured in a VRF, for all VRFs
 - Reallocate Gipo: Reallocate some non-stretched BD gipos to make room for stretched BDs.
 - Enforce Domain Validation: Validation check if a static path is added but no domain is associated to an EPG
 - Opflex Client Authentication: To enforce Opflex client certificate authentication for GOLF and Linux

Disable Remote EP learning

- Introduced in Disabling Remote EP Learn on Border Leaf – CSCuz19695/ CSCva72341
- Available in 2.2(2)
- Disable remote IP-EP learning (/32 host) on Border leaf
- Only apply to vrf in the default mode of Ingress policy enforcement
- Needed to fix the issue of Stale remote EP on BL after the EP move from one remote leaf to another remote leaves.

Disable Remote EP learning on BL – check

```
module-1# show system internal epmc vrf RD-BGP:RD detail

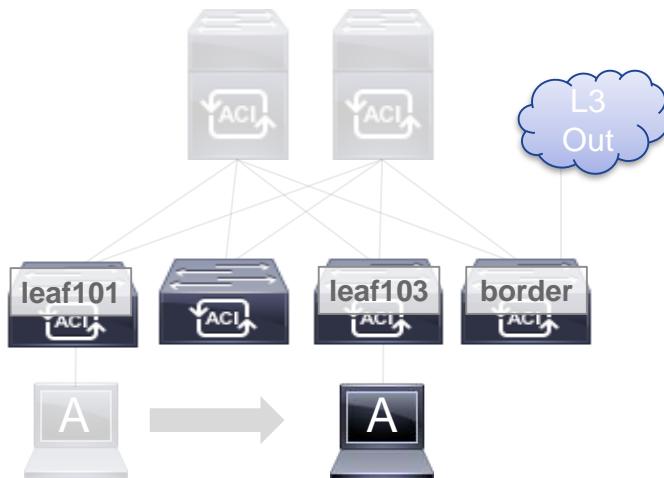
VRF RD-BGP:RD
vrf type : Tenant
context id : 11 :::: vnid : 2654211
v4_usd_tbl_id: 0xb :::: v4_tbl_idx : 0xb :::: v6_tbl_idx : 0x8000000b
Scope : 11 :::: Sclass: 16386
EP retention policy valid : Yes
Local EP timeout : 900 :::: Remote EP timeout : 300
EP bounce timeout : 630 :::: EP hold timeout : 300
EP move frequency : 256
Endpoint count : 2
Border Leaf : yes
Learning disabled :no
Learning xr ip disabled :no
Learning bl xr ip disabled :yes
Policy mode :Ingress
::::
```

Properties

Disable Remote EP Learning: To disable remote endpoint learning in VRFs containing external bridged/routed domains

Stale Endpoint Software Fix

Feature: EP Announce on Bounce Delete



Leaf101

Addr	Interface	Detail
A	tun3, bounce	XR -> leaf103 with bounce bit set

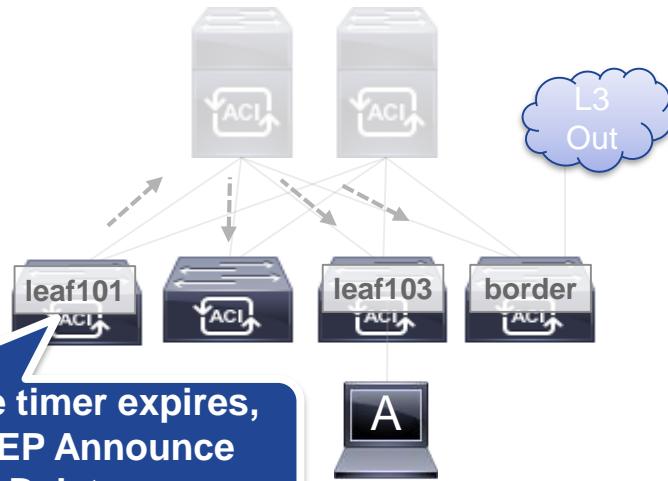
Border Leaf

Addr	Interface	Detail	Hit
A	tun1	XR -> leaf101 TEP	Yes

- Host-A moved from leaf101 to leaf103, a bounce entry is present on leaf 101 for Host-A, and **some flow** is resetting the XR hit-bit on the border leaf toward leaf101

Stale Endpoint Software Fix

Feature: EP Announce on Bounce Delete



Leaf101

Addr	Interface	Detail
A	-	Bounce entry timed-out

Border

Addr	Interface	Detail
A	-	Deleted by announce

Triggers XR delete on any
leaf still pointing to leaf101

- Enabled by default in **3.2.2** and above, no configuration required
- Supports Gen1 and Gen2
- Prevents stale endpoint issues
- CSCvj17665 EP Announce support for stale IP XR EPs

Available in 2.1(1)

IP aging policy

- Introduced in 2.1(1) per CSCut23815 ACI: unused local IP endpoint should be aged out separately from its MAC endpoint
- Without this feature, aging only runs on Mac for Local EP. So an IP could stick forever if the mac never ages out

The screenshot shows the Cisco Application Policy Infrastructure Controller (APIC) web interface. The top navigation bar includes links for System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, and Apps. The user is logged in as 'admin'. The main menu on the left is under 'System Settings' and has a selected item 'Endpoint Controls'. The central panel is titled 'Endpoint Controls' and contains a sub-section 'IP Aging Policy'. At the bottom of this section, there is a 'Properties' table with one row. The first column is 'Administrative State:' and the second column is a button labeled 'Enabled'. A red box highlights this button. In the top right corner of the main panel, there are two tabs: 'Ep Loop Protection' and 'Ip Aging', with 'Ip Aging' being the active tab. Another red box highlights the 'Ip Aging' tab. The bottom right corner of the main panel has buttons for 'Policy' and 'History'.

Administrative State:	Enabled
-----------------------	---------

IP aging (cont.)

- In a EP with MAC with one or more IP bound to it. Each IP is tracked separately .
- Also relies on Host tracker to send ARP refresh at 75% of the aging interval if no response that ip is age out.

IP aging use case

- Before this option was available, an endpoint (such as an interface on a virtual machine) might have unused IP addresses stuck on the same MAC address. For example, when booting, a Microsoft Windows virtual machine that does not receive a Domain Host Configuration Protocol (DHCP) address (and does not have a static IP address) will automatically obtain an address from the 169.254.0.0/16
- At some point, the virtual machine will obtain a routable address, and the endpoint will then consist of one MAC address and two IP addresses
- Without this option the 169 address may never age out

Disable Data Plane Learning

new in 4.0

ACI Endpoint learning

- By default, ACI fabric learns endpoint IP and MAC through ARP/GARP/ND and also through dataplane.
- In traditional network, IP and MAC mapping is happened through ARP/GARP/ND (not through dataplane). So there is a difference between ACI fabric and traditional network. For some use cases, we need to disable dataplane endpoint IP learning. (For example, DSR or some Act/Act Nic teaming mode)

To disable dataplane IP learning

Prior to 4.0 release

- We have two options. (Prior to 4.0 release)
- Configure L4-L7 VIP under EPG
 - Need to specify /32 host IP to disable data path learning
 - Good for the case when we know which IP we should disable dataplane learning.
 - **The only tested and supported use case for this option is with Layer 2 DSR.**
- Disable dataplane learning on BD
 - Good for the case when we have many IPs we should disable dataplane learning or we are not 100% sure which IPs we should disable dataplane learning. (for example, servers are managed by different team)
 - It's for PBR use case only. There is a consideration for general use case. **Disabling dataplane learning at the BD level is not considered a "General Availability" feature.**

To disable dataplane IP learning

After 4.0 release

- **Disable dataplane IP learning on VRF**
 - It's available for non-PBR case as well.
 - Dataplane IP learning is disabled on VRF
 - L3 multicast is supported.
 - Local MACs and remote MACs still get learned from data plane.
 - **Local IPs are not learned from dataplane.**
 - Remote IPs are not learned from unicast packets but from multicast packets from data plane.
 - **Local IPs still get learned from ARP/GARP/ND.**
 - When Dataplane IP learning is disabled on VRF
 - Existing remote IPs are flushed immediately.
 - Existing learnt local IPs are retained, but will be aged unless control plane packets keep it alive
 - Bounce entries will be deleted when bounce timer expires

Configuration

- Tenant > Networking > VRFs > VRF

The screenshot shows the Cisco Application Policy Infrastructure Controller (APIC) interface for Tenant ABC. The left sidebar lists various networking components under 'VRFs'. The 'vrf' item is selected and highlighted with a red box. On the right, the 'Policy' tab is active, displaying configuration options for OSPF Timers, OSPF Context Per Address Family, Endpoint Retention Policy, Monitoring Policy, and EIGRP Context Per Address Family. At the bottom, there are fields for creating an SNMP context, route target profile, and DNS labels. The 'IP Data-plane Learning' checkbox is explicitly highlighted with a red box and a callout box stating 'Enabled by default'.

IP Data-plane Learning
Enabled by default

Switch CLI

check status

```
F2-P1-Leaf-301# show system internal epm vrf ABC:vrf detail

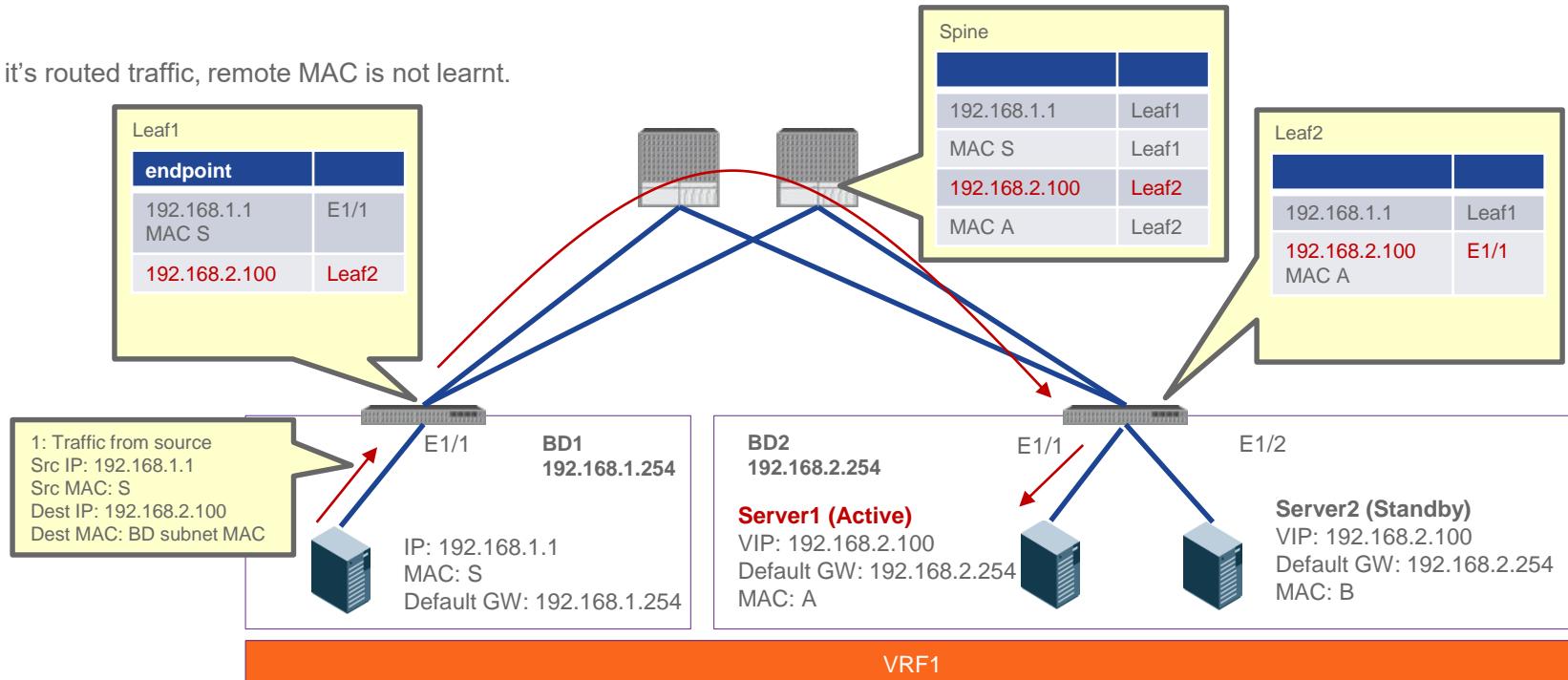
VRF ABC:vrf
vrf type : Tenant :::: vrf valid : yes
context id : 7 :::: vnid : 2392068
Scope      : 2392068 :::: Sclass: 16386
EP retention policy valid : Yes
Local EP timeout : 900 :::: Remote EP timeout : 300
EP bounce timeout : 630 :::: EP hold timeout : 300
EP move frequency : 256
Valid : Yes :::: Learn Enable : Yes :::: IP Learn Enable : No
Endpoint count : 5
::::
```

Use case example 1

Failover of VIP in server cluster (when we need disable dataplane IP learning?)

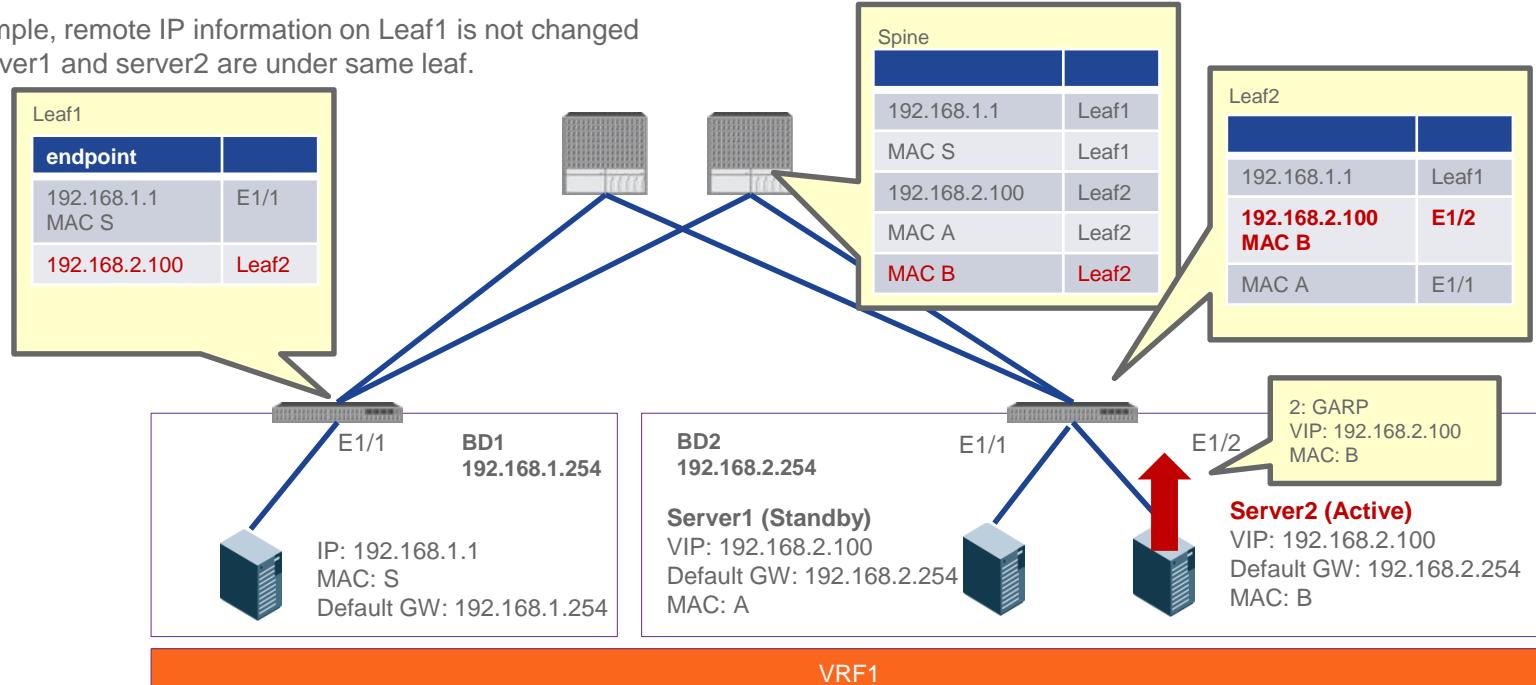
Use case example 1 – Cluster Server failover

If it's routed traffic, remote MAC is not learnt.



Use case example 1(cont) Server2 takes over Active role

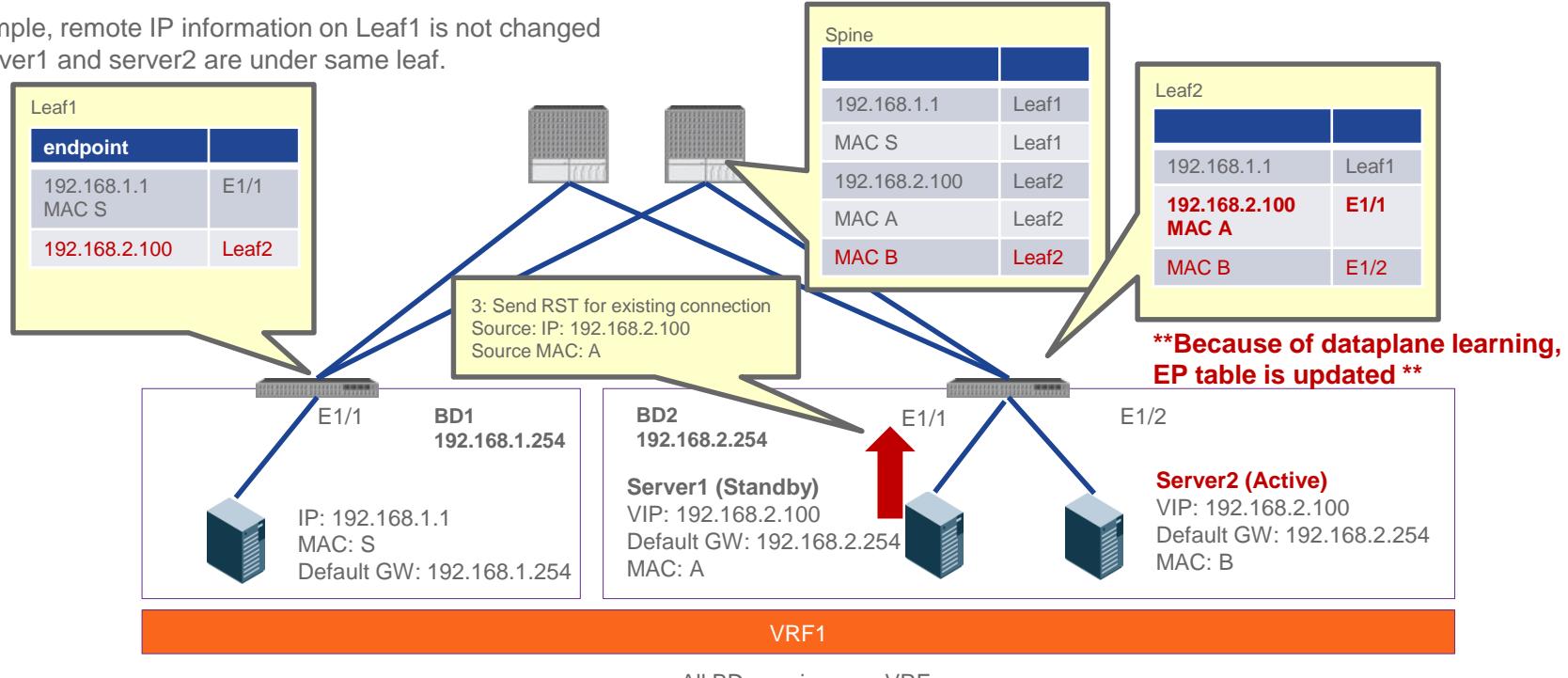
In this example, remote IP information on Leaf1 is not changed as both server1 and server2 are under same leaf.



All BDs are in same VRF

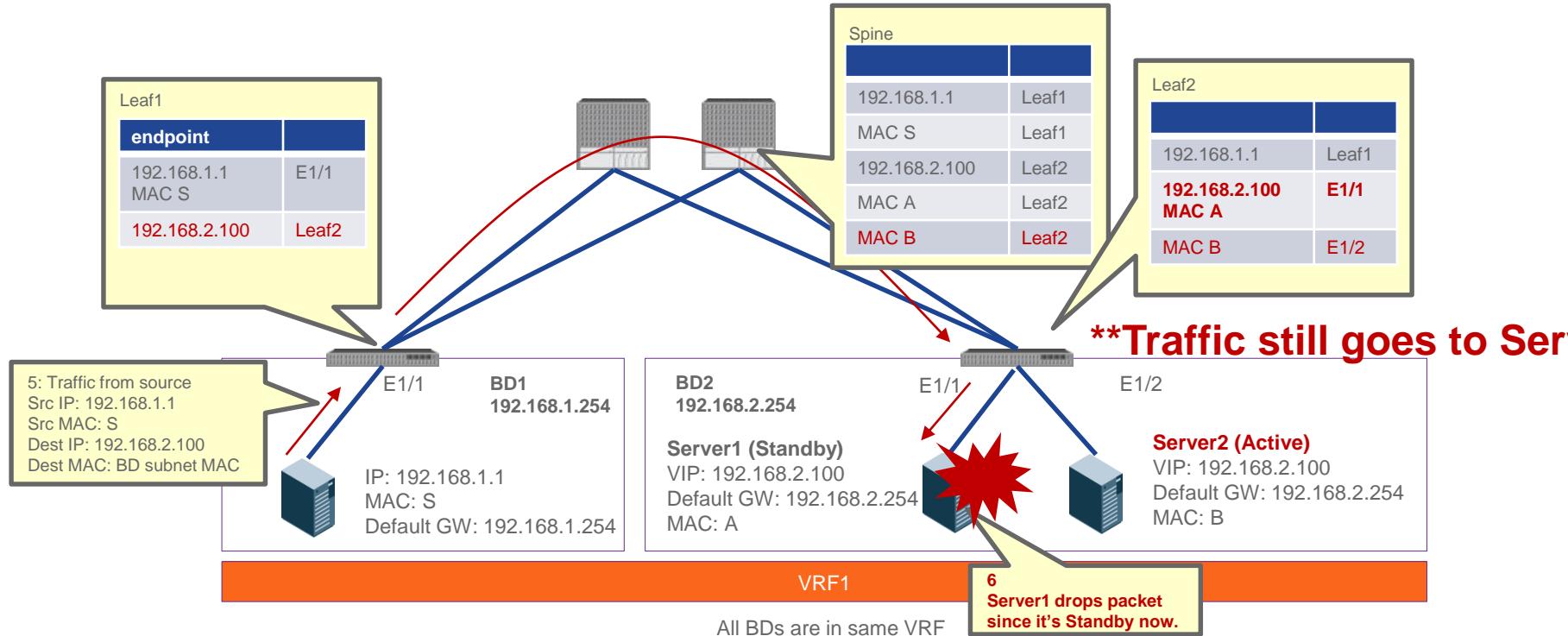
Use case example 1(cont) Server1(standby) send RST

In this example, remote IP information on Leaf1 is not changed as both server1 and server2 are under same leaf.



Use case example 1(cont)

Client tries to re-connect



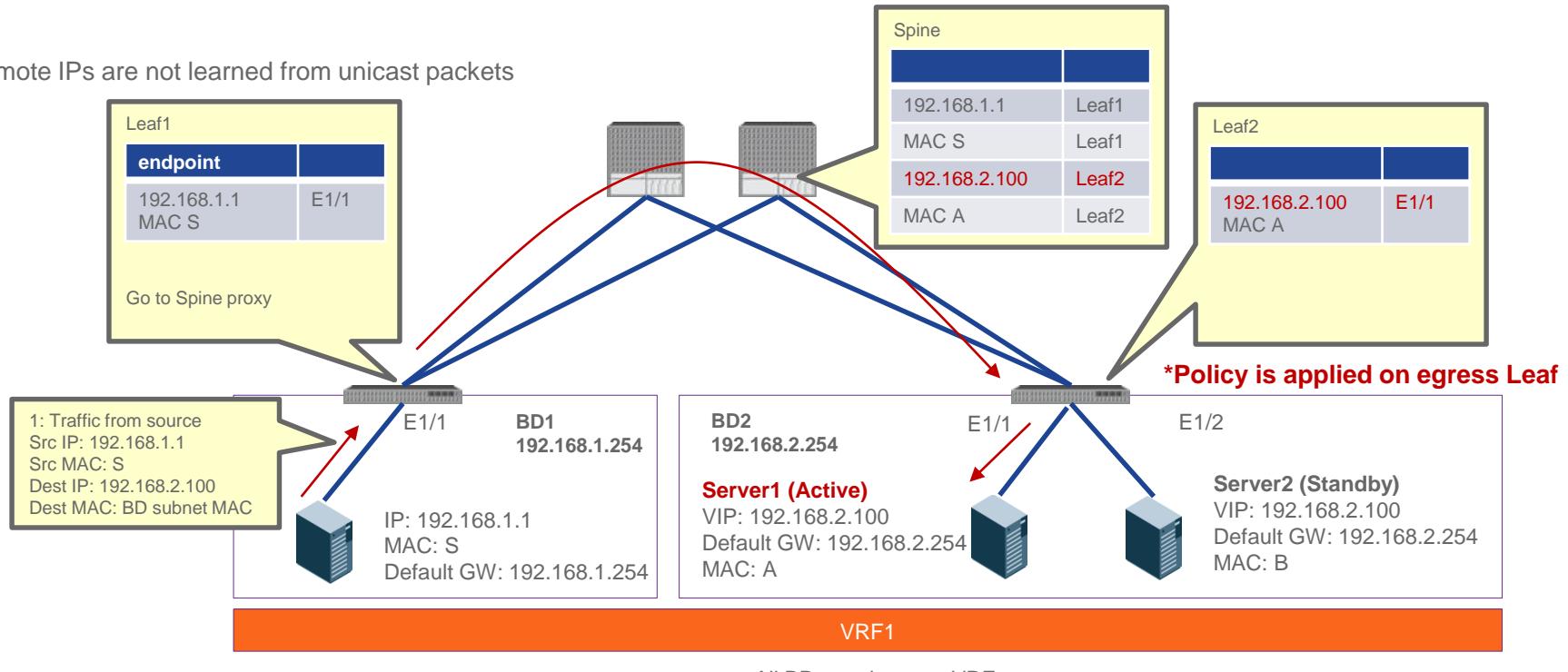
Use case example 1

If dataplane IP learning is disabled

- Local MACs and remote MACs still get learned from data plane.
- Local IPs are not learned from dataplane.
- Remote IPs are not learned from unicast packets but from multicast packets from data plane.
- Local IPs still get learned from ARP/GARP/ND.

Use case example 1

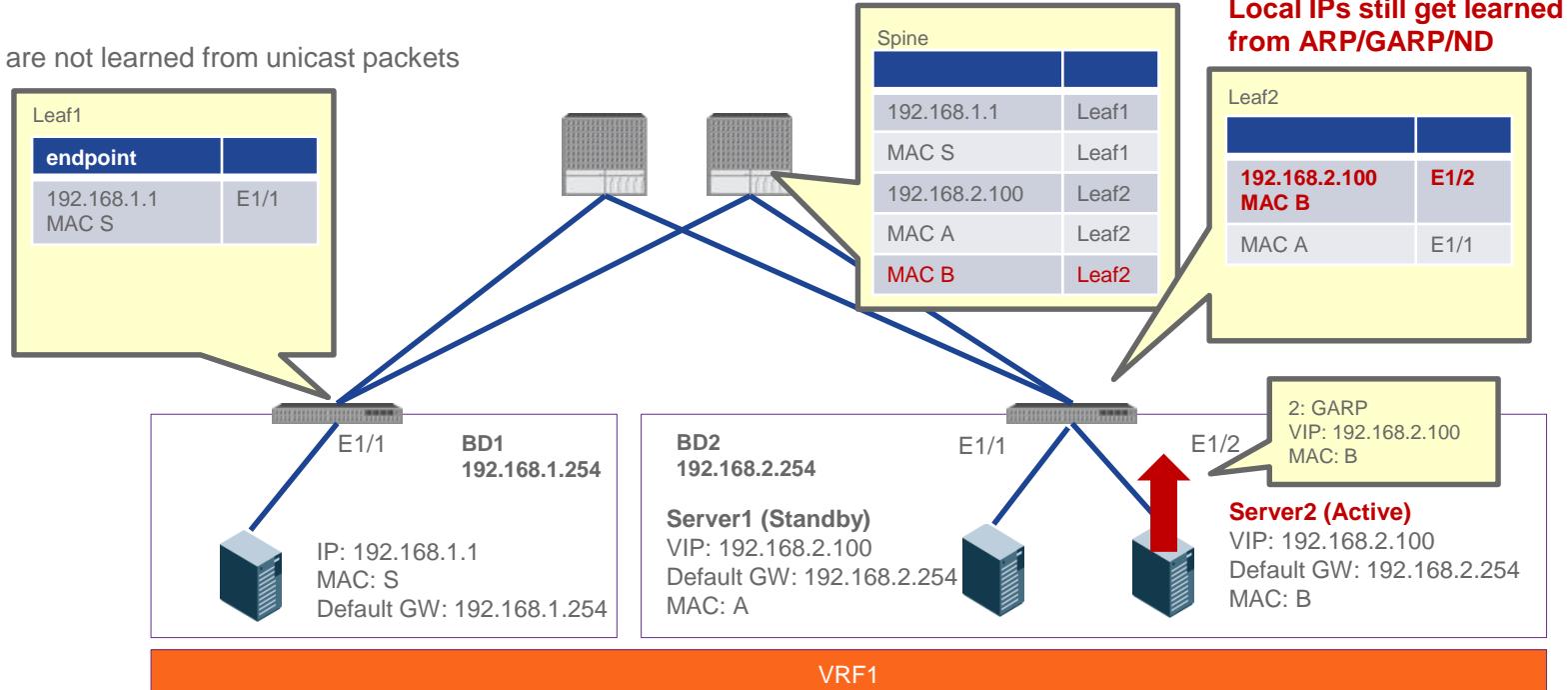
Remote IPs are not learned from unicast packets



All BDs are in same VRF

Use case example 1(cont) Server2 takes over Active role

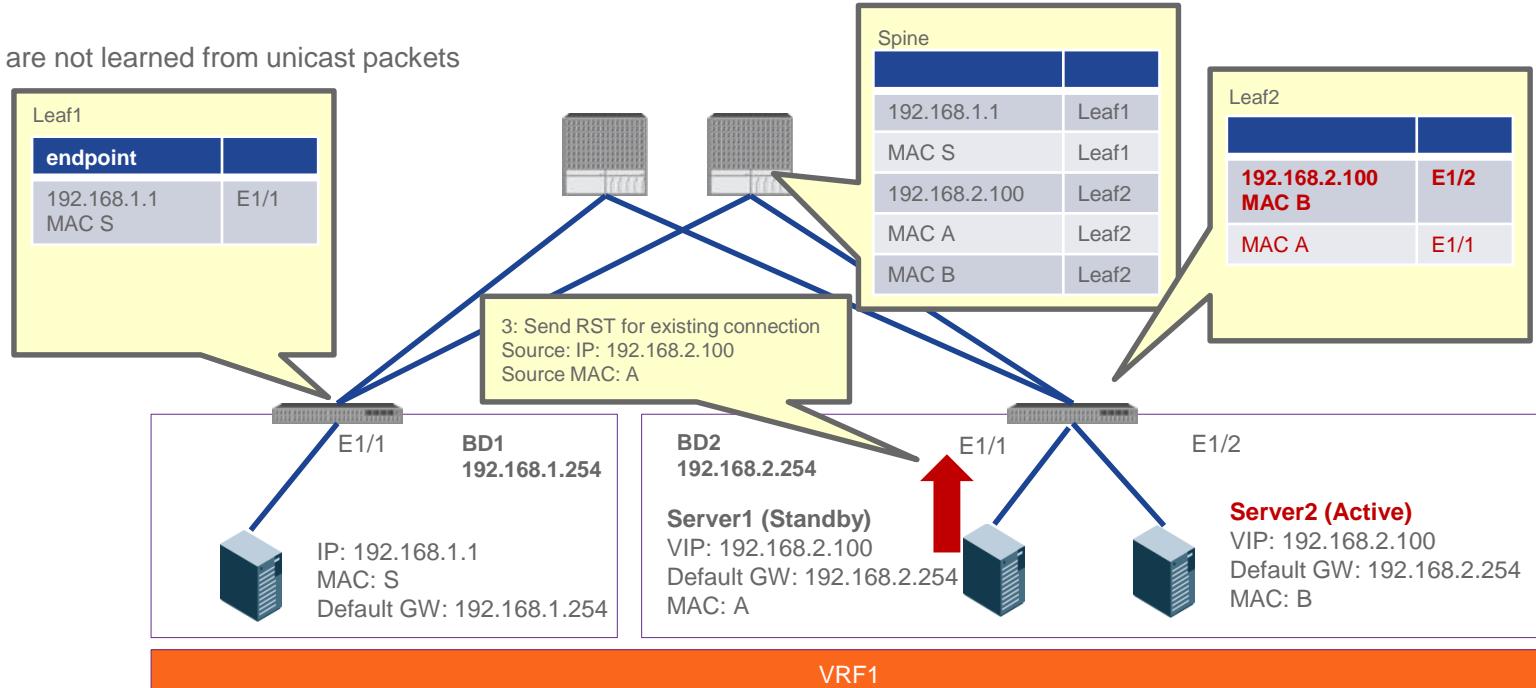
Remote IPs are not learned from unicast packets



Use case example 1(cont) Server1(standby) send RST

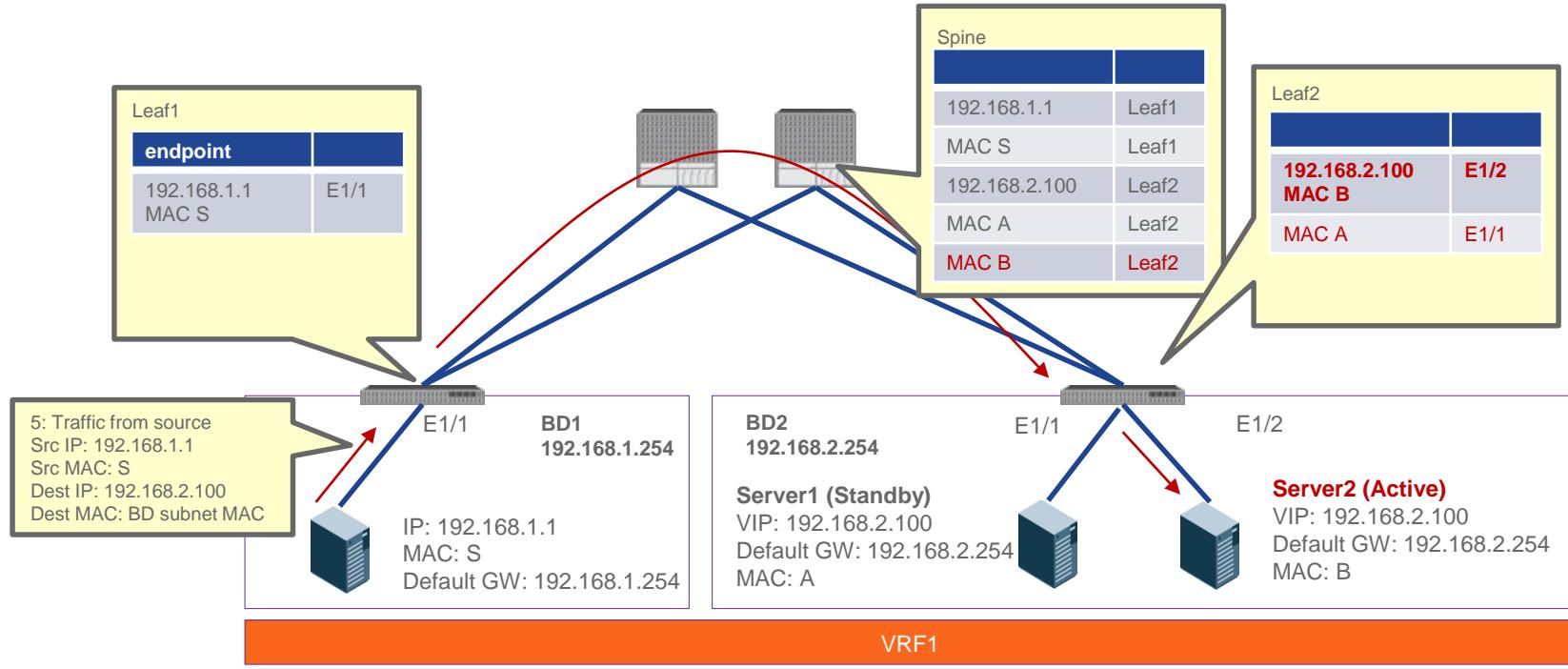
Local MACs still get learned from dataplane.
Local IPs are not learned from dataplane

Remote IPs are not learned from unicast packets



Use case example 1(cont)

Client tries to re-connect



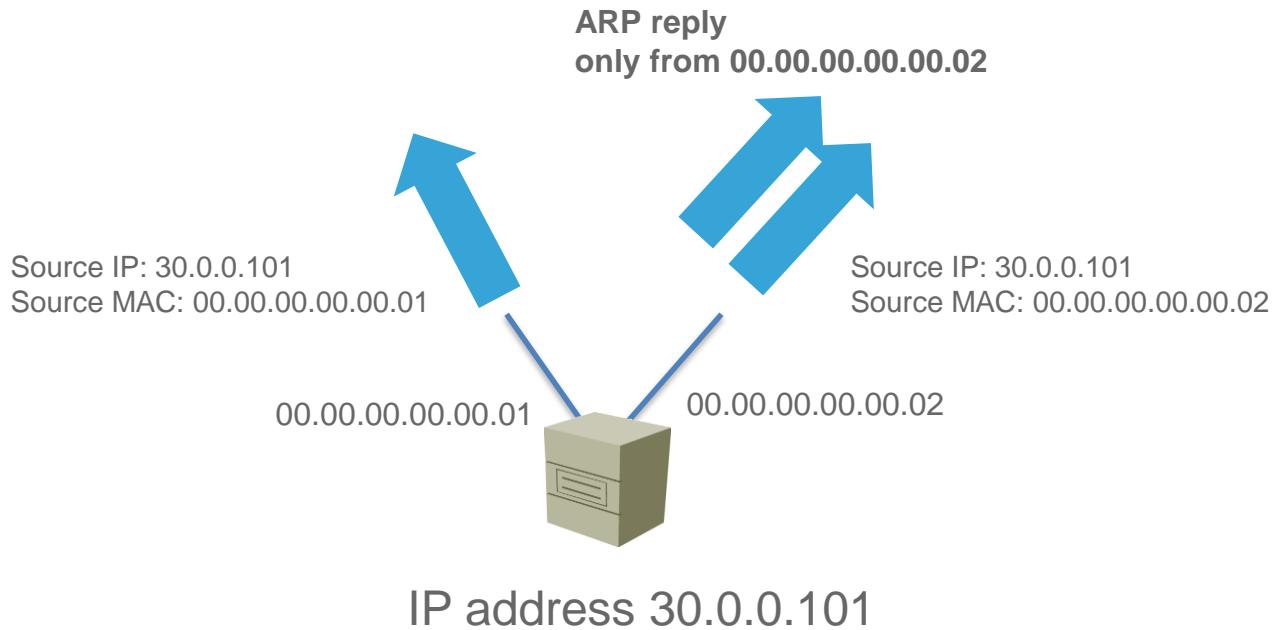
Use case example 2

Active/Active Nic teaming server

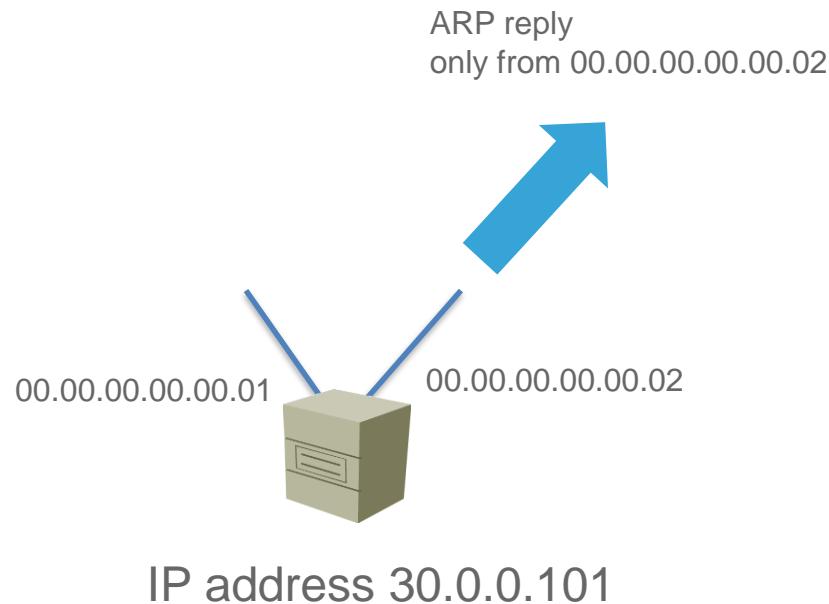
Teaming Modes that don't work with ACI: *Bonding balance TLB and Switch Independent Teaming*

- In Linux bonding this is called mode “5”: *balance-tlb*
 - <https://help.ubuntu.com/community/UbuntuBonding>
 - In Hyper-V it is called:
 - Switch Independent with Dynamic distribution on Hyper-V 2012 R2
-
- Essentially this is asymmetric traffic flow for the server:
 - Server sends Upstream traffic on both NIC
 - But except incoming traffic only one one NIC (the one replying to ARP)
 - NO Port-channel or bonding in this case

Server forwarding traffic with Switch Independent / TLB Outbound



CCOP/EP Database Learning with ARP

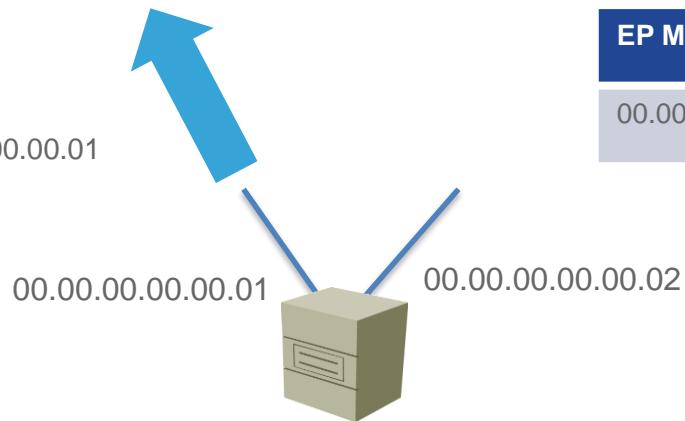


Mapping Database

EP MAC	EP IP	Leaf
00.00.00.00.00.02	30.0.0.101	Leaf2

COOP/EP Database Learning with Routed Traffic

Dmac: RMAC
Source IP: 30.0.0.101
Source MAC: 00.00.00.00.00.01

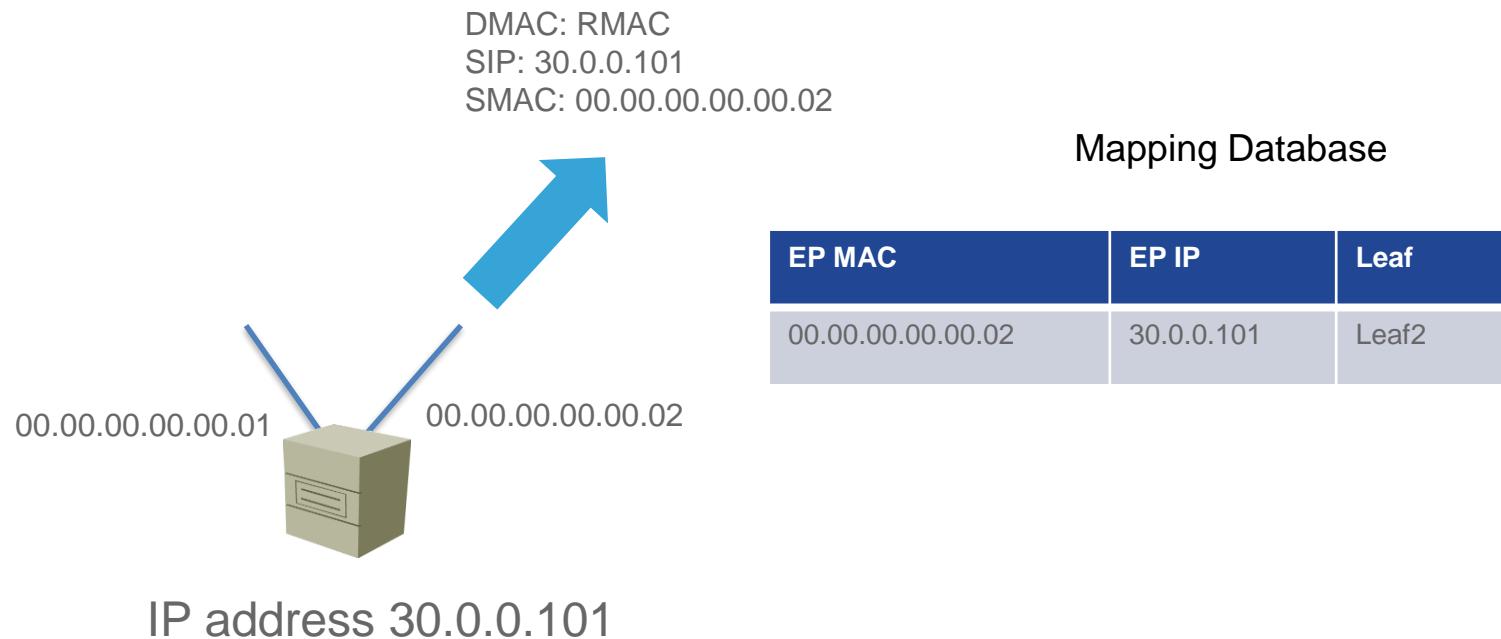


Mapping Database

EP MAC	EP IP	Leaf
00.00.00.00.00.01	30.0.0.101	Leaf1

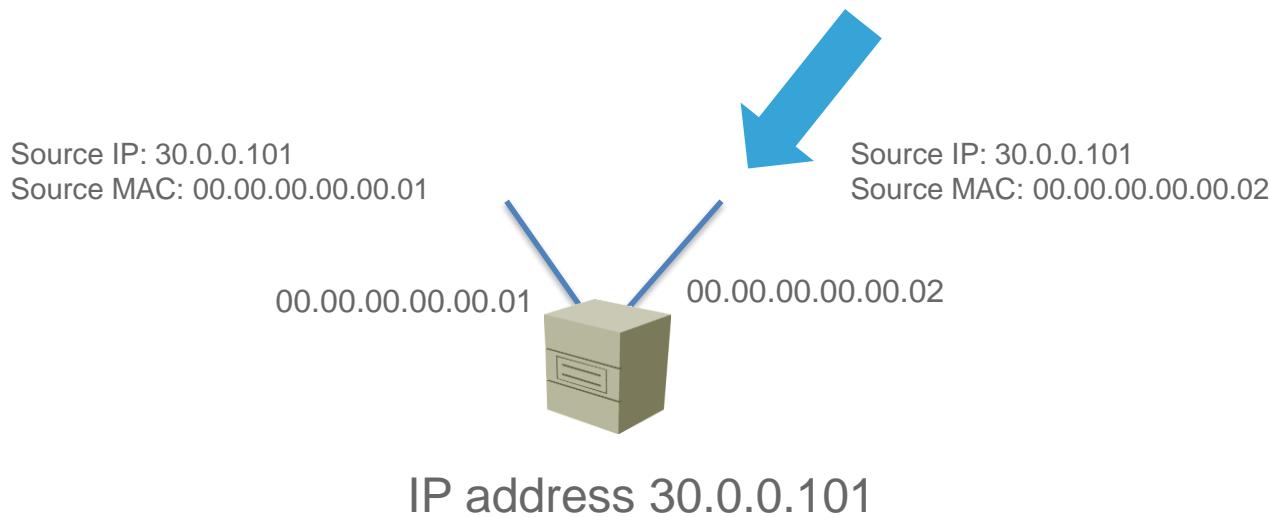
IP address 30.0.0.101

COOP/EP Database Learning with Routed Traffic



Server forwarding traffic with Switch Independent / TLB Inbound

Server Expects inbound traffic from one interface only, the one where it sent ARP replies



Endpoint IP would flap between leaf and ports

- When using this type of Teaming where both NICs can send traffic with the same source IP address the following happens:
- **With the BD configured for Routing the Endpoint IP address would be learned on two different ports / leafs**
- **So this IP address would be continuously flapping**

Solutions

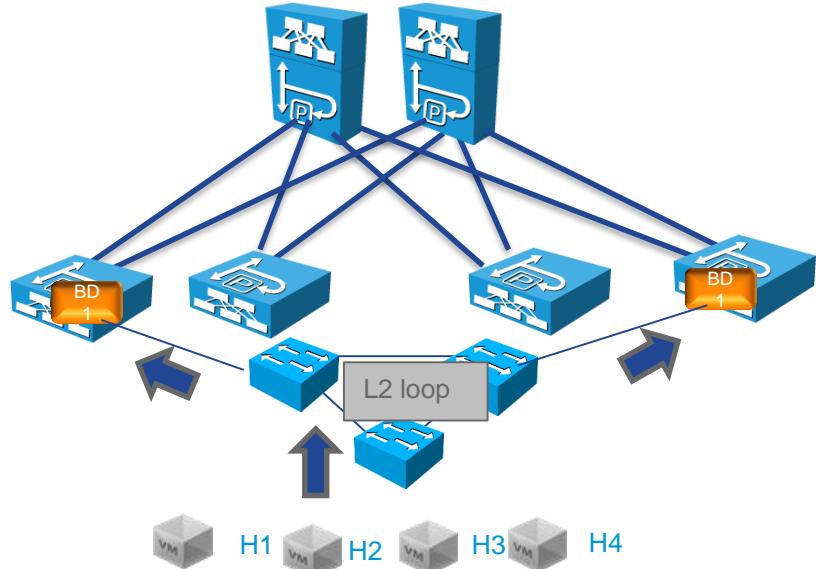
- Use other teaming configurations for Physical Hosts:
 - Active/Standby
 - LACP/Port-Channels
- Use other teaming configurations for Virtualized Hosts:
 - MAC pinning
 - LACP/Port-Channels
- Last resort disable dataplane learning would help (similar to server cluster case before). We would learn only from ARP not from IP datapath

EP move protection against loop

EP move dampening

EP move dampening

- EP move dampening measures the number of moves per second on the BD *of the aggregate of the endpoints*
- *Count is not fully accurate (aka not exactly number of packet moved). It is hardware dependant (we count software notification)*



H1-4 are moving between
Leaf 1 and 4 10 times/s, so
BD sees 40 moves/s

Configured under BD as EP retention policy

Bridge Domain - BD1

The screenshot shows the 'Properties' section of a Bridge Domain configuration. It includes the following settings:

- L2 Unknown Unicast: Flood (selected)
- L3 Unknown Multicast Flooding: Flood (selected)
- Multi Destination Flooding: Flood in BD (selected)
- PIM:
- IGMP Policy: default
- ARP Flooding:
- Endpoint Dataplane Learning:
- Clear Remote MAC Entries:
- Limit in Learning To Subnet:
- Endpoint Retention Policy: select a value (dropdown menu)
- IGMP Snoop Policy: select a value (dropdown menu)

A red box highlights the 'Endpoint Retention Policy' dropdown.

By default when aggregate move per sec within a BD exceed 256. We disable learning for 300 sec an any leaves that got the move (one or more)

Create End Point Retention Policy

Define Endpoint Retention policy

The screenshot shows the 'Create End Point Retention Policy' configuration page. It includes the following fields:

- Name: (input field)
- Description: optional
- Hold Interval (sec): 300 (highlighted with a red box)
- Bounce Entry Aging Interval (sec): 630
- Local Endpoint Aging Interval (sec): 900
- Remote Endpoint Aging Interval (sec): 300
- Move Frequency (per sec): 256 (highlighted with a red box)

Checking move per BD vlan

```
module-1# show system internal epmc vlan 33 detail
```

```
VLAN 33VLAN type : Tenant BD
```

```
hw id : 12 :::: scope : 3014659 :::: sclass : 32770
```

```
[...]
```

```
EP retention policy valid : Yes
```

```
Local EP timeout : 900 :::: Remote EP timeout : 300
```

```
EP bounce timeout : 630 :::: EP hold timeout : 300 ::::
```

```
Endpoint count : 1
```

```
Learning disabled :no
```

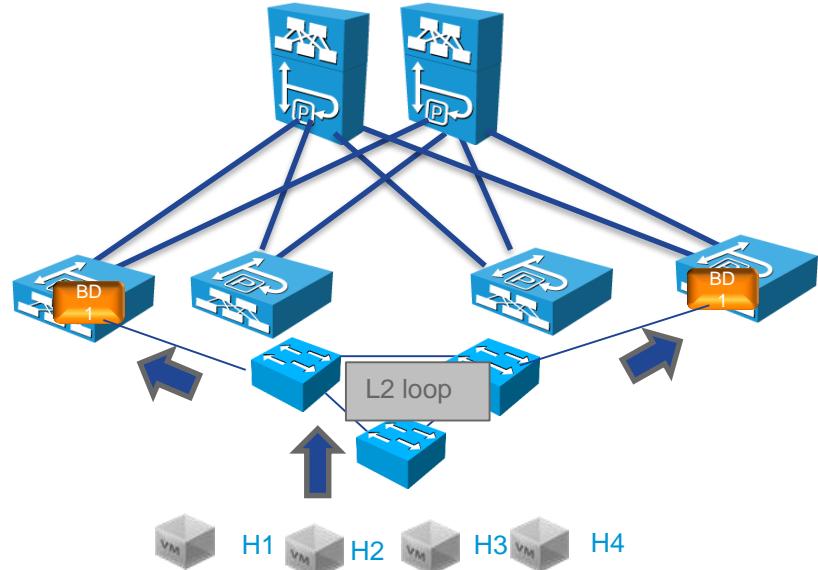
```
No of ep moves 24 since 22:05:22.911670
```

EP move frequency : 256

EP Loop protection

EP loop protection

- EP Loop Protection measures the number of moves per second on the BD *from the individual endpoint perspective*
- Count of move is not exact number of packet moved (hw deduplicate) E.g in gen-2 100 move/sec will be seen as around 10 move/sec
- Disabled by default



H1-4 are moving between
Leaf 1 and 4 10 times/s, so
EP Loop protection sees 10 moves/s

Fabric wide global settings

The screenshot shows the Cisco Application Policy Infrastructure Controller (APIC) web interface. The top navigation bar includes links for System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, and Apps. Below the navigation is a secondary menu with QuickStart, Dashboard, Controllers, System Settings (which is selected and highlighted in blue), Smart Licensing, Faults, Config Zones, Events, and Audit Log.

The main content area is titled "System Settings" and "Endpoint Controls". On the left sidebar under "System Settings", the "Endpoint Controls" option is selected and highlighted in blue. Other options include Quota, APIC Connectivity Preferences, BD Enforced Exception List, Control Plane MTU, Fabric Wide Setting, Port Tracking, System Global GIPo, BGP Route Reflector, COOP Group, Load Balancer, and Precision Time Protocol. The right panel displays "Properties" for Endpoint Controls, showing the Administrative State set to "Disabled" (which is highlighted in blue), a Loop Detection Interval of 60, a Loop Detection Multiplication Factor of 4, and an Action section where "Port Disable" is checked (highlighted in blue).

Here if a leaf detect the same endpoint that moves over 4 times per 60 seconds we will disable the port.

Port disabled remains errdisable unless we configure an errdisable policy

Option 2 is BD learning disable (similar to EP dampening)

In order to recover the port you need to enable Errdisable Recovery

- Fabric / External Access Policy / Policies /Global Policies > Error Disabled Recovery Policy

The screenshot shows the Cisco ACI Fabric UI interface. The top navigation bar includes tabs for System, Tenants, Fabric (selected), Virtual Networking, L4-L7 Services, Admin, Operations, and Apps. Below the navigation bar, a secondary navigation bar shows Inventory, Fabric Policies, and External Access Policies. The main content area is titled "Error Disabled Recovery Policy". On the left, a sidebar menu lists Policies, Switches, Modules, Interfaces, and various policy types like Attachable Access Entity Profiles, QOS Class, DHCP Relay, MCP Instance Policy default, and Error Disabled Recovery Policy (which is selected and highlighted in blue). The main panel displays "Properties" for the Error Disabled Recovery Policy, including an "Error disable recovery interval (sec)" input field set to 300. It also shows sections for "Events" (Frequent EP move, Loop indication by MCP) and "Recover" (both set to False). At the bottom, there are "Update" and "Cancel" buttons, and a checkbox labeled "BPDU Guard" which is checked (indicated by a blue border).

Policies

- > Quick Start
- > Switches
- > Modules
- > Interfaces
- < Policies
 - > Switch
 - > Interface
 - < Global
 - > Attachable Access Entity Profiles
 - > QOS Class
 - > DHCP Relay
 - MCP Instance Policy default
 - Error Disabled Recovery Policy

Recommendation

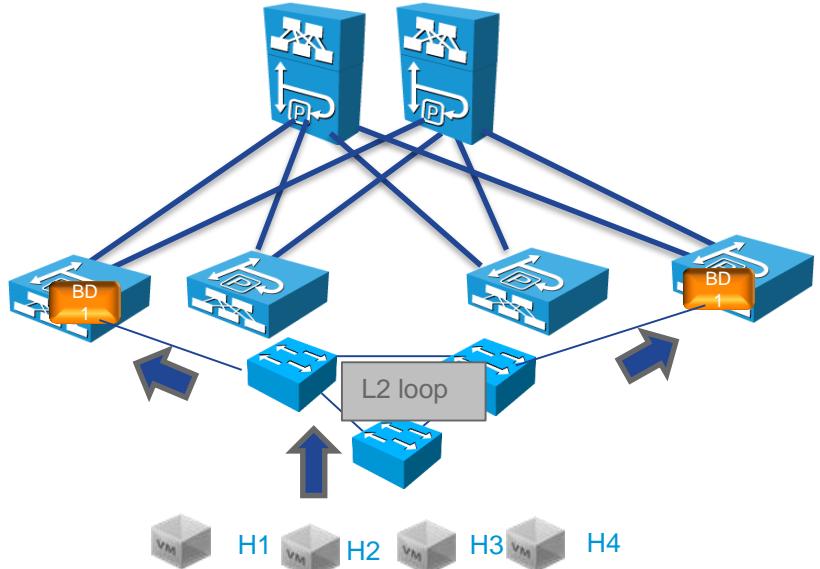
Not RECOMMENDED

- Action is ***potentially disruptive*** to other stable endpoints.
 - BD Learn disable prevents new learns on the entire BD
 - Port disable may impact a critical port such as fabric-interconnect or DCI link.
No mechanism to prioritize a host port.

Rogue EP feature (only to use as of 3.2)

Rogue End point detection

- Rogue Endpoint Detection measures the number of moves per second on the BD *from the endpoint perspective*
- With Rogue EP, only the misbehaving endpoint (MAC/IP) is quarantined and a fault is raised to allow easy identification of the problematic endpoint.



H1-4 are moving between Leaf 1 and 4 10 times/s, so
Rogue Endpoint protection sees 10 moves/s

Rogue Endpoint Detection (from ACI 3.2)

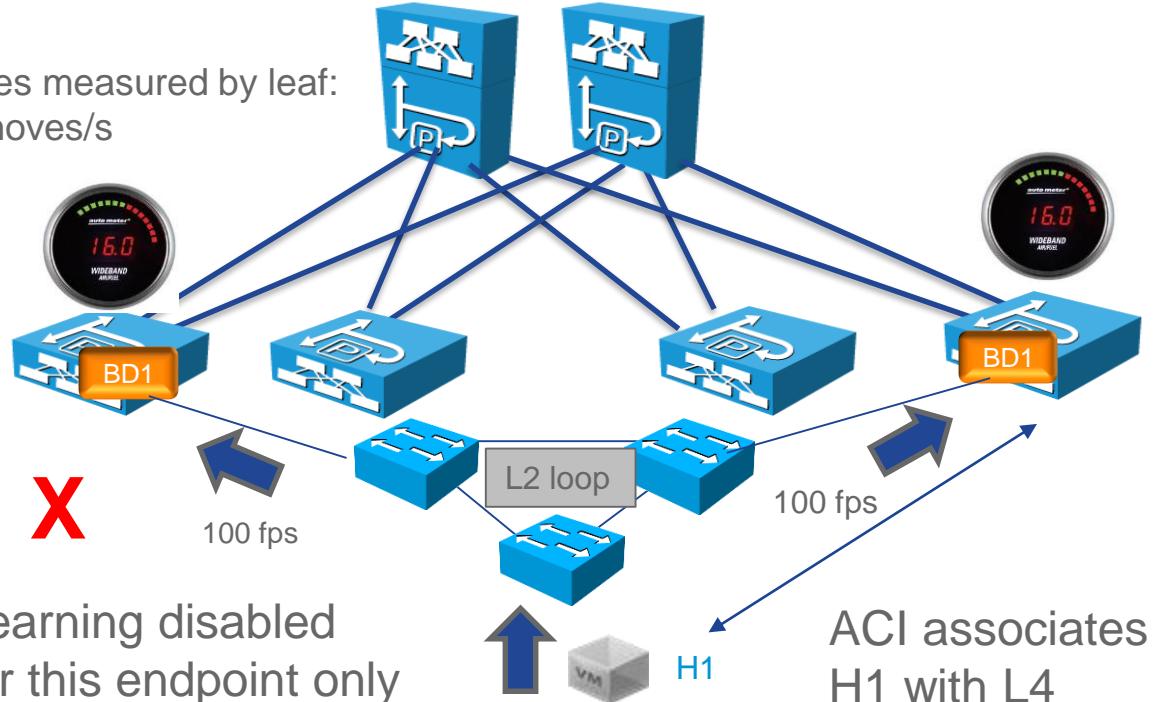
- Scope/Granularity: **Global**
- Move Frequency configured as "Multiplication Factor": you should not enter a value greater than 10 moves (but if you do ACI programs this as 10 on the leafs).
- **You should enter 6 moves.**
- A single endpoint should not move more than 6 times in an interval of 30s
- Rogue EP Detection Interval: minimum is 30s
- **Action:** it stops the movement by temporarily making endpoints static
- Before 3.2(6) When the EP is static Rogue all traffic from/to that EP IS DROPPED !!!!
- After 3.2(6), when EP is Rogue, it is moved to static to the last known location.
- It keeps the endpoint static for a minimum **of 1800 seconds**
- It generates a **host tracking** packet to enable the system to re-learn the impacted MAC or IP address
- **It raises a Fault**

ACI detects too many moves for the MAC, the MAC is statically programmed on one of the leafs



- Moves measured by leaf:
- 10 moves/s
- **Action:**
- Learning disabled for the endpoint that was moving only

Moves measured by leaf:
10 moves/s



Rogue Endpoint Detection Configuration

System Settings

- > Quota
- APIC Connectivity Preferences
- BD Enforced Exception List
- BGP Route Reflector
- Control Plane MTU
- COOP Group
- Endpoint Controls**
- Fabric Wide Setting
- Load Balancer
- Precision Time Protocol
- System Global GIPo

Ep Loop Protection **Rogue EP Control** Ip Aging

Minimum value: 30

Properties

Administrative State:

Disabled **Enabled**

Rogue EP Detection Interval:

30

Rogue EP Detection Multiplication Factor:

6

Hold Interval (sec):

1800

You should enter 6

Minimum value 1800 seconds

Fault is raised on the leaf

The screenshot shows a network management interface with a sidebar on the left and a main content area on the right.

Left Sidebar:

- Topology
- Pod 1
 - leaf1-a1 (Node-301)
 - Chassis
 - Fabric Extenders
 - Interfaces

Main Content Area:

A table listing a single fault entry:

Severity	Acked	Cause	Creation Time	Affected Object	Description	Code
!		ep-mac-is-r...	2018-09-13T23:03:51.9...	topology/pod-1/node-301/sys/ctx-[vxlan-3014659]/bd-[vxlan-15925221]/db-ep/rogueMacEP-00:00:00:00:01	EP MAC 00:00:00:00:01 is rogue on interface tunnel26 of Node 301, Pod 1	F3014

How to Check Rogue Endpoints

```
module-1# show system internal epm endpoint all rogue
VRF : DC:VRF-1 :: Context id : 35 :: Vnid : 3014659
MAC : 0000.0000.0001 :: Num IPs : 0
ep move cnt: 9 ::
```

Packet towards Rogue EP are dropped

If packet is destined to a rogue EP, we will classify it as dest pcTag (dclass) of 0x8 and it will dropped with UC_PC_CFG_TABLE_DROP

```
module-1(DBG-elam-insel14)# report | egrep "leaf.dclass|leaf.sclass|lux_drop_vec"
    sug_lurw_vec.info.ifabric_leaf.dclass: 0x8
    sug_lurw_vec.info.ifabric_leaf.sclass: 0x8002
sug_elam_out_vec.pbx_header_sidebnd_drop_vec.lux_drop_vec: 0x20000000000000000000000000000000
=====
    lux_drop_vec = 0x00200000000000000000000000000000
lux_drop_vec: UC_PC_CFG_TABLE_DROP           condition set
```

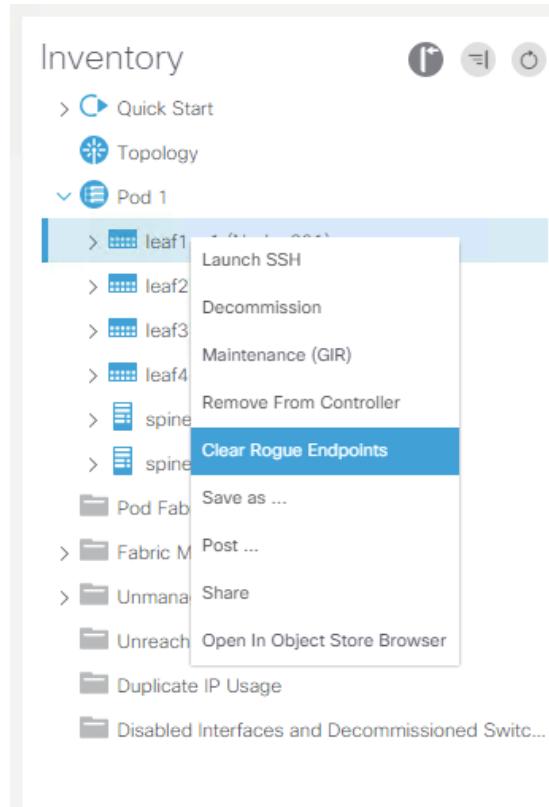
EPMC trace (3.2)

```
[2018 Apr 24 14:24:16.629895508:79177:epmc_ep_track_moves:3119:t] For EP mv ts: 19:28:23.660803 move cnt: 6
[2018 Apr 24 14:24:16.629896604:79178:epmc_ep_track_moves:3125:E] EP moving too frequently
Sending ep event ROGUE_MAC_ON      EPM EP MAC key :: BD 16351139 MAC:=0050.56b8.51cb
Sending ep event ROGUE_IP_ON       EPM EP IPV4 key :: VRF 3112963 IP:=56.9.0.102

[2018 May  3 22:34:03.295805925:15711800:epmc_track_ip_moves:3041:t] For IP mv ts: 23:07:36.890802 move cnt: 10
[2018 May  3 22:34:03.295807242:15711801:epmc_track_ip_moves:3044:E] IP 10.3.1.1 moving too frequently
[2018 May  3 22:34:03.295808452:15711802:epmc_ep_age_init:558:t] Inititing the age entry index to 0
[2018 May  3 22:34:03.295809719:15711803:epmc_ep_update_aging:1240:t] Init the timer index to 0
[2018 May  3 22:34:03.295810552:15711804:epmc_ep_age_init:558:t] Inititing the age entry index to 0
[2018 May  3 22:34:03.295812314:15711805:epmc_ep_age_add:607:t] adding timer for Rogue for 1800 secs for EP
0x106ea004 IP 0x104055fc l3_idx 0          EPM EP IPV4 key :: VRF 2850816 IP:=10.3.1.1
```

Clear Rogue Endpoints

- In Fabric > Inventory,
- right-click on the leaf then select
- Clear Rogue Endpoint



Design Limitations

- Changing rogue parameters will not affect existing rogue EPs
- If rogue EP is enabled loop detection and BD move frequency will not take effect
- Disabling rogue EP feature will clear all rogue EPs
- **During upgrade/downgrade rogue EP should be disabled explicitly by the user**
- EPM has its limit for rogue EP parameters. If user configures outside this range a fault will be raised for each mismatch parameter
- **Rogue EP is not supported in RL/Multisite**

ACI L2 extension and spanning-tree

Two forms of L2 extension for ACI

- L2 extension refers to extending a BD or an EPG to external devices
- There are essentially two ways to make this happen:
 - **Static binding to a path** (port + encap-ID) inside an EPG
 - This is exactly like adding a bare-metal host to an EPG
 - **Using a External Bridged Network configuration (EBN)**
- Both approaches differ conceptually:
 - Extending an EPG only extends that EPG
 - External MACs are learnt, BPDUs are forwarded inside the EPG
 - External Bridged Networks extend the entire BD
 - EBN are governed by contracts (they are modeled as External EPGs)
 - Mac are also learned

EPG extension

- Only extend an EPG
- All EP behind the l2 extended port are EP in the EPG
- All is part of same EPG hence no contract can be applied

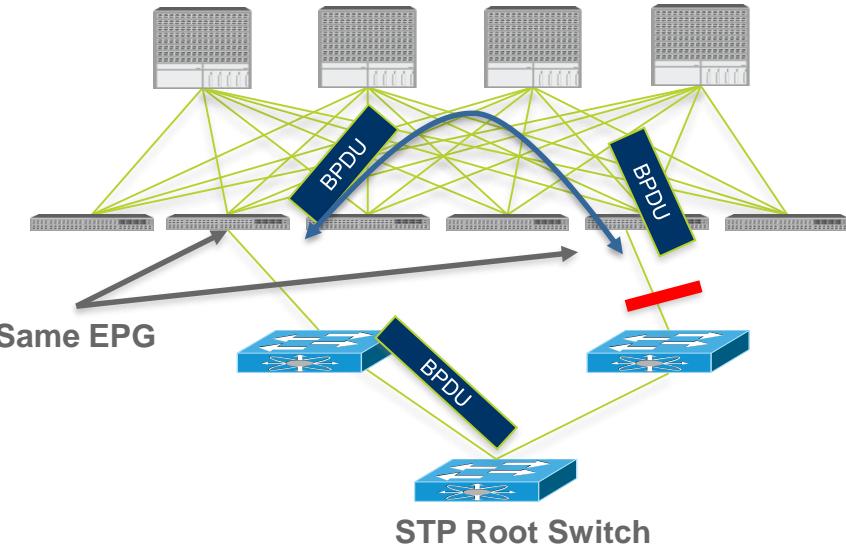
L2 out

- Extend a full BD
- Contract can be configured between L2 out and infra EPG

Spanning-Tree with ACI

ACI Interaction with STP in Legacy Network

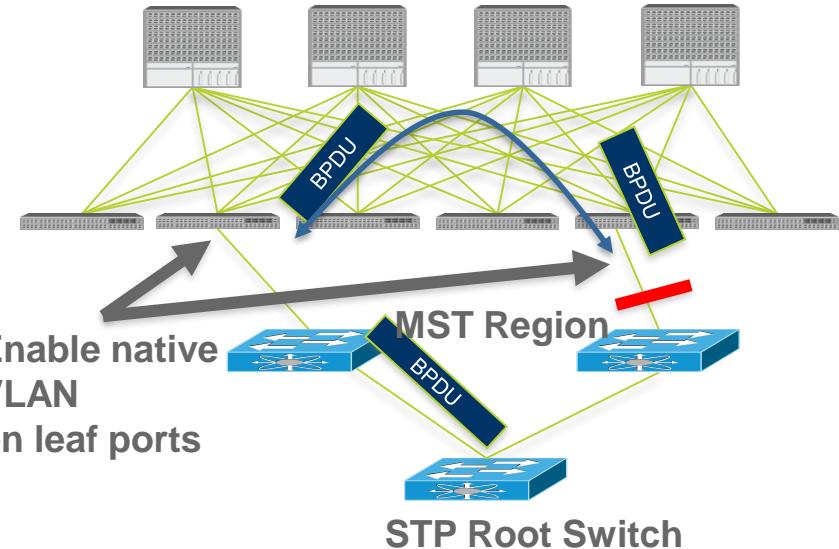
- No STP running within ACI fabric
- BPDU frame is flooded **within EPG**.
No configuration required
- External switches break any potential loop upon receiving the flooded BPDU from ACI fabric
- BDPU guard is enabled on FEX ports by default
- User can enable BDPU guard on leaf edge ports
- PVST+ - BPDU tagged with vlan ID – OK



ACI Interaction with MST

Configuration Required for BPDU Flooding

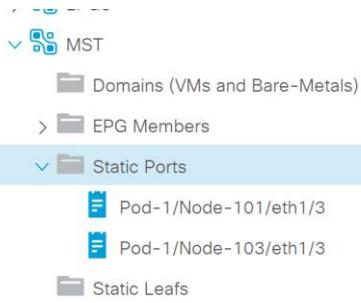
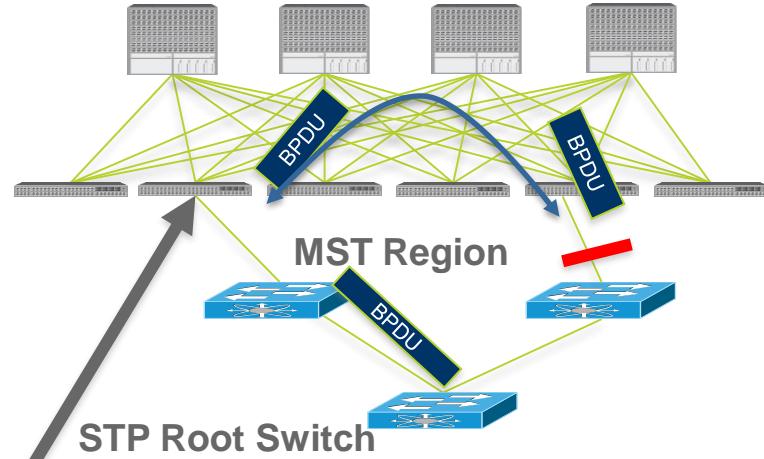
- BPDU for PVST and RPVST has VLAN tagging and will be flooded within the EPG for the VLAN. BPDU is flooded when the EPG is configured for data traffic.
- BPDU frame for MST is sent over the native VLAN and doesn't have VLAN tagging
- By default ACI leaf port is not in VLAN 1 and VLAN 1 is not native VLAN
 - Different behavior than traditional switch
- On the leaf ports that connect to legacy network running MST user need to explicitly configure EPG for an untagged VLAN and assign the ports to this EPG. Without this, BPDU frame for MST won't be flooded although data forwarding works
- Typically done with a dedicated EPG



ACI Interaction with MST

Enable untagged VLAN on ACI Leaf Port

- The configuration enables untagged vlan on interface eth1/3 of leaf node 101 and 103
- Choose mode “802.1p” (or untagged) to indicates leaf expects untagged frame and will assign them to vlan 100 (in Aci you can pick whatever vlan you want)
- You may pick up another vlan than 1 as far as this vlan is untagged from ACI Point of view



Path	Primary VLAN for Micro-Seg	Port Encap (or Secondary VLAN for Micro-Seg)	Deployment Immediacy	Mode
Node: Pod-1				
Pod-1/Node-101/eth1/3	unknown	vlan-100	On Demand	Access (Untagged)
Pod-1/Node-103/eth1/3	unknown	vlan-100	On Demand	Access (Untagged)

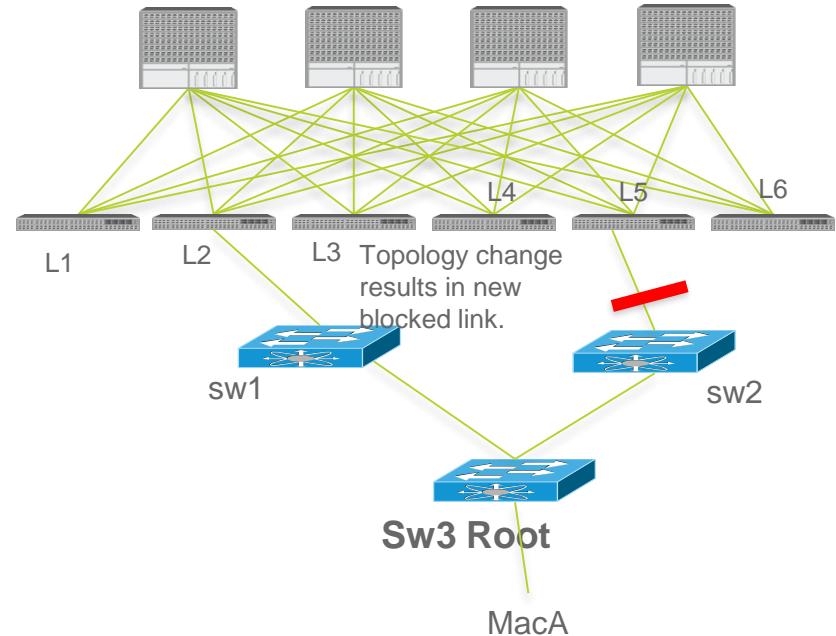
BPDUs forwarding detail

- BPDUs are flooded
 - to Mcast GIPo of the BD (outer L3 IP address)
 - Encapsulated in FD_VLAN VNID of the access encap of the EPG
- Based on FTAG/GIPo tree, each leaf where the BD is does receive the BPDUs
- If there are no FD_VLAN with the same VNID (same access encap) on the leaf, the BPDUs are dropped (MET empty)

ACI Interaction with STP in Legacy Network

STP TCN Snooping

- Fabric intercept the BPDU TCN frame
- APIC flushes the MAC address for the corresponding EPG that has the STP topology change
- Bridge domain flooding vs. Convergence time with TCN.
- Can be seen in epmc-trace
- With MSTP user need to configure instance to VLAN mapping so APIC knows for what EPGs it need to flush the MAC
- **Recommend to have vPC connection to legacy switches to minimize the TCN**



Example :

Mac A is behing Leaf2 , if we lose link between Sw1 and Sw3, we need to unblock Sw2 to L5
→ ACI can't keep track of MAC A behind Leaf 2!

ACI Interaction with MST in Legacy Network

TCN Snooping

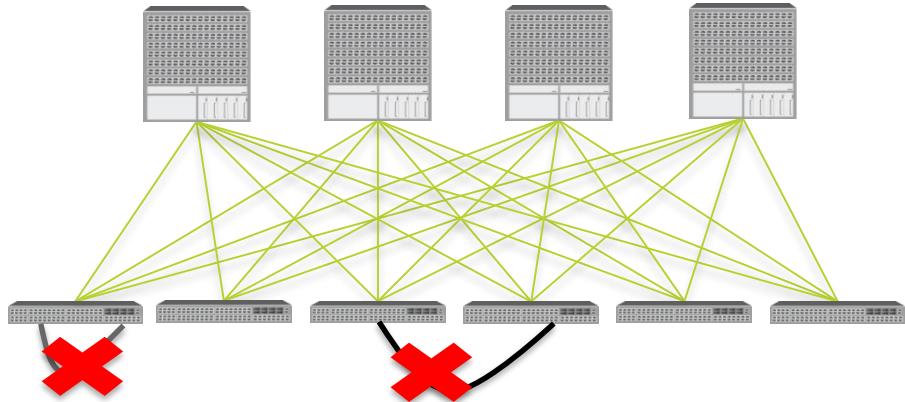
- For PVST and RPVST, VLAN tagging in TCN frame indicates the VLAN that has topology, APIC flush the end points for the corresponding EPG for the VLAN
- For MST, TCN frame indicates instances that has topology change. The instance to VLAN mapping has to be configured on APIC in order for APIC to flush end points for the correct encap VLANs.

The screenshot shows the Cisco ACI Fabric interface. The top navigation bar includes System, Tenants, Fabric (selected), Virtual Networking, L4-L7 Services, Admin, Operations, and Apps. Below the navigation bar is a secondary menu with Inventory, Fabric Policies, and Access Policies. The main content area is titled "Spanning Tree Domain Policy - inst1". On the left, there is a sidebar with icons for Quick Start, Switches, Modules, Interfaces, Policies (selected), Switch, Spanning Tree (selected), default, and MST. A list at the bottom of the sidebar shows entries for inst1 and inst2. The main panel displays the properties for the MST instance "inst1":

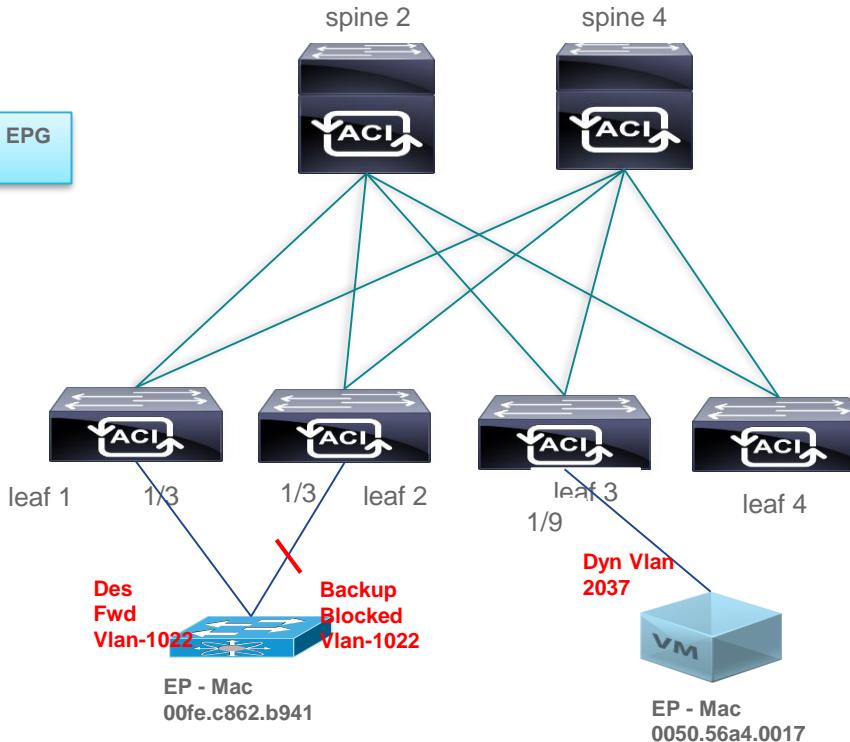
- Name: inst1
- Description: optional
- MST Instance: 1
- Encap Blocks:
 - Type: VLAN
 - ID Range: [101-200]

ACI Leaf Local Loop Detection

- ACI Fabric doesn't generate STP BPDU frame
- Loop between two leaf switch ports are blocked by cable plant verification (LLDP)
 - Non-fabric ports are not allowed to connect to each other on leaf



Lab setup



A single EPG in BD

Two static path :

Leaf 1 and leaf 2 vlan 1022 going to legacy network running RPVST+

One Dyn Path :

Leaf 3 on 1/9 going to UCS hosting VM in dyn vlan 2037

Spanning-tree view from External switch

```
bdsol-aci32-n3k-1# show spanning-tree vlan 1022
```

VLAN1022

Spanning tree enabled protocol rstp

Root ID Priority 33790
Address 00fe.c862.b941
This bridge is the root
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Bridge ID Priority 33790 (priority 32768 sys-id-ext 1022)
Address 00fe.c862.b941
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Interface	Role	Sts	Cost	Prio.Nbr	Type
-----------	------	-----	------	----------	------

Eth1/1	Desg	FWD	2	128.1	P2p
Eth1/2	Back	BLK	2	128.5	P2p
Eth1/3	Desg	FWD	4	128.9	P2p

Dest Fwd port going to ACI leaf1

Backup port blocked by our own BPDU forwarding through leaf 2

Port dest fwd going to another L2 switch

FD_VLAN on each leaf

Leaf1 only have encap 1022=0x3fe with VNID 8313 = 0x2079

```
bdsol-aci32-leaf1# vsh_lc -c 'show system internal eltmc info vlan brief' | egrep "==|Vlan|Type| 1022 | 2037 "
VlanId  HW_VlanId Type          Access_enc Access_enc Fabric_enc Fabric_enc BDVlan
                    Type          Type
=====
 36      38        FD_VLAN    802.1q    1022      VXLAN    8313      35
```

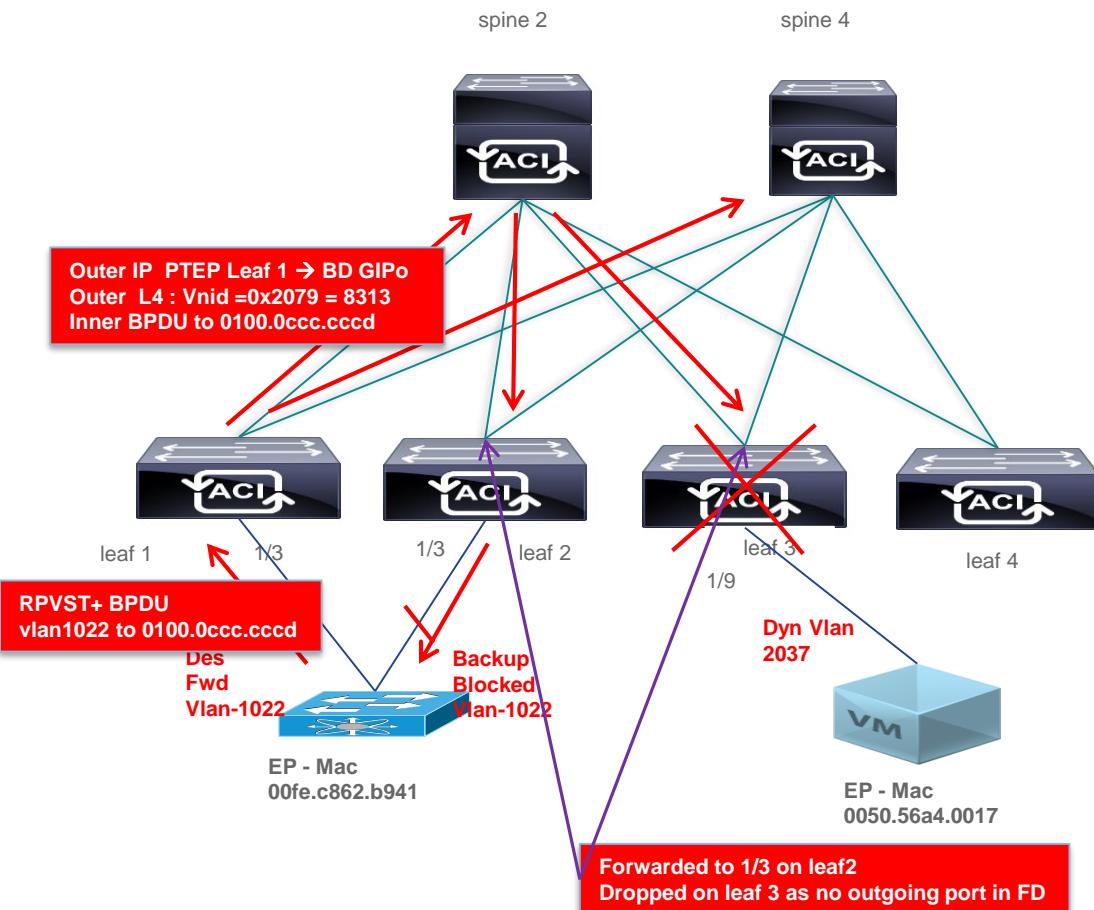
Leaf2 same as leaf 1, only PI VlanID and HwVlan Differs

```
bdsol-aci32-leaf2# vsh_lc -c 'show system internal eltmc info vlan brief' | egrep "==|Vlan|Type| 1022 | 2037 "
VlanId  HW_VlanId Type          Access_enc Access_enc Fabric_enc Fabric_enc BDVlan
                    Type          Type
=====
 30      32        FD_VLAN    802.1q    1022      VXLAN    8313      29
```

Leaf3 do not know the static vlan 1022 , but does have the dyn 2037 part of same epg but diff encaps

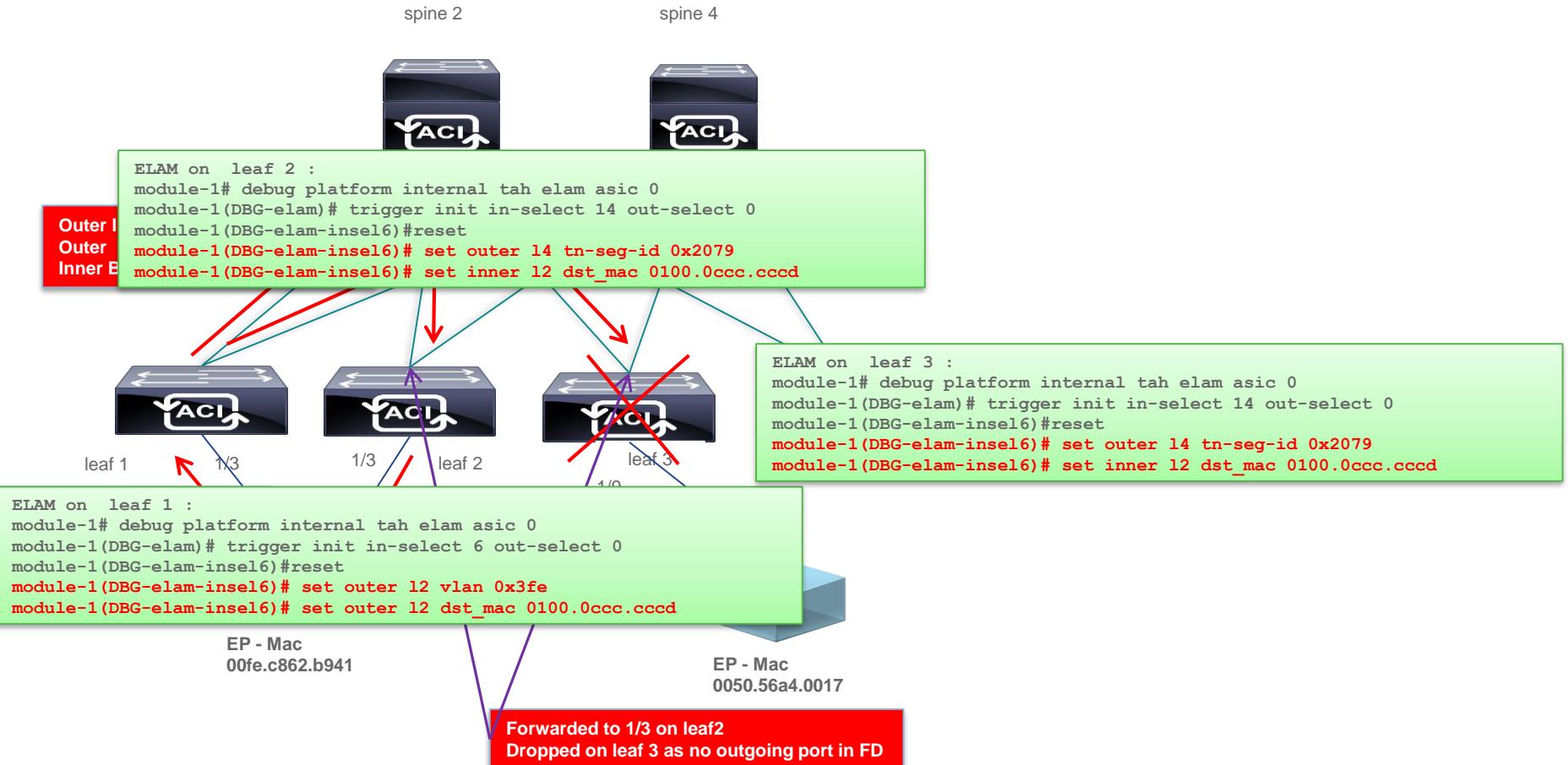
```
bdsol-aci32-leaf3# vsh_lc -c 'show system internal eltmc info vlan brief' | egrep "==|Vlan|Type| 1022 | 2037 "
VlanId  HW_VlanId Type          Access_enc Access_enc Fabric_enc Fabric_enc BDVlan
                    Type          Type
=====
 38      40        FD_VLAN    802.1q    2037      VXLAN    8429      37
bdsol-aci32-leaf3#
```

BPDU – Flow

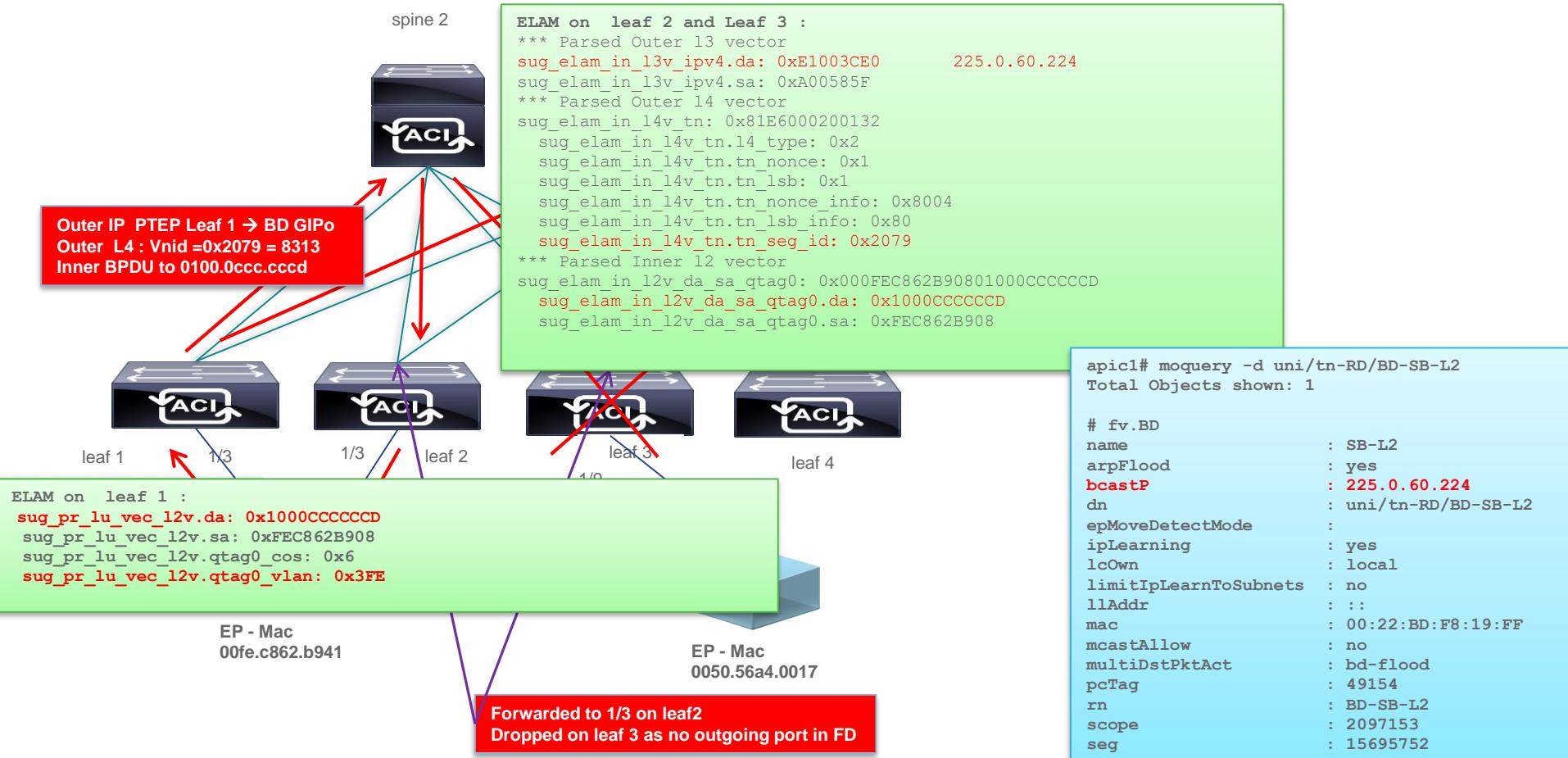


1. BPDU is send to Des Port towards leaf 1
2. Leaf encap to BD GIPo and vnid of FD_VLAN and floods it to BD GIPo topology
3. Leaf 2 received it and decaps to send on 1/3. Receiving L2 switch recognise its own BPDU and blocks as backup port (not alternate)
4. Leaf 3 drops it

BPDU – Flow – ELAM Trigger on each leaf



BPDU – Flow – ELAM data field



Epmc-trace when a tcn is received

Flush happens in leaf 1 and 2 that have access encap 1022,
Not on leaf 3

```
bdsol-aci32-leaf1# tail -f /var/sysmgr/tmp_logs/epmc-trace.txt | egrep "lush|MST"
[2017 Apr 21 14:14:02.574896578:233627:epmc_walk_multi_vlan_multi_port:699:t] ifidx 0 flush type 4
[2017 Apr 21 14:14:02.574900338:233628:epmc_walk_multi_vlan_multi_port:747:t] mp mv flush for if index 0,
vlan 36
[2017 Apr 21 14:14:02.574901498:233629:epmc_ep_flush_vlan:752:t] Flushing all EPs in vlan 36
[2017 Apr 21 14:14:02.574948729:233638:epmc_ep_xr_flush_vlan_bd:824:t] Flushing all XR-EPs in vlan 36
[2017 Apr 21 14:14:02.576064562:233649:epmc_handle_multi_port_multi_vlan_flush:840:t] multi port multi
vlan rsp to tvl RV:0 tlv_size:12 NBF:12
```

```
bdsol-aci32-leaf2# tail -f /var/sysmgr/tmp_logs/epmc-trace.txt | egrep "lush|MST"
[2017 Apr 21 14:14:02.574731054:837257:epmc_walk_multi_vlan_multi_port:699:t] ifidx 0 flush type 4
[2017 Apr 21 14:14:02.574734649:837258:epmc_walk_multi_vlan_multi_port:747:t] mp mv flush for if index 0,
vlan 30
[2017 Apr 21 14:14:02.574736132:837259:epmc_ep_flush_vlan:754:t] Flushing all EPs in vlan 30
[2017 Apr 21 14:14:02.574782923:837267:epmc_ep_xr_flush_vlan_bd:826:t] Flushing all XR-EPs in vlan 30
[2017 Apr 21 14:14:02.576328145:837278:epmc_handle_multi_port_multi_vlan_flush:840:t] multi port multi
vlan rsp to tvl RV:0 tlv_size:12 NBF:12
```

```
bdsol-aci32-leaf3# tail -f /var/sysmgr/tmp_logs/epmc-trace.txt | egrep "lush|MST"
```

Troubleshooting MCP CLI Sample Output

- On interface level, we have following commands to show the internal info:
 - show mcp internal info interface [all] <if_name>]

```
bdsol-aci32-leaf1# show mcp internal info interface ethernet 1/3
-----
Interface: Ethernet1/3
    Native PI VLAN: 0
    Native Encap VLAN: 0
        BPDU Guard: disabled
        BPDU Filter: disabled
        Port State: up
        Layer3 Port: false
        Switching State: enabled
        Mac Address: e0:e:da:a2:f1:e5
Interface MCP disabled: MCP is disabled globally
----- STP STATS -----
    MSTP Count: 0
    RSTP Count: 0
    MSTP TC Count: 0
    RSTP TC Count: 0
    PVRSTP TC Count: 74
    TCN Count: 0
    PVID Error BPDU Count: 0
    Error Packet Count: 0
    BPDU Guard Event Count: 0
Last TC received at Fri Apr 21 14:14:02 2017
```

Troubleshooting MCP CLI Sample Output

- On VLAN level, we support the following commands:
 - show mcp internal info vlan [all]<encap_vlan>

```
bdsol-aci32-leaf1# show mcp internal info vlan 1022
-----
      PI VLAN: 36 Up
      Encap VLAN: 1022
PVRSTP TC Count: 28
      RSTP TC Count: 0
Last TC flush at Fri Apr 21 14:14:00 2017
      on Ethernet1/3
```

Trigger ELAM of BPDU

- Use Dest MAC always
 - For spine to leaf or leaf to spine – add outer I4 vnid (FD_VLAN_VNID) field in gen-2 or gen-3 is similar to **sug_elam_in_14v_tn.tn_seg_id**
 - Set outer I3 tn-seg-id 0x..
 - Set inner I2 dst_mac
 - For front panel BPDU - add dot1q encap vlan and eventually port
 - Set srcid ...
 - Set outer I2 dst_mac
 - Set outer I2 vlan 0x...

TCPDUMP on BPDU

- Seems we can capture bpdu with tcpdump on leaf (at least on gen-2)

```
bdsol-aci32-leaf1# tcpdump -xxvvn1 kpm_inb stp
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
10:33:14.870756 STP 802.1w, Rapid STP, Flags [Proposal], bridge-id 8001.24:e9:b3:9a:cd:c1.8099, length 43
    message-age 1.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15.00s
    root-id 8001.00:fe:c8:62:b9:41, root-pathcost 2, port-role Designated
    0x0000: 0180 c200 0000 24e9 b39a cda0 0027 4242
    0x0010: 0300 0002 020e 8001 00fe c862 b941 0000
    0x0020: 0002 8001 24e9 b39a cdc1 8099 0100 1400
    0x0030: 0200 0f00 0000 0000 0000 0000

10:33:14.871086 STP 802.1w, Rapid STP, Flags [Proposal], bridge-id 8001.24:e9:b3:9a:cd:c1.808d, length 43
    message-age 1.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15.00s
    root-id 8001.00:fe:c8:62:b9:41, root-pathcost 2, port-role Designated
    0x0000: 0180 c200 0000 24e9 b39a cd94 0027 4242
    0x0010: 0300 0002 020e 8001 00fe c862 b941 0000
    0x0020: 0002 8001 24e9 b39a cdc1 808d 0100 1400
```

Tracking source of BPDU on ACI leaf

- Mctp traces shows BPDU or TC BPDU receive.
- Assume we see this bpdu, how can we find out where it comes from ?
- Let's check Vlan (68) and Phy_if_index

```
bdsol-aci32-leaf4# show mcp internal event-history trace detail  
17) Event: E_DEBUG at 609262 usecs after Tue Jan  8 12:51:20 2019  
    mcp_istack_rcvd_stp_bpdu:1500: Received BPDU on logic if_index 0x901003d phy if_index 0x18010008,  
    pdu_len 36, vlan 68, type: 2, version: 2, encaps type: IEEE
```

Finding src interface and vlan

```
bdsol-aci32-leaf4# show system internal epm interface all | egrep 0x18010008  
Tunnel8          0x18010008 UP      No  10.0.80.94        0000.0000.0000 0
```

```
bdsol-aci32-leaf4# show system internal epm vlan 68  
Warning: could not get list of reserved vlans
```

VLAN ID	Type	Access Encap (Type Value)	Fabric Encap	H/W id	BD VLAN	Endpoint Count
68	FD	vlan 802.1Q	1013	8304	72	61

So that's a BPDU travelling across the fabric received from leaf with TEP 10.0.80.94 in encap vlan 1013.

Next : to do same on leaf with that src TEP 😊

STP and ACI : What shall I remember from STP

STP – Where shall I see BPDU send ?

- Legacy 802.1d -> BPDU only in one direction
- RSTP → BPDU mostly in one direction . But upstream BPDU can/will occur (during Topol change, during convergence proposal/agreement)
- MST → likely in both direction (depending on instances and vlan on the trunk)
- Some feature will change it : Bridge assurance have always BiDir BPDU.
- Otherwise expected direction from send only on Designated port

What is BPDU destination Mac address

- IEEE Mac for BPDU is 01:80:C2:00:00:00. It is used
 - On non trunk – always
 - With MST – Always
 - On trunk running (R)PVST+ : only in native vlan
- Cisco MAC for BPDU **01:00:0C:CC:CC:CD**. It is used
 - In every VLAN on trunk running R(PVST)+

What is source Mac if BPDU

- It is NOT the MAC of Bridge ID
- It is not the MAC of the SVI of the sending bridge
- It is the Mac address of the L2 switchport (Ethernet controller) can originate the bpdu. Typically in cisco switches can be figured out from show module
- ACI (like any other switches) DO NOT LEARN MAC from BPDU. So this mac will not appear anywhere

ACI interactions with Catalyst Etherchannel guard misconfig

- Most Catalyst runs by default a feature called etherchannel guard on port-channel
- On a port-channel we expect to receive/send a single BPDU per instances/vlan . Not one per port of the channel
- Etherchannel guard monitor src mac (not sender BID mac) of all BPDU receive on a port-channel if it receive X BPDU from 2 different src it assumes the upstream devices is not properly configured for etherchannel and disable the port

ACI interactions with Catalyst Etherchannel guard misconfig (cont.)

- ACI behind a shared media for STP point of view it is not unusual multiple BPDU will be received per vlan/instances on a Port-channel
- Hence this functionality MUST be disable is connecting catalyst switches to ACI
- If it is enable problem may not show up directly (depending of BPDU direction and number of switches attached to the EPG)
- In any case the feature is useless on port-channel running LACP .

Avoid TCN with ACI

- TCN on ACI leads to flushing EP table like regular switch, however impact on ACI may be higher due to possible proxy nature of ACI (no flood to relearn)
1. If it is mandatory to get STP across ACI, try reducing/sanatizing legacy STP network before connecting to ACI
 2. Keep L2 extended BD in flood mode to avoid proxy in those vlan.

Why do we recommend to avoid overlapping vlan space

- Some packets (mostly STP BPDU) are forwarded in ACI fabric as vxlan packet with VNID of their FD_VLAN (ext encap vlan). This is done to avoid spanning other encap vlan about STP info
- FD_VLAN VNID directly depends of the physical domain from which the vlan is taken from
- If in the same EPG you have two Phys domain and leaf1 gets its encap for vlan X from Dom1 and leaf2 gets its encap for same vlan X from Dom2 we would end up with BPDU not flooded between those 2 leaf → Risk of STP loop !

Overlapping name space

Vlan pool Pool1 uses base 9662 for vlan 2100
Vlan pool Pool2 uses base 9592 for vlan 2100

```
admin@pod2-apic1:~> moquery -c stpAllocEncapBlkDef -f
'stp.AllocEncapBlkDef.encapBlk == "uni/infra/vlanns-
[Pool1]-static/from-[vlan-2100]-to-[vlan-2299]"'
Total Objects shown: 1

# stp.AllocEncapBlkDef
encapBlk      : uni/infra/vlanns-[Pool1]-static/from-
[vlan-2100]-to-[vlan-2299]
base        : 9692
childAction   :
descr         :
dn            : allocencap-[uni/infra]/encapnsdef-
[uni/infra/vlanns-[Pool1]-static]/allocencapblkdef-
[uni/infra/vlanns-[Pool1]-static/from-[vlan-2100]-to-
[vlan-2299]]
from          : vlan-2100
lcOwn         : local
modTs         : 2016-06-09T13:04:26.677+02:00
monPolDn     : uni/fabric/monfab-default
name          :
ownerKey      :
ownerTag      :
rn             : allocencapblkdef-[uni/infra/vlanns-
[Pool1]-static/from-[vlan-2100]-to-[vlan-2299]]
status         :
to             : vlan-2299
```

```
admin@pod2-apic1:~> moquery -c stpAllocEncapBlkDef -f
'stp.AllocEncapBlkDef.encapBlk == "uni/infra/vlanns-
[Pool2]-static/from-[vlan-2100]-to-[vlan-2199]"'
Total Objects shown: 1

# stp.AllocEncapBlkDef
encapBlk      : uni/infra/vlanns-[Pool2]-static/from-
[vlan-2100]-to-[vlan-2199]
base        : 9592
childAction   :
descr         :
dn            : allocencap-[uni/infra]/encapnsdef-
[uni/infra/vlanns-[Pool2]-static]/allocencapblkdef-
[uni/infra/vlanns-[Pool2]-static/from-[vlan-2100]-to-
[vlan-2199]]
from          : vlan-2100
lcOwn         : local
modTs         : 2016-06-09T10:57:28.214+02:00
monPolDn     : uni/fabric/monfab-default
name          :
ownerKey      :
ownerTag      :
rn             : allocencapblkdef-[uni/infra/vlanns-
[pool2]-static/from-[vlan-2100]-to-[vlan-2199]]
status         :
to             : vlan-2199
```

Preventing overlapping vlan pool by default (4.1)

- CSCvn93839: UI: include infraSetPol.validateOverlappingVlans in system settings
- CSCvj84175 Enhancement: Knob to prevent overlapping VLAN pools associated to domain

```
admin@apic1:~> moquery -c infraSetPol
Total Objects shown: 1

# infra.SetPol
annotation          :
childAction         :
descr               :
dn                  : uni/infra/settings
domainValidation   : no
enableRemoteLeafDirect : no
enforceSubnetCheck : no
extMngdBy          :
lcOwn               : local
modTs               : 2018-12-23T06:13:00.915+00:00
name                : default
nameAlias           :
opflexpAuthenticateClients : no
opflexpUseSsl       : yes
ownerKey             :
ownerTag             :
reallocategipo      : no
rn                  : settings
status               :
uid                 : 0
unicastXrEpLearnDisable : no
validateOverlappingVlans : no    → set it to yes to prevent
overlapping gVLAN
```

iPING

iPing CLI

Hint: To check list of VRF names:
show vrf

usage:

```
iping [-V vrf] [-c count] [-i wait] [-p pattern] [-s packetsize] [-t timeout] [-S source ip] host
```

options:

- V : vrf to use for ping (management/overlay-1/Tenant VRF)
- c : # of requests to send.
- i : interval between ICMP echo packets.
- t : Timeout for responses.
- p : Data pattern in payload.
- s : Size
- S : Source – Interface name/ IP address.

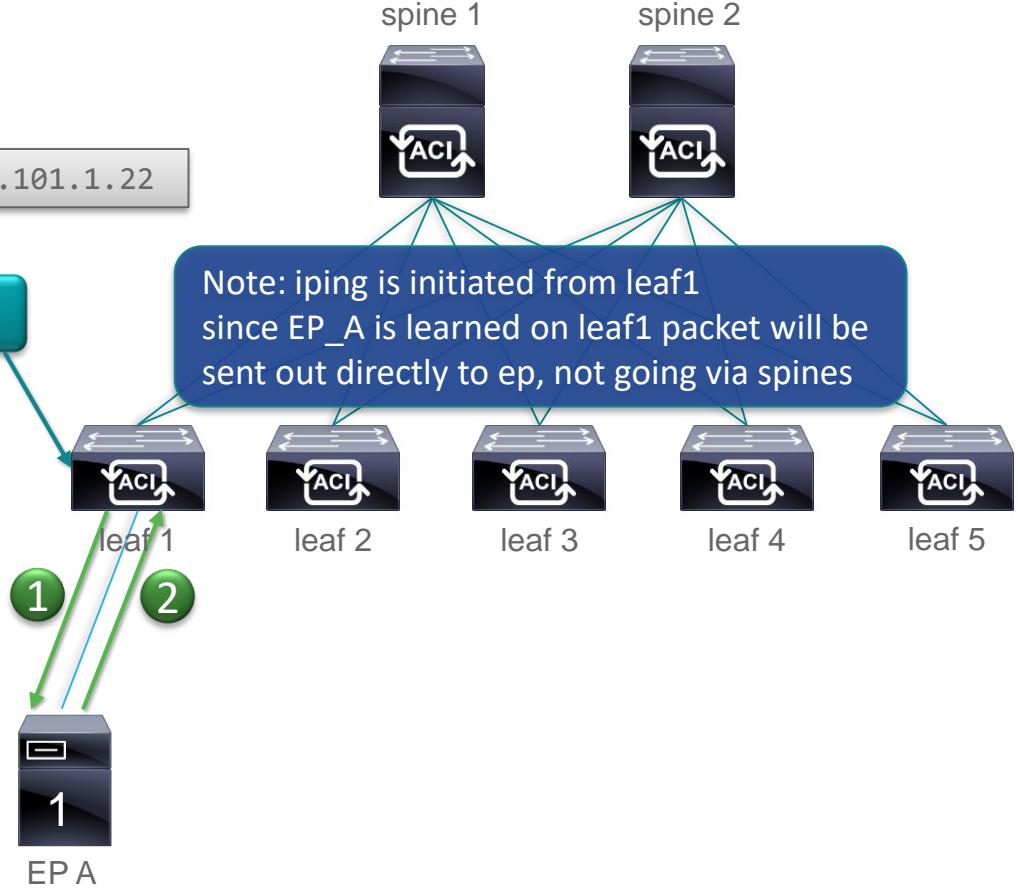
iping internals

```
leaf1# iping -V tenant:vrf01 -S 64.101.1.1 64.101.1.22
```

Recommended: set the source IP address desired GW (BD IP)

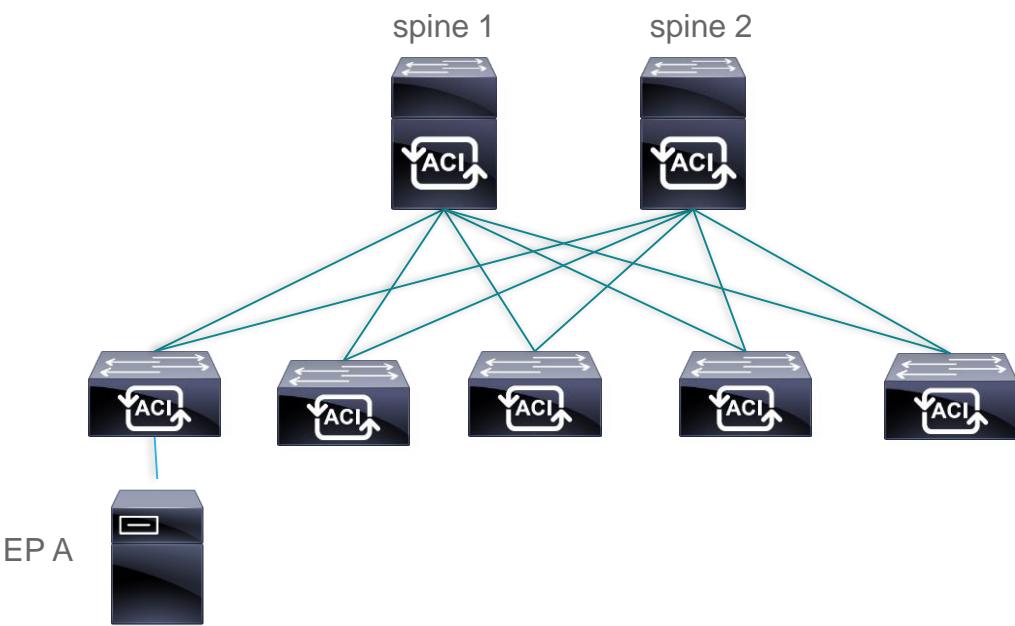
- 1 leaf1: iping to Endpoint_A (EP_A)
- 2 EP_A (.22): responds to leaf1

Note: iping is initiated from leaf1 since EP_A is learned on leaf1 packet will be sent out directly to ep, not going via spines



iPING use case

- Assuming I already know ICMP echo are not reaching EP A (tcpdump on EP A for example)
- But I want to doublecheck whether the icmp ECHO are leaving the leaf or not
- Send fast rate ping and check stats



```
pod2-leaf1# show interface ethernet 1/33 | egrep -A 1 "TX"
 TX
 279767 unicast packets  661013 multicast packets  39478 broadcast packets
pod2-leaf1# iping -V Interctx:ctxcons 10.11.1.102 -t 0 -i 0 -c 10000
pod2-leaf1# iping -V Interctx:ctxcons 10.11.1.102 -t 0 -i 0
PING 10.11.1.102 (10.11.1.102) from 10.11.1.1: 56 data bytes
Request 0 timed out
Request 1 timed out
```

```
pod2-leaf1# show interface ethernet 1/33 | egrep -A 1 "TX"
 TX
 289779 unicast packets  661023 multicast packets  39479 broadcast packets
pod2-leaf1#
```

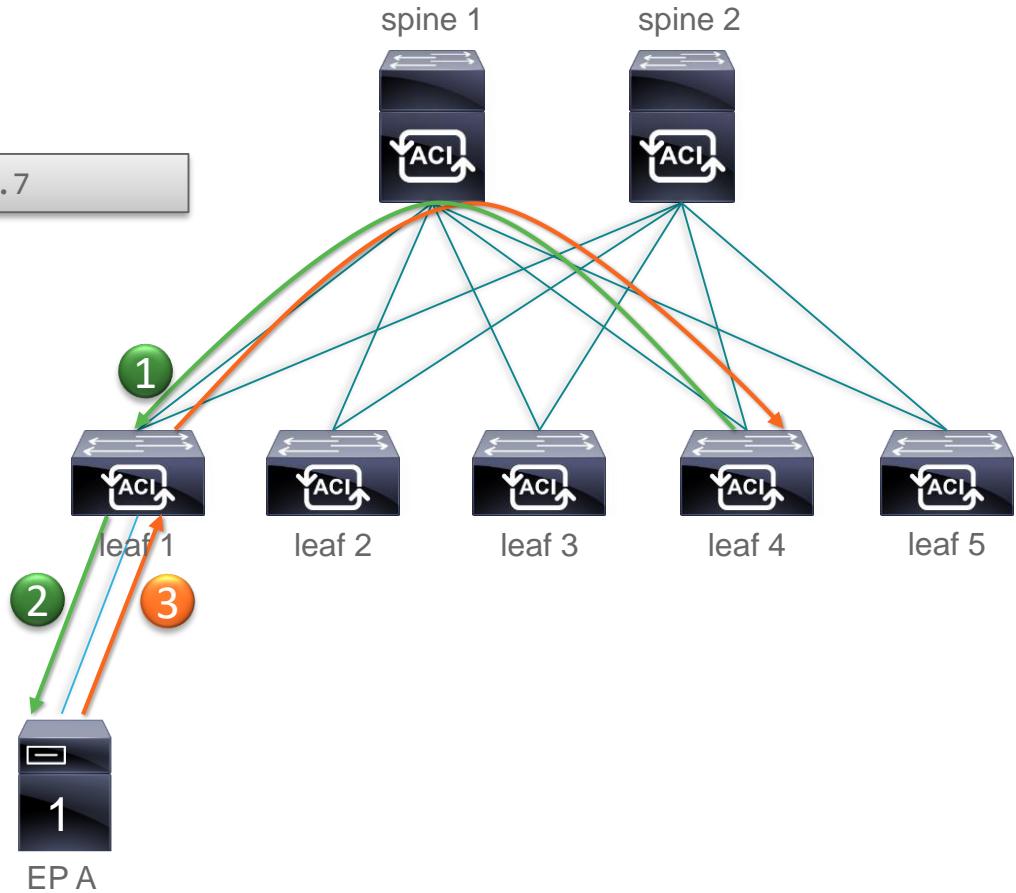
Likely a server issue
(teaming, ...)
Or sth in the middle
(blade switch..)

BD subnet : 7.7.7.254/24

iping internals

```
leaf4# iping -V RD-OSPF:RD -S 7.7.7.254 7.7.7.7
```

- 1 leaf4: iping to Endpoint_A 7.7.7.7
(icmp echo request to leaf1 TEP)
- 2 leaf1: ping is switched to EP_A
- 3 EP_A (7.7.7.7): responds to BD subnet 7.7.7.254



Iping from leaf 4 is successful

```
bdsol-aci32-leaf4# iping -V RD-OSPF:RD 7.7.7.7 -c 1000
PING 7.7.7.7 (7.7.7.7) from 7.7.7.254: 56 data bytes
64 bytes from 7.7.7.7: icmp_seq=0 ttl=255 time=0.744 ms
64 bytes from 7.7.7.7: icmp_seq=1 ttl=255 time=0.678 ms
64 bytes from 7.7.7.7: icmp_seq=2 ttl=255 time=0.728 ms
64 bytes from 7.7.7.7: icmp_seq=3 ttl=255 time=0.689 ms
64 bytes from 7.7.7.7: icmp_seq=4 ttl=255 time=0.633 ms
64 bytes from 7.7.7.7: icmp_seq=5 ttl=255 time=0.674 ms
```

Tcpdump –xxvvi kpm_inb icmp

Leaf 4 only sees the ECHO REQUEST

```
Leaf 4 only sends ECHO  
15:27:47.357594 IP (tos 0x0, ttl 65, id 17393, offset 0, flags [none],  
proto ICMP (1), length 84)  
    7.7.7.254 > 7.7.7.7: ICMP echo request, id 10241, seq 11008, length  
64  
        0x0000: 0000 0000 0000 0000 0000 0000 0800 4500  
        0x0010: 0054 43f1 0000 4101 18a6 0707 07fe 0707  
        0x0020: 0707 0800 a545 2801 2b00 73f7 695a 9d74  
        0x0030: 0500 daab acab 0a00 585a e3dd caac acab  
        0x0040: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0050: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0060: cdab  
15:27:49.358618 IP (tos 0x0, ttl 65, id 17581, offset 0, flags [none],  
proto ICMP (1), length 84)  
    7.7.7.254 > 7.7.7.7: ICMP echo request, id 10241, seq 11264, length  
64  
        0x0000: 0000 0000 0000 0000 0000 0000 0800 4500  
        0x0010: 0054 44ad 0000 4101 17ea 0707 07fe 0707  
        0x0020: 0707 0800 a041 2801 2c00 75f7 695a 9f78  
        0x0030: 0500 daab acab 0a00 585a e3dd caac acab  
        0x0040: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0050: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0060: cdab
```

Leaf 1 only sees the ECHO REPLY

```
15:28:07.377074 IP (tos 0x0, ttl 255, id 19392, offset 0, flags  
[none], proto ICMP (1), length 84)  
    7.7.7.7 > 7.7.7.254: ICMP echo reply, id 10241, seq 13568,  
length 64  
        0x0000: 0022 bdf8 19ff 24e9 b39a cdc1 0800 4500  
        0x0010: 0054 4bc0 0000 ff01 52d6 0707 0707 0707  
        0x0020: 07fe 0000 a7fa 2801 3500 87f7 695a 84bf  
        0x0030: 0500 daab acab 0a00 585a e3dd caac acab  
        0x0040: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0050: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0060: cdab  
15:28:09.380972 IP (tos 0x0, ttl 255, id 19393, offset 0, flags  
[none], proto ICMP (1), length 84)  
    7.7.7.7 > 7.7.7.254: ICMP echo reply, id 10241, seq 13824,  
length 64  
        0x0000: 0022 bdf8 19ff 24e9 b39a cdc1 0800 4500  
        0x0010: 0054 4bc1 0000 ff01 52d5 0707 0707 0707  
        0x0020: 07fe 0000 35f3 2801 3600 89f7 695a f3c6  
        0x0030: 0500 daab acab 0a00 585a e3dd caac acab  
        0x0040: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0050: cdab 0000 cdab 0000 cdab 0000 cdab 0000  
        0x0060: cdab
```

Tcpdump on leaf with no icmp filter !!

```
15:29:10.945228 IP (tos 0x40, ttl 63, id 49594, offset 0, flags [none], proto UDP (17), length 112)
bdsol-aci32-leaf1.65125 > bdsol-aci32-leaf4.33334: [udp sum ok] UDP, length 84
0x0000: 000c 0c0c 0c0c 000d 0d0d 0d0d 0800 4540
0x0010: 0070 c1ba 0000 3f11 f4c9 0a00 585f 0a00
0x0020: 585a fe65 8236 005c b9e0 4500 0054 70e9
0x0030: 0000 ff01 2dad 0707 0707 0707 07fe 0000      - IP 7.7.7.7 to 7.7.7.254
0x0040: 1f3f 8201 f500 c6f7 695a 0d21 0e00 daab
0x0050: acab 0a00 585a 8236 caac acab cdab 0000
0x0060: cdab 0000 cdab 0000 cdab 0000 cdab 0000
0x0070: cdab 0000 cdab 0000 cdab 0000 cdab
```

Leaf 1 relay reply to Leaf 4 as UDP Packet !!!
ICMP is in payload ?

How does leaf 4 knows what to do ?

Let's look at the echo request from leaf 4 again

```
15:27:47.357594 IP (tos 0x0, ttl 65, id 17393, offset 0, flags [none], proto ICMP (1), length 84)
 7.7.7.254 > 7.7.7.7: ICMP echo request, id 10241, seq 11008, length 64
    0x0000: 0000 0000 0000 0000 0000 0000 0800 4500  beginning of ip header
    0x0010: 0054 43f1 0000 4101 18a6 0707 07fe 0707  7.7.7.254 to 7.7.7.7
    0x0020: 0707 0800 a545 2801 2b00 73f7 695a 9d74
    0x0030: 0500 daab acab 0a00 585a e3dd caac acab  10.0.88.90 in payload
    0x0040: cdab 0000 cdab 0000 cdab 0000 cdab 0000
    0x0050: cdab 0000 cdab 0000 cdab 0000 cdab 0000
    0x0060: cdab
```

```
bdsol-aci32-leaf4# acidiag fnvread | egrep leaf4
 104      1      bdsol-aci32-leaf4      FDO20230UST      10.0.88.90/32      leaf      active      0
bdsol-aci
```

Every iping request contains the Initiating Leaf TEP in the payload !!

No.	Time	Source	Destination	Protocol	Length	Info
151	3.804693	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=4096/16, ttl=65 (no response found!)
152	3.805106	10.0.88.95	10.0.88.90	UDP	126	65125 → 36613 Len=84
153	3.805230	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=4352/17, ttl=65 (no response found!)
154	3.805658	10.0.88.95	10.0.88.90	UDP	126	65125 → 36613 Len=84
155	3.805766	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=4608/18, ttl=65 (no response found!)
156	3.806225	10.0.88.95	10.0.88.90	UDP	126	65125 → 36613 Len=84
157	3.806347	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=4864/19, ttl=65 (no response found!)
158	3.824337	10.0.88.95	10.0.88.90	UDP	126	65125 → 36613 Len=84
159	3.824474	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=5120/20, ttl=65 (no response found!)
160	3.824929	10.0.88.95	10.0.88.90	UDP	126	65125 → 36613 Len=84
161	3.825059	7.7.7.254	7.7.7.7	ICMP	98	Echo (ping) request id=0x3c01, seq=5376/21, ttl=65 (no response found!)
162	3.830783	10.0.0.3	10.0.88.90	TCP	74	40404 → 22 [SYN] Seq=0 Win=29120 Len=0 MSS=1456 SACK_PERM=1 TSval=560869050 TSecr=0 WS=128
163	3.830848	10.0.88.90	10.0.0.3	TCP	74	22 → 40404 [SYN, ACK] Seq=0 Ack=1 Win=27744 Len=0 MSS=9260 SACK_PERM=1 TSval=134117214 TSecr=560869050 WS=1024
164	3.830959	10.0.0.3	10.0.88.90	TCP	66	40404 → 22 [ACK] Seq=1 Ack=1 Win=29184 Len=0 TSval=560869051 TSecr=134117214
165	3.831142	10.0.0.3	10.0.88.90	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_7.3)
166	3.831174	10.0.88.90	10.0.0.3	TCP	66	22 → 40404 [ACK] Seq=1 Ack=2 Win=29672 Len=0 TSval=134117214 TSecr=560869051

> Frame 160: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)

> Ethernet II, Src: Itsupport_0d:0d:0d (00:0d:0d:0d:0d:0d), Dst: ApproTec_0c:0c:0c (00:0c:0c:0c:0c:0c)

▀ Internet Protocol Version 4, Src: 10.0.88.95, Dst: 10.0.88.90

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x40 (DSCP: CS2, ECN: Not-ECT)
Total Length: 112
Identification: 0xe437 (58423)
> Flags: 0x00
Fragment offset: 0
Time to live: 63
Protocol: UDP (17)
Header checksum: 0xd24c [validation disabled]
[Header checksum status: Unverified]
Source: 10.0.88.95
Destination: 10.0.88.90
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
```

▀ User Datagram Protocol, Src Port: 65125, Dst Port: 36613
Source Port: 65125
Destination Port: 36613
Length: 92
Checksum: 0xad11 [unverified]
[Checksum Status: Unverified]
[Stream index: 1]

▀ Data (84 bytes)
Data: 450000545a030000ff0144930707070707fe00004d30...
[Length: 84]

Conclusion

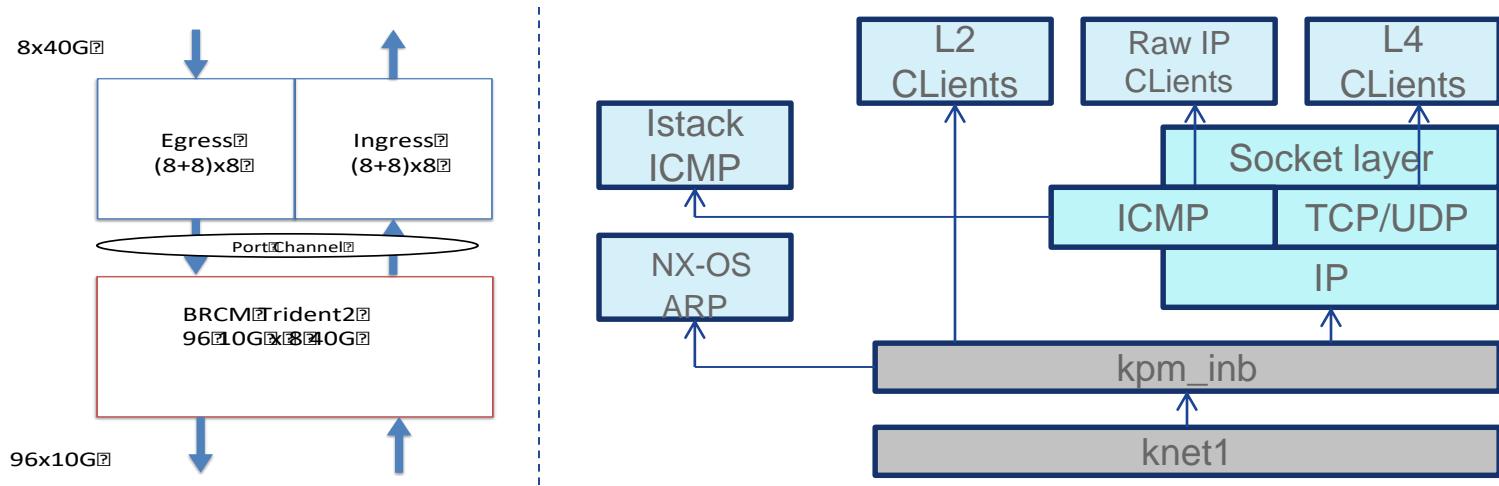
- Understand Iping, it is a tool and do not mimic real flow from EP to EP
- In this scenario you will never catch icmp reply on leaf4 and it is expected !!!
- Sending leaf encode its PTEP in request and Rx leaf relay it back

Tcpdump for control Plane troubleshooting

TCPDUMP

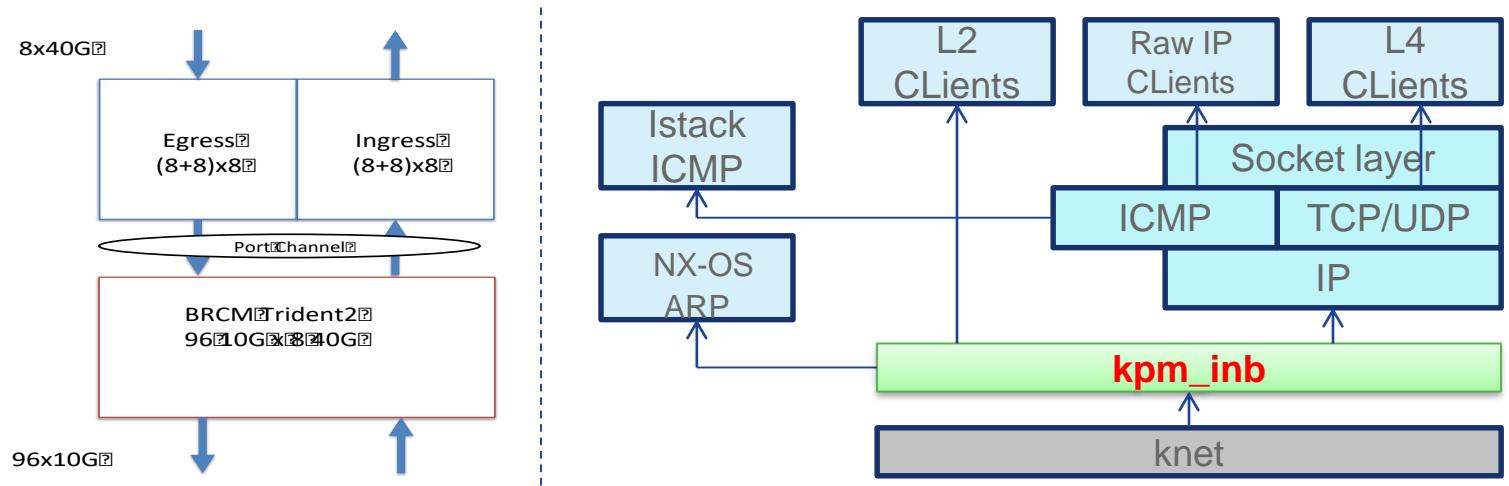
- Each leaf spine is fully linux system.
- Support TCPDUMP
- Only usefull for packet reaching CPU (arp of pervasive gateway, ntp, snmp,...)
- Which Interface should we use ?
 - For most inband traffic use **kpm_inb** – note for arp for example kpm_inb only shows traffic in one direction
 - For oob traffic use eth0

Inband path on leaf



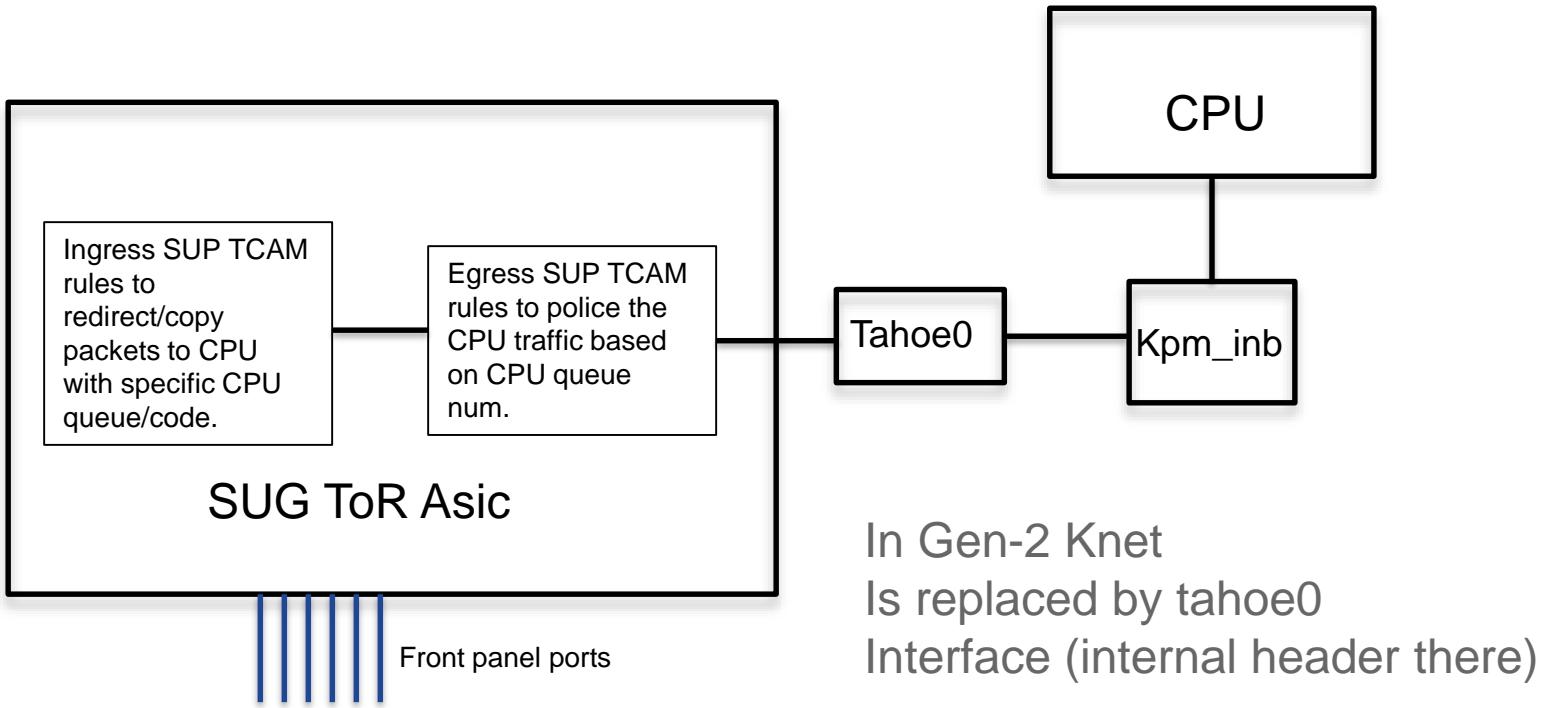
- All packets DMA'ed from the Bcom 2 ASIC
- Punt may be initiated by NS or Trident 2

Inband path on leaf – Gen1



- All packets DMA'ed from the Bcm ASIC
- Punt may be initiated by NS or BCM
- Knet (knet0, knet1 or knet2) depending on soft have Internal header
- Kpm_inb no more internal header

CPU inband Path – Gen 2 - LEaf



Ifconfig on leaf

```
bdsol-aci32-leaf3# ifconfig kpm_inb
kpm_inb    Link encap:Ethernet HWaddr 44:45:41:44:42:45
            inet addr:127.1.1.1 Bcast:127.1.255.255 Mask:255.255.0.0
            inet6 addr: fe80::4645:41ff:fe44:4245/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST MTU:9300 Metric:1
                      RX packets:15820759 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:8704957 errors:0 dropped:204 overruns:0 carrier:0
                      collisions:0 txqueuelen:0
                      RX bytes:114417104155704 (104.0 TiB) TX bytes:1407017100 (1.3 GiB)

bdsol-aci32-leaf3# ifconfig eth0
eth0    Link encap:Ethernet HWaddr cc:16:7e:7d:88:02
            inet addr:10.48.25.73 Bcast:10.48.25.127 Mask:255.255.255.128
            inet6 addr: fe80::ce16:7eff:fe7d:8802/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                      RX packets:468099 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:297246 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:63154914 (60.2 MiB) TX bytes:49956839 (47.6 MiB)

bdsol-aci32-leaf3# ifconfig tahoe0
tahoe0    Link encap:Ethernet HWaddr 44:45:41:44:42:45
                      UP BROADCAST RUNNING MULTICAST MTU:9500 Metric:1
                      RX packets:15821657 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:8705148 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:4089077720 (3.8 GiB) TX bytes:1441939139 (1.3 GiB)
```

Example : SNMP trap generated by leaf on the oob interface

```
pod2-leaf1# tcpdump -i eth0 -f port 162
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes

14:19:15.376847 IP 10.48.16.112.52967 > 10.48.37.151.snmp-trap: V2Trap(173) system.sysUpTime.0=33988813 S:1.1.4.1.0=E:cisco.9.276.0.1
interfaces.ifTable.ifEntry.ifIndex.436363264=436363264 interfaces.ifTable.ifEntry.ifAdminStatus.436363264=1
interfaces.ifTable.ifEntry.ifOperStatus.436363264=2 31.1.1.1.1.436363264="eth1/39" interfaces.ifTable.ifEntry.ifType.436363264=6
..
```

Example 2 – ntp packet received on inband

```
ESY-B-DR1-D05-LF-133# tcpdump -x -X -vv -i kpm_inb "port 123"
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
11:00:55.172450 IP (tos 0xb8, ttl 65, id 40819, offset 0, flags [none], proto UDP (17), length 76)
  10.6.7.47.ntp > 10.2.10.56.ntp: [udp sum ok] NTPv4, length 48
    Client, Leap indicator: clock unsynchronized (192), Stratum 0 (unspecified), poll 4s, precision -20
    Root Delay: 0.000000, Root dispersion: 2.302169, Reference-ID: (unspec)
    Reference Timestamp: 0.000000000
    Originator Timestamp: 3683520527.735999949 (2016/09/22 11:08:47)
    Receive Timestamp: 3683520039.174356446 (2016/09/22 11:00:39)
    Transmit Timestamp: 3683520055.172412201 (2016/09/22 11:00:55)
      Originator - Receive Timestamp: -488.561643502
      Originator - Transmit Timestamp: -472.563587747
  0x0000: 45b8 004c 9f73 0000 4111 b407 0a06 072f E..L.s..A...../
  0x0010: 0a02 0a38 007b 007b 0038 ea3f e300 04ec ...8.{.{.8.?....
  0x0020: 0000 0000 0002 4d5b 494e 4954 0000 0000 .....M[INIT...
  0x0030: 0000 0000 db8e 100f bc6a 7e20 db8e 0e27 .....j~....'
  0x0040: 2ca2 9fc3 db8e 0e37 2c23 34bf ,.....7,#4.
```

Example 3 – BGP packet from/to specific neighbour

```
bdsol-aci32-spine2# tcpdump -xxvvi kpm_inb -f port 179 and host 10.10.32.101
tcpdump: WARNING: kpm_inb: no IPv4 address assigned
tcpdump: listening on kpm_inb, link-type EN10MB (Ethernet), capture size 65535 bytes
08:29:17.957022 IP (tos 0xc0, ttl 102, id 4447, offset 0, flags [none], proto TCP (6), length 71)
    10.10.32.212.bgp > 10.10.32.101.59930: Flags [P.], cksum 0x67b6 (correct), seq 4207201624:4207201643,
ack 2111682847, win 30453, options [nop,nop,TS val 36279073 ecr 15056510], length 19: BGP, length: 19
        Keepalive Message (4), length: 19
        0x0000:  0000 0000 0000 0000 0000 0800 45c0
        0x0010:  0047 115f 0000 6606 ed45 0a0a 20d4 0a0a
        0x0020:  2065 00b3 ea1a fac4 cd58 7ddd b91f 8018
        0x0030:  76f5 67b6 0000 0101 080a 0229 9321 00e5
        0x0040:  be7e ffff ffff ffff ffff ffff ffff ffff
        0x0050:  ffff 0013 04
```

Tcpdump on knet or tahoe interface

Knet interface on Gen-1 and tahoe0 interface on Gen-2 Hardware shows all packet but they have an internal header so not fully readable

In Gen-2 Hardware you can use tcpdump2 which is script decoding internal header on the top of tcpdump

- bdsol-aci32-leaf2# ifconfig tahoe0
- tahoe0 Link encap:Ethernet HWaddr 44:45:41:44:42:45
- UP BROADCAST RUNNING MULTICAST MTU:9500 Metric:1
- RX packets:43270365 errors:0 dropped:8 overruns:0 frame:0
- TX packets:35159769 errors:0 dropped:0 overruns:0 carrier:0
- collisions:0 txqueuelen:1000
- RX bytes:19129761712 (17.8 GiB) TX bytes:5155845661 (4.8 GiB)

Traffic to/from CPU – Gen-2

```
bdsol-aci32-leaf1# cat /proc/tahoe/tahoed/stats
Tahoe ASIC CPU RX/TX Statistics
Asic: 0:
RX Packets: 87537650
RX Bytes: 47551576011
RX Interrupts: 88507233
NAPI Scheduled: 72545700
RX Packet Error Count: 0
RX Packet CRC Error Count: 0
RX Packet CRC Stomp Error Count: 0
RX Debug count: 0
RX ISR Yield count: 0
Packet RX rate: 32 pps
Packet RX rate Last 1 second: 6
MAX size of packet received: 8043
TX Packets: 75494357
TX Bytes: 11883261458
TX Dropped: 0
TX Interrupts: 73767422
Packet TX rate: 28 pps
Packet TX rate Last 1 second: 9
MAX size of packet sent: 5787
```

- SUP Traffic
 - One Pseudo Interface per ASIC – tahoe0, tahoe1... (eq of knet int on Gen-1 platform)
 - Uses DMA for packet RX/TX
 - On SUP for TOR, on FCs for Spine
 - **RX/TX use IETH header (similar to higig/sobmh in T2)**
 - tcpdump2 parses all of IETH header fields

```
bdsol-aci32-leaf1# ifconfig tahoe0
tahoe0      Link encap:Ethernet  HWaddr 44:45:41:44:42:45
            UP BROADCAST RUNNING MULTICAST  MTU:9500  Metric:1
            RX packets:87518502 errors:0 dropped:0 overruns:0 frame:0
            TX packets:75479593 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:47538616748 (44.2 GiB)  TX bytes:11879643424 (11.0 GiB)
```

Example

```
bdsol-aci32-leaf1# tcpdump2 -i tahoe0 icmp
tcpdump2: verbose output suppressed, use -v or -vv for full protocol decode
listening on tahoe0, link-type CISCO_IETH (TX header 16 Bytes), capture size 262144 bytes
17:25:30.560216 This is a TX packet
IETH INFO : SUP_CODE: 40, SRC_PORT: 0, SRC_CHIP: 0, SRC_IDX: 0, SRC_BD: 515
IP 10.20.1.254 > 10.20.1.1: ICMP echo reply, id 38998, seq 42496, length 64
17:25:31.561080 This is a TX packet
IETH INFO : SUP_CODE: 40, SRC_PORT: 0, SRC_CHIP: 0, SRC_IDX: 0, SRC_BD: 515
IP 10.20.1.254 > 10.20.1.1: ICMP echo reply, id 38998, seq 42752, length 64
```

```
bdsol-aci32-leaf1# tcpdump2 -xvvi tahoe0 -Q out icmp
tcpdump2: listening on tahoe0, link-type CISCO_IETH (TX header 16 Bytes), capture size 262144 bytes
17:31:06.605753 This is a TX packet
```

PKT_HASH:	0, SRC_IS_PEER:	0, TCLASS:	5, COS_DE:	10
SUP_CODE:	8, SUP_TX:	1, L2_TUNNEL:	0, DST_IS_TUNNEL:	0
SRC_IS_TUNNEL:	0, IP_TTL_BYPASS:	0, ALT_IF_PROFILE:	0, SPAN:	0
DONT_LRN:	0, TRACEROUTE:	0, BD_LO:	1, BD_HI:	257
OUTER_BD:	0, DST_PORT:	0, DST_CHIP_LO:	0, DST_CHIP_HI:	0
SRC_PORT:	0, SRC_CHIP:	0, DST_IDX_LO:	0, DST_IDX_HI:	0
SRC_IDX:	0, OPCODE:	0, EXT_HDR:	0, HDR_TYPE:	0
IP (tos 0x0, ttl 65, id 4965, offset 0, flags [none], proto ICMP (1), length 84)				
10.20.1.254 > 10.20.1.1: ICMP echo reply, id 38998, seq 62977, length 64				
0x0000:	0022 bdf8 19ff 0022 bdf8 19ff 0800 4500			
0x0010:	0054 1365 0000 4101 4f1e 0a14 01fe 0a14			
0x0020:	0101 0000 4a6c 9856 f601 00a6 9c58 b4b5			
0x0030:	0000 cdab 0000 cdab 0000 cdab 0000 cdab			
0x0040:	0000 cdab 0000 cdab 0000 cdab 0000 cdab			
0x0050:	0000 cdab 0000 cdab 0000 3031 3233 3435			
0x0060:	3637			

ELAM utility App

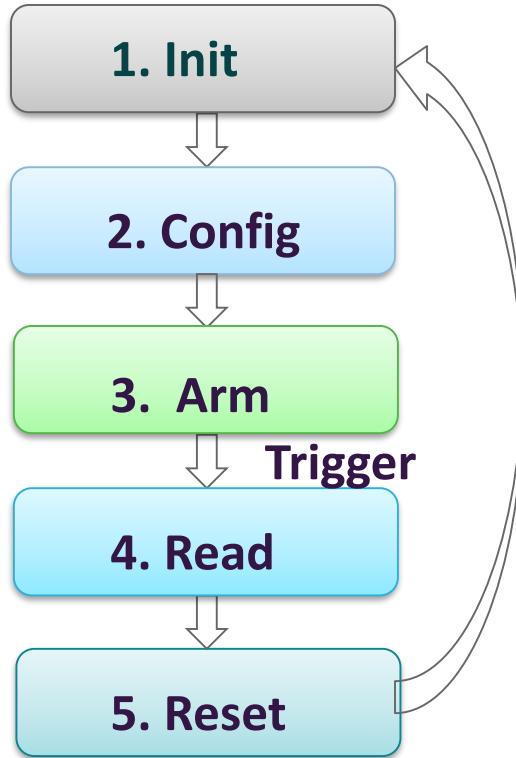
What is ELAM?

ELAM stands for Embedded Logic Analyzer Module

It is a logic that is present in the ASICs that provides the capability to capture and view one packet, that match a user specified criteria, from the stream of packets that are processed by the ASIC

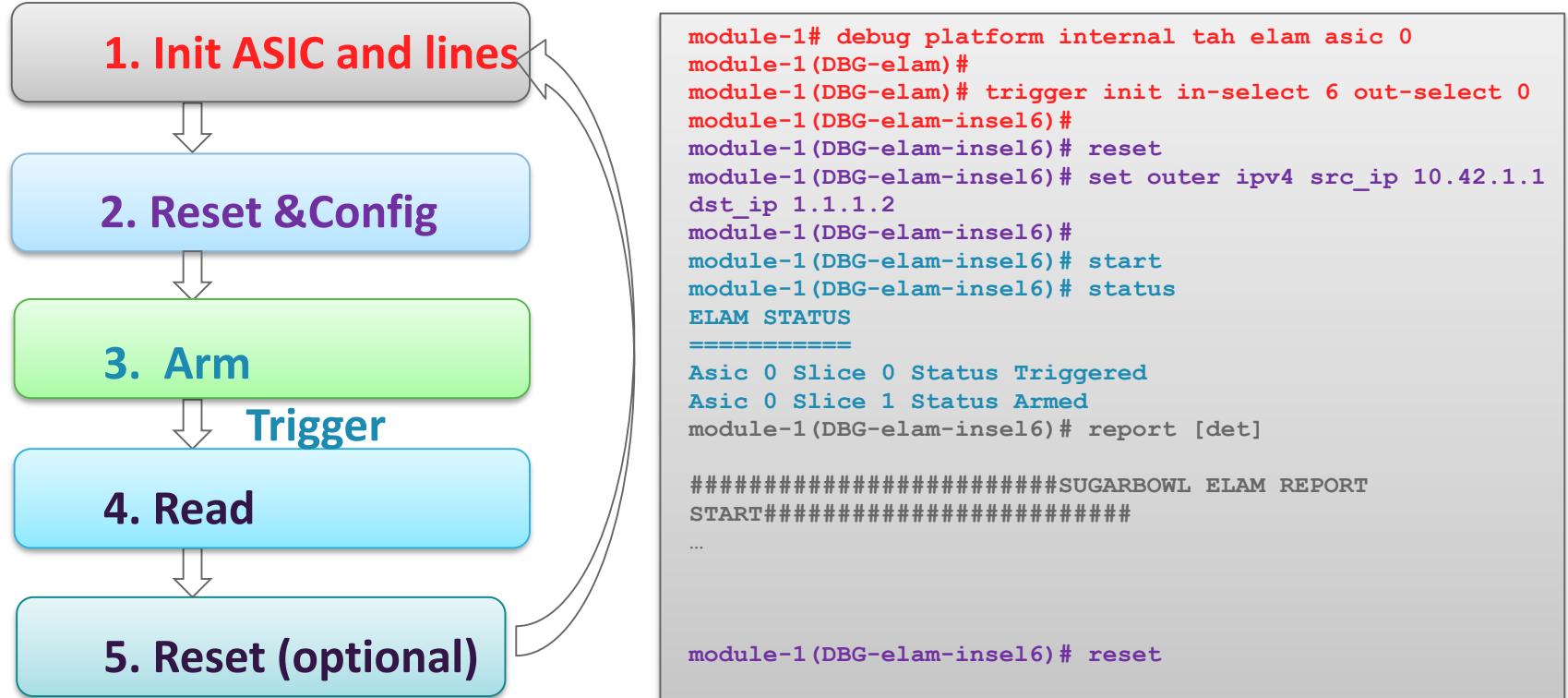
Cisco is using it for almost 15 years (catalyst 6500 sup720, cisco7600, nexus7k,..)

ELAM Configuration



- Init – Initialize the ELAM – **select the asic instance, pipeline and select lines (what type of field do we want to trigger on and do we want to see)**
- Reset and Config –
 - Always reset first to avoid using old trigger (clean phase)
 - Configure the trigger based on different fields in the packet
- Arm – Arm the trigger by setting the fields to match in hardware (start)
- Read – Once the trigger is triggered, read the report.
- Reset – Once the process is complete, reset the trigger to restart the process

ELAM Configuration – ingress on front panel on leaf example



ELAM Configuration – ingress from fab uplink on leaf example

1. Init ASIC and lines



2. Reset &Config



3. Arm



Trigger

4. Read



5. Reset (optional)

```
module-1# debug platform internal tah elam asic 0
module-1(DBG-elam)#
module-1(DBG-elam)# trigger init in-select 14 out-select 0
module-1(DBG-elam-insel6)#
module-1(DBG-elam-insel6)# reset
module-1(DBG-elam-insel6)# set inner ipv4 dst_ip 10.42.1.1 src_ip
1.1.1.2
module-1(DBG-elam-insel6)#
module-1(DBG-elam-insel6)# start
module-1(DBG-elam-insel14)# stat
ELAM STATUS
=====
Asic 0 Slice 0 Status Armed
Asic 0 Slice 1 Status Triggered
module-1(DBG-elam-insel6)# report [det]

#####SUGARBOWL ELAM REPORT
START#####
...
module-1(DBG-elam-insel6)# reset
```

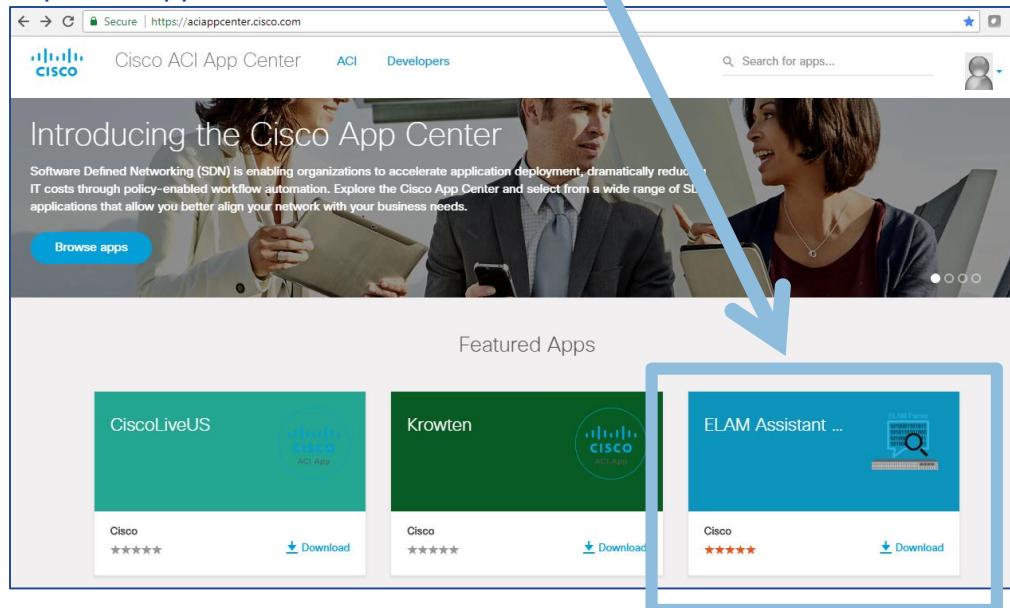
ELAM challenge

- CLI is hard to use and remember unless you do it everyday
- Report is complex and long. Some part are cryptic.
- Soluce – ELAM Apps...

ELAM Assistant in ACI AppCenter

Interested in more detail packet forwarding verification ?
➤ ELAM Assistant!!

<https://aciappcenter.com>



ELAM (Embedded Logic Analyzer Module)

- Perform an ASIC level packet capture

ELAM Assistant

- You can perform ELAM like a TAC engineer!
- With a nicely formatted result report

Detail Explanations:

- <https://aciappcenter.cisco.com/elam-assistant-2-2-1n-0-2-262.html>
 - How to use video, pictures
 - A download link for ELAM Assistant
- <https://learningnetwork.cisco.com/docs/DOC-34985>
 - ACI webinar for ELAM Assistant

ELAM Assistant in ACI AppCenter (example)

1. Perform ELAM

The screenshot shows the Cisco Application Center (ACI AppCenter) interface. The top navigation bar includes links for System, Tenants, Fabric, Virtual Networking, L4-L7 Services, Admin, Operations, Apps, Apps, and Faults. The user is logged in as 'admin'. The main content area is titled 'Capture a packet with ELAM (Embedded Logic Analyzer Module)'.

ELAM Assistant sidebar:

- Capture (Perform ELAM)
- node-105 (fab3-p1-leaf5)
- node-106 (fab3-p1-leaf6)
- node-202 (fab3-p2-leaf2)
- node-203 (fab3-p2-leaf3)
- node-204 (fab3-p2-leaf4)
- node-2001_slot1 (fab3-p2-spine1)
- node-2001_slot2 (fab3-p2-spine1)
- Unsupported Nodes

ELAM PARAMETERS section:

Name your capture: (optional)

Status	Node	Direction	Source I/F	Parameters	VxLAN (outer) header
Report Ready	node-202	from SPINE	any	+ dst ip 192.168.2.23	(+)
Report Ready	node-203	from frontport	any	+ dst ip 192.168.2.23	(+)
Set	node-2001_slot1	from LEAF/IPN	any	+ dst ip 192.168.2.23	(+)
Report Ready	node-2001_slot2	from LEAF/IPN	any	+ dst ip 192.168.2.23	(+)

Set Parameters button

ELAM Report Parse Result (report name: node-203_slot1_asic0_elam_report.txt)

Express Detail Raw

Triggered!! and **Report is Ready**

ELAM Assistant in ACI AppCenter (example)

2. Read a report

The screenshot illustrates the ELAM Assistant interface within the ACI AppCenter. The main window displays the following sections:

- ELAM Assistant**: A sidebar listing nodes and unsupported nodes.
- Capture a packet with ELAM (Embedded Logic Analyzer Module)**: A central panel for configuring packet capture parameters. It includes fields for Status, Direction, Source I/F, Parameters, and VxLAN (outer) header. Buttons include Report Ready, Set, and a blue Set button. Buttons for Set ELAM(s) and Check Trigger are also present.
- Packet Forwarding Information**: A panel showing forwarding details. It includes fields for Destination Type (To another ACI node (or AVS/AVE)), Destination TEP (11.0.40.66 (fab3-p2-leaf2)), Destination Physical Port (eth1/49), Sent to SUP/CPU instead (no), and SUP Redirect Reason (SUP code) (NONE).
- Contract**: A panel showing contract details. It includes fields for Destination EPG pcTag (dclass) (49154 (TK:APP1:EPG2-3)), Source EPG pcTag (sclass) (32777 (TK:APP1:EPG1-1)), and Contract was applied (1 (Contract was applied on this node)).
- Drop**: A panel showing drop code (no drop).

Annotations highlight specific areas:

- A blue box labeled "Click to see a report" points to the Report Ready buttons in the capture parameters section.
- A blue box labeled "Report shows up here" points to the "Report name: node-202" entry in the ELAM Report Parse Result section.
- A large blue arrow labeled "Scroll down" points towards the bottom of the interface, indicating where to find more detailed packet information.

ELAM Report Parse Result (report name: node-202)

Express Detail Raw

Captured Packet Information

Basic Information

Device Type	LEAF
Packet Direction	ingress (front panel port -> leaf)
Incomming I/F	eth1/21

L2 Header

Destination MAC	0022.BDF8.19FF
Source MAC	001A.2FD7.E2CB
Access Encap VLAN	1431
CoS	0

L3 Header

L3 Type	IPv4
Destination IP	192.168.2.23

Apps

ELAM Assistant (Beta)  Capture Packet node-101 (bdsol-aci32-leaf1) node-102 (bdsol-aci32-leaf2) node-103 (bdsol-aci32-leaf3)  node-104 (bdsol-aci32-leaf4) node-1105 (bdsol-aci32-leaf5) node-1106 (bdsol-aci32-leaf6) node-1201 (bdsol-aci32-spine5) node-2001 (bdsol-aci32-leaf-11-1) node-201 (bdsol-aci32-spine1) slot 1 (N9K-X9732C-EX) node-202 (bdsol-aci32-spine2) slot 1 (N9K-X9732C-EX) node-203 (bdsol-aci32-spine3) slot 2 (N9K-X9732C-EX)

node-103

ELAM REPORT FILES

	file name	date
 Parse	node-103_Test	2018-06-20 08:40:21 +00:00

ELAM REPORT PARSE RESULT

 Express  Detail  Raw 

Captured Packet Information --- [node-103_Test]

Trigger Information

Packet Direction	Front Panel Port (or CPU) -> LEAF	
Incoming Interface	eth1/11	Slice Source ID(Ss) 0x1c in "show plat int hal"

Outer L2 Header

Destination MAC	0022.BDF8.19FF
-----------------	----------------

Express report

Trigger Information			
Tracked Direction	Front Panel port (or CPU) -> LEAF		
Incoming Interface	eth1/11		
Slice Source ID(Ss) 0x1c in "show plat int hal12 port gpd"			
Outer L2 Header			
Destination MAC	0022.BDF8.19FF		
Source MAC	24E9.B30A.CDC1		
CoS	0x0 (0)		
Access Encap VLAN	0x518 (1304)		
Outer L3 Header			
L3 Type	IPv4		
DSCP	0x0 (0)		
IP Packet Length	0x3C (60) (= IP header(28 bytes) + IP payload)		
Don't Fragment Bit	0x1		
TTL	0x2F (63)		
IP Protocol Number	6 (TCP)		
IP CheckSum	0x548B		
Destination IP	192.168.102.1		
Source IP	192.168.101.1		
Outer L4 Header			
L4 Type	TCP		
Packet Forwarding Information			
Destination is Local port			
Destination Port	eth1/9		Ovec (0x18) in "show plat int hal12 port gpd"
Destination Logical Port	ETH1/9		EIO (0x41) / m -> show plat int hal12 port gpd
EPG Classification (pcTag)			
Destination EPG pcTag (dclass)	0x8002 (32770)		RD-PBR:One:Server
Source EPG pcTag (sclass)	0xC002 (49154)		RD-PBR:3k3-pbr-one-armcbsONE:one-arm:
Policy Applied	1 (Contract was applied)		
Lookup Drop			

End point tracker App

Recommend Troubleshooting Apps

<https://aciappcenter.cisco.com/>



EnhancedEndpointTracker

The EnhancedEndpointTracker is a Cisco ACI application that maintains a database of endpoint events on a per-node basis allowing for unique fabric-wide analysis. The application can be configured to analyze, notify, and automatically remediate various endpoint events. This gives ACI fabric operators better visibility and control over the endpoints in the fabric.



Enhanced Endpoint Tracker

Endpoint Tracker

Fast search for IP or MAC

IP Address	Type	Port	Interface	Encapsulation	pcTAG	MAC Address
2001::1	ipv6	-	uni/tn-scale/ctx-v-large	-	-	-
2001::1000	ipv6	vlan-999	uni/tn-scale/ctx-v-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large
2001::1001	ipv6	vlan-999	uni/tn-scale/ctx-v-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large
2001::1002	ipv6	vlan-999	uni/tn-scale/ctx-v-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large
2001::1003	ipv6	vlan-999	uni/tn-scale/ctx-v-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large
2001::1004	ipv6	vlan-999	uni/tn-scale/ctx-v-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large	uni/tn-scale/ap-scale-large/epg-epg-large

Time	Local Node	Status	Interface	Encap	pcTAG	MAC
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1002	vlan-101	49153	00:00:00:00:
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1002	vlan-101	49153	00:00:00:00:
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:
Mar 05 2019 - 12:32:11	(101,102)	created	ag_po1002	vlan-101	49153	00:00:00:00:

Elements Console Sources Network Performance Memory Application Security Audits

View: Group by frame Preserve log Disable cache Offline Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Status	Type	Ini...	Size	Time	Waterfall
endpoint?filter=and(eq(%22fabric%22,%22fab4...)	200	xhr	po...	8.8 KB	151 ms	
endpoint?filter=and(eq(%22fabric%22,%22fab4...)	200	xhr	po...	8.8 KB	157 ms	

~150ms for search to complete

Enhanced Endpoint Tracker



Historical tables to browse various events along with browsing all endpoints in the fabric

The screenshot shows a sidebar menu with the following items:

- Dashboard
- Endpoints
- Settings
- About

A search bar at the top right says "Search MAC or IP for this fabric. i.e., 00:50:56:01:BB:12, 10.1.1.101, or 2".

Top moves in the fabric, quickly see any unstable/misconfigured endpoints

The screenshot shows a table titled "Moves" with the following data:

Time	Type	Address	Event Count	VRF/BD
Mar 05 2019 - 11:46:46	ipv4	10.1.1.101	8,295	uni/tn-ag/ctx-v1
Mar 05 2019 - 10:47:16	mac	00:00:00:00:00:A	104	uni/tn-ag/BD-bd1
Mar 05 2019 - 10:42:27	ipv4	10.1.1.102	102	uni/tn-ag/ctx-v1
Mar 05 2019 - 10:47:06	ipv4	10.5.1.101	101	uni/tn-ag/ctx-v1
Mar 05 2019 - 10:00:51	mac	00:00:01:50:01:01	3	uni/tn-ag/BD-bd1
Mar 05 2019 - 10:00:51	ipv4	10.1.1.111	1	uni/tn-ag/ctx-v1

At the bottom left, it says "6 total".



Enhanced Endpoint Tracker

Full details of current state of endpoint within the fabric including local and XR learns

Endpoint Tracker

Dashboard

Endpoints

Settings

About

Search MAC or IP for this fabric. I.e., 00:50:56:01:BB:65

10.1.1.101

Fabric fab4 VRF uni/tn-ag/ctx-v1 EPG uni/tn-ag/ap-app/epg-e1

Local on pod-1 node (101,102) interface ag_po1001 encaps vlan-101 mac 00:00:00:00:00:0A

Remotely learned on 1 node. ▾

103

37,738 Moves 68 Rapid events 0 OffSubnet events 0 Stale events 0 Clear events

Actions ▾

History Detailed Move Rapid OffSubnet Stale Cleared

Time	Local Node	Status	Interface	Encap	pcTAG	MAC	EPG
Feb 20 2017 10:24	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:00:0A	uni/tn-ag/ap-app/epg-e1
Feb 20 2017 10:24	(101,102)	created	ag_po1002	vlan-101	49153	00:00:00:00:00:0B	uni/tn-ag/ap-app/epg-e1
Feb 20 2017 10:24	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:00:0A	uni/tn-ag/ap-app/epg-e1
Feb 20 2017 10:24	(101,102)	created	ag_po1001	vlan-101	49153	00:00:00:00:00:0A	uni/tn-ag/ap-app/epg-e1

Also per-node detailed history, move events, rapid/offsubnet/stale/and clear events

History of where endpoint was learned or if it was deleted from the fabric





Enhanced Endpoint Tracker

The screenshot shows a network management interface for an endpoint tracker. At the top, it displays the IP address 10.1.1.111 and two nodes: 101 and 102, both labeled as 'stale'. Below this, the fabric information is shown as Fabric fab4 VRF uni/tn-ag/ctx-v1 EPG uni/tn-ag/ap-app/epg-e1. A 'Nodes' section lists 101, 102, and 103. On the right, there is an 'Actions' dropdown menu with options: Dataplane Refresh, Delete Events, and Clear Endpoint. A large blue callout bubble points to the 'Clear Endpoint' button in the dialog box, which is highlighted with a blue border. The callout contains the text: 'Clear problem endpoints on multiple nodes quickly'. The dialog box itself has a title 'Clear Endpoint' and a warning message: 'This action may impact dataplane traffic on the fabric until the endpoint is relearned.' It includes a 'Nodes' input field with placeholder text 'Enter range or comma separated nodes to be cleared (e.g. 100-106, 301, 303)', three toggle buttons for filtering nodes (all learned, offsubnet, or stale), and two buttons at the bottom: 'Cancel' and a large red 'Clear Endpoint' button.

Ftriage Tool

What is Ftriage ?

- Fabric triaging tool [A unicorn really!]
- **Python utility, runs on APIC in admin mode.**
- **Logs into switch nodes to capture requested data with commands/query/elam.**
- Driven by specific user inputs which are validated first.
- Runs Elam to determine packet data path
- Runs show commands/moquery on APIC/switch to determine control plane
- **Traces a packet hop by hop till the point where it exits the fabric or gets dropped.**
- Provides detailed info on interfaces + nodes where packet capture is attempted
- **Drop reason is provided along with node, interface info.**

Software available

- Feature exists for a long time already
- Need data traffic running (as relying on ELAM)
- Not a tool for intermittent connectivity issue
- Might take few minutes to complete (depending on type of flow)
- **A lot of enhancement in 4.0 code**
 - Input simplified (src and dst ip + input port) is enough
 - Full log of session is available for further debugging

Help

```
apic1# ftriage -h
usage: ftriage [-h] [-loglevel loglevel] [-user username]
                [-nsdropaction nsdropaction] [-xt xt] [-passwd] [-wait WAIT]
                {bridge,example,route} ...
```

Tool to automate some of the common fabric triaging tasks. It takes the packet fields as seen on wire as input. The more the fields, the higher the chance of tracing the right flow. Please maintain a minimum rate of 1 pps for the concerned flow. As some of the filters are not tenant aware, please ensure that there are no other flows with the same packet signature. The node names in the arguments refer to what is displayed by the APIC command 'acidiag fnvread' under the column 'Name'.nodes can also be specified as arguments using node Id.Use PC/VPC when ports are part of a bundle.

What's not yet supported: Bounce, Multisite-Multicast

optional arguments:

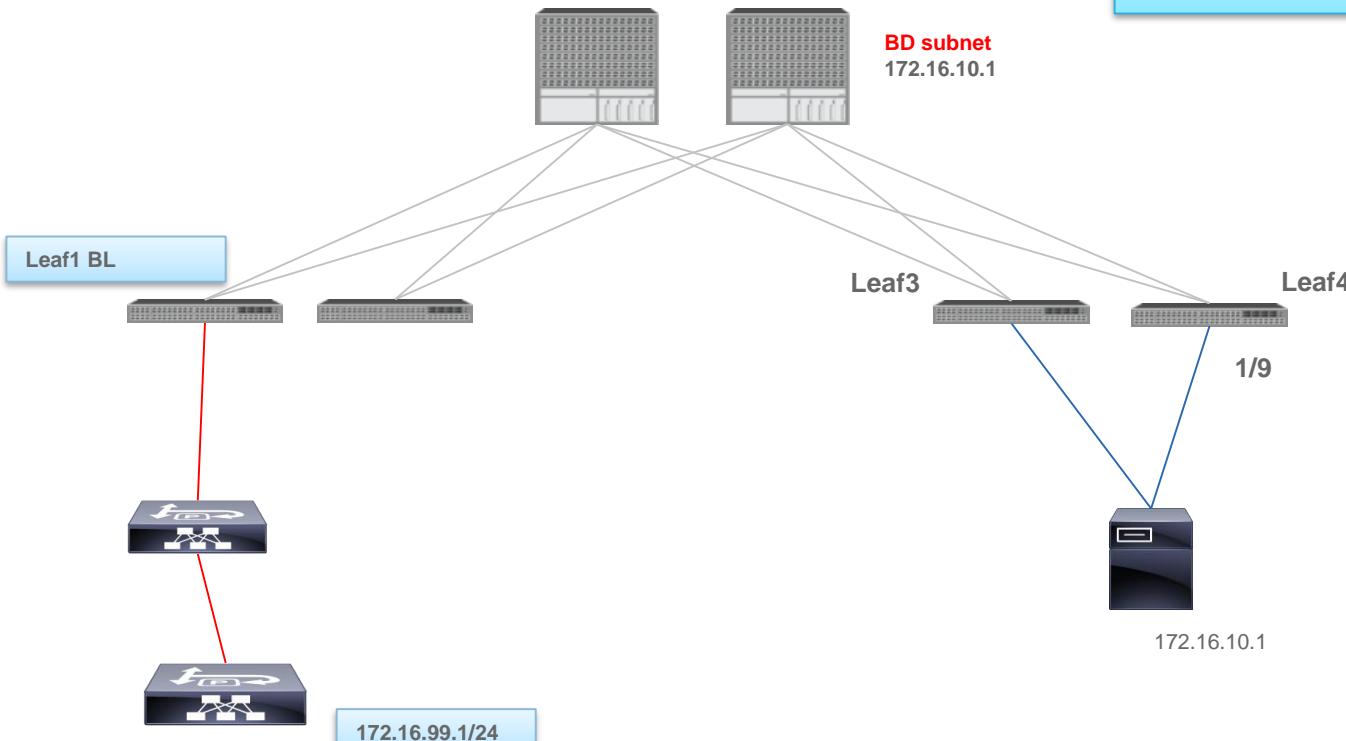
-h, --help	show this help message and exit
-loglevel loglevel	enable script debugging level (1=ERROR, 2=WARNING, 3=INFO, 4=DEBUG/VERBOSE)
-wait WAIT	packet capture wait time in secs (default: 10 secs)

subcommands:

{bridge,example,route}	
bridge	Transit bridge
example	examples help
route	Transit route

Lab topology (tested in 4.0)

Example flow from leaf4 eth1/9
Src 172.16.10.1 to
Dst 172.16.99.1 (behind BGP L3 out on leaf1)



Example – packet from an EP in leaf 4 to a l3 out ip 172.16.99.1

```
apic1# ftriage route -sip 172.16.10.1 -dip 172.16.99.1 -ii 104:eth1/9
Starting ftriage
Log file name for the current run is: ftlog_2018-09-12-10-47-48-445.txt
2018-09-12 10:47:49,960 INFO /controller/bin/ftriage route -sip 172.16.10.1 -dip 172.16.99.1 -ii 104:eth1/9
..
2018-09-12 10:48:05,732 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-leaf4 IF: Eth1/9
2018-09-12 10:49:20,091 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-leaf4 Ingress: Eth1/9 Egress: Eth1/52
2018-09-12 10:49:20,091 INFO ftriage: main:233 Computed ingress encap string vlan-2016
2018-09-12 10:49:20,604 INFO ftriage: main:262 Building ingress BD(s), Ctx
2018-09-12 10:49:23,189 INFO ftriage: main:279 Ingress BD(s) bd-[vxlan-16154560]
2018-09-12 10:49:23,189 INFO ftriage: main:282 Ingress Ctx ctx-[vxlan-2654211]
2018-09-12 10:50:25,242 INFO ftriage: nxos:1296 nxos matching rule id:4246 scope:16 filter:5
2018-09-12 10:50:36,190 INFO ftriage: main:954 SIP 172.16.10.1 DIP 172.16.99.1
2018-09-12 10:50:36,191 INFO ftriage: unicast:940 bdsol-aci32-leaf4: Ingress Sub function
2018-09-12 10:50:38,630 INFO ftriage: unicast:1021 bdsol-aci32-leaf4: Dst EP is a WAN EP
2018-09-12 10:50:38,630 INFO ftriage: unicast:1033 bdsol-aci32-leaf4: Policy enforcement mode is ingress
2018-09-12 10:50:38,630 INFO ftriage: unicast:1128 bdsol-aci32-leaf4: Dst EP is remote
2018-09-12 10:50:39,692 INFO ftriage: misc:691 bdsol-aci32-leaf4: DMAC(00:22:BD:F8:19:FF) same as RMAC(00:22:BD:F8:19:FF)
2018-09-12 10:50:39,693 INFO ftriage: misc:693 bdsol-aci32-leaf4: L3 packet getting routed/bounce in SUG
2018-09-12 10:50:43,993 INFO ftriage: misc:691 bdsol-aci32-leaf4: RwdMAC DIPo(10.0.88.95) is one of dst TEPs ['10.0.88.95']
2018-09-12 10:50:48,588 INFO ftriage: node:945 Computing next set of nodes
2018-09-12 10:53:00,319 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-spine1 IF: Eth1/4
2018-09-12 10:53:27,093 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-spine4 IF: Eth1/4
2018-09-12 10:54:42,024 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-spine4 Ingress: Eth1/4 Egress: LC1-0 FC24-0 Po2 Port-1
2018-09-12 10:54:54,821 INFO ftriage: fib:267 bdsol-aci32-spine4: Transit in spine
2018-09-12 10:55:12,432 INFO ftriage: unicast:1572 bdsol-aci32-spine4: Infra route 10.0.88.95 present in RIB
2018-09-12 10:56:25,186 INFO ftriage: unicast:1604 bdsol-aci32-spine4: L3 packet Spine Transit egress pkt seen on bdsol-aci32-spine4
Ingress: LC1-0 FC24-0 Po2 Port-1 Egress: Eth1/1
2018-09-12 10:56:38,144 INFO ftriage: node:945 Computing next set of nodes
2018-09-12 10:57:24,963 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-leaf1 IF: Eth1/52
2018-09-12 10:58:39,528 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-leaf1 Ingress: Eth1/52 Egress: Eth1/3
2018-09-12 10:59:37,161 INFO ftriage: fib:154 bdsol-aci32-leaf1: L3 out interface Ethernet1/3
2018-09-12 10:59:50,283 INFO ftriage: fib:433 bdsol-aci32-leaf1: Entry hit is in tcam
2018-09-12 10:59:50,283 INFO ftriage: fib:436 bdsol-aci32-leaf1: L2ptr is valid. Index hit: 32805
2018-09-12 10:59:56,399 INFO ftriage: main:496 Computed egress encap string vlan-1104
2018-09-12 10:59:56,927 INFO ftriage: main:293 Building egress BD(s), Ctx
2018-09-12 10:59:59,553 INFO ftriage: main:309 Egress BD(s) bd-[vxlan-14909415]
2018-09-12 10:59:59,553 INFO ftriage: main:310 Egress Ctx ctx-[vxlan-2654211]
2018-09-12 11:00:01,298 INFO ftriage: unicast:1871 bdsol-aci32-leaf1: Dst EP is local
2018-09-12 11:00:01,298 INFO ftriage: unicast:583 : Ftriage Completed with hunch: Nonen
```

Log file

- Log file was created with all detail info if TAC needs further look at it
- Contains all show command performed and all ELAM output !

```
apic1# pwd
/home/admin
apic1# ls -al ftlog_2018-09-12-10-47-48-445.txt
-rw-r--r-- 1 admin admin 5946083 Sep 12 11:00 ftlog_2018-09-12-10-47-48-445.txt
apic1#
```

Example 2- Unknown unicast L2 flooded frame in BD GIPo

```
ftriage bridge -ii 104:eth1/9 -sip 172.16.10.1 -dip 172.16.10.20
```

Grep on output :

```
2018-09-17 14:47:50,219 INFO ftriage: mtree:475 Tracing packet in pod-id : 1 ,ingress interface : bdsol-aci32-leaf4:Eth1/9
2018-09-17 14:47:50,219 INFO ftriage: mtree:476 GIPO : 225.1.120.144
2018-09-17 14:47:50,219 INFO ftriage: mtree:478 FTAG : 0
2018-09-17 14:47:55,451 INFO ftriage: mtree:97 Egress Interfaces : ['Eth1/49', 'Eth1/51', 'Eth1/50', 'Eth1/52']
2018-09-17 14:48:40,637 INFO ftriage: mtree:112 bdsol-aci32-leaf4:Eth1/49---->bdsol-aci32-spine1.cisco.com:Eth1/4
2018-09-17 14:48:40,638 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-spine1 at Eth1/4
2018-09-17 14:50:04,793 INFO ftriage: mtree:97 Egress Interfaces : ['Eth1/3', 'Eth1/2']
2018-09-17 14:50:51,641 INFO ftriage: mtree:112 bdsol-aci32-spine1:Eth1/3---->bdsol-aci32-leaf3.cisco.com:Eth1/49
2018-09-17 14:50:51,641 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-leaf3 at Eth1/49
2018-09-17 14:52:16,214 INFO ftriage: mtree:97 Egress Interfaces : []
2018-09-17 14:52:59,051 INFO ftriage: mtree:112 bdsol-aci32-spine1:Eth1/2---->bdsol-aci32-leaf2.cisco.com:Eth1/49
2018-09-17 14:52:59,051 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-leaf2 at Eth1/49
2018-09-17 14:54:23,376 INFO ftriage: mtree:97 Egress Interfaces : []
2018-09-17 14:55:06,441 INFO ftriage: mtree:112 bdsol-aci32-leaf4:Eth1/51---->bdsol-aci32-spine3.cisco.com:Eth2/4
2018-09-17 14:55:06,441 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-spine3 at Eth2/4
2018-09-17 14:56:30,311 INFO ftriage: mtree:97 Egress Interfaces : []
2018-09-17 14:57:13,042 INFO ftriage: mtree:112 bdsol-aci32-leaf4:Eth1/50---->bdsol-aci32-spine2.cisco.com:Eth1/4
2018-09-17 14:57:13,042 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-spine2 at Eth1/4
2018-09-17 14:58:36,963 INFO ftriage: mtree:97 Egress Interfaces : []
2018-09-17 14:59:19,817 INFO ftriage: mtree:112 bdsol-aci32-leaf4:Eth1/52---->bdsol-aci32-spine4.cisco.com:Eth1/4
2018-09-17 14:59:19,817 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-spine4 at Eth1/4
2018-09-17 15:00:44,019 INFO ftriage: mtree:97 Egress Interfaces : []
2018-09-17 15:00:51,975 INFO ftriage: mtree:475 Tracing packet in pod-id : 2 ,ingress interface : bdsol-aci32-spine5:Eth1/11
2018-09-17 15:00:51,975 INFO ftriage: mtree:476 GIPO : 225.1.120.144
2018-09-17 15:00:51,975 INFO ftriage: mtree:44 Capturing L2 frame on bdsol-aci32-spine5 at Eth1/11
```

```
apic1# ls -al ftlog_2018-09-17-14-46-12-044.txt
-rw-r--r-- 1 admin admin 10163495 Sep 17 15:01 ftlog_2018-09-17-14-46-12-044.txt
```

Example 3 – Packet dropped by a contract icmp not allowed per contract

```
apic1# ftriage route -ii 104:eth1/9 -sip 172.16.10.1 -dip 172.16.99.1
Starting ftriage
Log file name for the current run is: ftlog_2018-09-17-16-07-08-518.txt
2018-09-17 16:07:10,043 INFO      /controller/bin/ftriage route -ii 104:eth1/9 -sip 172.16.10.1 -dip 172.16.99.1
2018-09-17 16:08:38,414 INFO      ftriage: main:778 L3 packet Seen on bdsol-aci32-leaf4 Ingress: Eth1/9 Egress:
Eth1/21
2018-09-17 16:09:23,820 INFO      ftriage: unicast:1128 bdsol-aci32-leaf4: Dst EP is remote
2018-09-17 16:09:23,820 ERROR drop reason
SECURITY_GROUP_DENY condition set
2018-09-17 16:09:23,821 INFO      ftriage: unicast:227 bdsol-aci32-leaf4: L3 packet getting fwd dropped, checking
condition setcast:229 bdsol-aci32-leaf4: Drop reason - SECURITY_GROUP_DENY
2018-09-17 16:09:23,821 INFO      ftriage: unicast:244 bdsol-aci32-leaf4: policy drop flow sclass:49154 dclass:16387
sg_label:16 proto:1
2018-09-17 16:09:23,821 INFO      ftriage: unicast:250 bdsol-aci32-leaf4: nxos matching rule id:4209
2018-09-17 16:09:23,821 INFO      ftriage: unicast:256 bdsol-aci32-leaf4: tag info not match sclass: 0 dclass: 0
2018-09-17 16:09:23,821 INFO      ftriage: unicast:265 bdsol-aci32-leaf4: PE matching rule id:4247,
```

```
bdsol-aci32-leaf4# show zoning-rule | egrep 4209
4209          0            0      implicit      enabled      2654211      deny,log
bdsol-aci32-leaf4#
```



Example 4 – ftriage remote leaf to Server leaf in Pod

```
apic1# ftriage route -ii 2001:eth1/11 -sip 10.200.1.111 -dip 10.200.2.102
Log file name for the current run is: ftlog_2018-10-09-07-35-44-518.txt
2018-10-09 07:35:46,032 INFO /controller/bin/ftriage route -ii 2001:eth1/11 -sip 10.200.1.111 -dip 10.200.2.102
2018-10-09 07:35:54,243 INFO ftriage: misc:792 External interface bdsol-aci32-spine2 NodeId:202 Ifs:Eth1/11 LC:1 Port:11
2018-10-09 07:35:56,379 INFO ftriage: misc:792 External interface bdsol-aci32-spine4 NodeId:204 Ifs:Eth1/11 LC:1 Port:11
2018-10-09 07:36:00,513 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-rleaf-11 IF: Eth1/11
2018-10-09 07:37:14,677 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-rleaf-11 Ingress: Eth1/11 Egress: Eth1/49
2018-10-09 07:37:14,677 INFO ftriage: main:233 Computed ingress encap string vlan-1001
2018-10-09 07:37:15,221 INFO ftriage: main:262 Building ingress BD(s), Ctx
2018-10-09 07:37:17,639 INFO ftriage: main:279 Ingress BD(s) bd-[vxlan-15466400]
2018-10-09 07:37:17,654 INFO ftriage: main:282 Ingress Ctx ctx-[vxlan-3014656]
2018-10-09 07:38:26,123 INFO ftriage: fib:426 bdsol-aci32-rleaf-11: Correct ip-addr and vrf-id used in fibdakey
2018-10-09 07:38:26,123 INFO ftriage: fib:431 bdsol-aci32-rleaf-11: FIB-DA lkup was a hit at idx 3 prefix_len: 24
2018-10-09 07:38:26,124 INFO ftriage: fib:433 bdsol-aci32-rleaf-11: Entry hit is in tcam
2018-10-09 07:38:26,124 INFO ftriage: fib:436 bdsol-aci32-rleaf-11: L2ptr is valid. Index hit: 32788
2018-10-09 07:38:30,727 INFO ftriage: main:954 SIP 10.200.1.111 DIP 10.200.2.102
2018-10-09 07:38:30,728 INFO ftriage: unicast:940 bdsol-aci32-rleaf-11: Ingress Sub function
2018-10-09 07:38:34,062 INFO ftriage: misc:691 bdsol-aci32-rleaf-11: DMAC(00:22:BD:F8:19:FF) same as RMAC(00:22:BD:F8:19:FF)
2018-10-09 07:38:34,063 INFO ftriage: misc:693 bdsol-aci32-rleaf-11: L3 packet getting routed/bounced in SUG
2018-10-09 07:38:38,306 INFO ftriage: misc:691 bdsol-aci32-rleaf-11: RwdMAC DIPo(10.0.0.36) is one of dst TEPs ['10.0.0.36']
2018-10-09 07:38:40,543 INFO ftriage: unicast:1086 bdsol-aci32-rleaf-11: Dst EP is Spine TEP on Remote Leaf
2018-10-09 07:38:41,470 INFO ftriage: misc:691 bdsol-aci32-rleaf-11: DMAC(00:22:BD:F8:19:FF) same as RMAC(00:22:BD:F8:19:FF)
2018-10-09 07:38:41,471 INFO ftriage: misc:693 bdsol-aci32-rleaf-11: L3 packet getting routed/bounced in SUG
2018-10-09 07:40:33,957 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-spine2 IF: Eth1/11
2018-10-09 07:41:48,749 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-spine2 Ingress: Eth1/11 Egress: LC1-1 FC26-0 Po0 Port-0
2018-10-09 07:42:21,438 INFO ftriage: unicast:1151 bdsol-aci32-spine2: Proxy Sub function
2018-10-09 07:42:21,438 INFO ftriage: unicast:1203 bdsol-aci32-spine2: EP is known in COOP (DIPo = 10.0.80.94)
2018-10-09 07:42:25,409 INFO ftriage: unicast:1212 bdsol-aci32-spine2: Synth VRF 292 Synth IP 38.25.15.162
2018-10-09 07:42:29,389 INFO ftriage: unicast:1325 bdsol-aci32-spine2: Capturing Proxy egress LC Pkt L3 packet on Node: bdsol-aci32-spine2 IFS: Eth1/2
2018-10-09 07:43:42,307 INFO ftriage: unicast:1348 bdsol-aci32-spine2: L3 packet Proxy egress pkt seen on bdsol-aci32-spine2 Ingress: LC1-0 FC26-0 Po0 Port-0 Egress: Eth1/2
2018-10-09 07:43:57,149 INFO ftriage: node:945 Computing next set of nodes
2018-10-09 07:44:44,025 INFO ftriage: main:680 Capturing L3 packet Fex: False on node: bdsol-aci32-leaf2 IF: Eth1/50
2018-10-09 07:45:58,283 INFO ftriage: main:778 L3 packet Seen on bdsol-aci32-leaf2 Ingress: Eth1/50 Egress: Eth1/3
2018-10-09 07:46:57,513 INFO ftriage: nxos:1296 nxos matching rule id:4319 scope:10 filter:65535
2018-10-09 07:47:05,801 INFO ftriage: fib:426 bdsol-aci32-leaf2: Correct ip-addr and vrf-id used in fibdakey
2018-10-09 07:47:05,801 INFO ftriage: fib:431 bdsol-aci32-leaf2: FIB-DA lkup was a hit at idx 24736 prefix_len: 32
2018-10-09 07:47:05,801 INFO ftriage: fib:436 bdsol-aci32-leaf2: L2ptr is valid. Index hit: 2393
2018-10-09 07:47:11,893 INFO ftriage: main:496 Computed egress encap string vlan-1002
2018-10-09 07:47:12,447 INFO ftriage: main:293 Building egress BD(s), Ctx
.
2018-10-09 07:47:24,338 INFO ftriage: unicast:553 : Ftriage Completed with hunch: None
```

Ftriage

- Not all feature supported yet
 - For multicast L2 or L3 , front panel are not supported yet (mroute oil and igmp receiver). Only within fabric path is supported
- Constant improvement in usability
- Starting in 4.0 provides full log
- So even if ftriage do not complete a lot of logs already gathered

Issue seen with admin when tacacs is configured

```
apic-ams# ftriage route -sip 192.168.1.1 -dip 10.50.128.10 -ii 103:eth1/1
Starting ftriage
Log file name for the current run is: ftlog_2018-10-26-10-23-57-391.txt
2018-10-26 10:23:58,094 INFO    /controller/bin/ftriage route -sip 192.168.1.1 -dip 10.50.128.10 -ii 103:eth1/1
2018-10-26 10:24:04,973 ERROR    ftriage:      node:125 : Ftriage Completed with Incorrect apic#fallback\\admin password for
leaf-3
2018-10-26 08:24:04,974 ERROR          node:125 : Ftriage Completed with Incorrect apic#fallback\\admin password for leaf-3
```

Workaround

```
apic-ams# ftriage -user admin -passwd -loglevel 4 route -sip 192.168.1.1 -dip 10.50.128.10 -ii 103:eth1/1
Starting ftriage
Log file name for the current run is: ftlog_2018-10-26-10-26-55-497.txt
2018-10-26 10:26:56,179 INFO    /controller/bin/ftriage -user admin -passwd -loglevel 4 route -sip 192.168.1.1 -dip
10.50.128.10 -ii 103:eth1/1
Request password info for username: admin
Password:
2018-10-26 10:27:08,034 DEBUG    ftriage:      node:138 Added node:leaf-3 Spine:False
2018-10-26 10:27:08,035 INFO     ftriage:      main:1110 Invoking ftriage with username: admin
2018-10-26 10:27:08,035 DEBUG    ftriage:      main:1112          APIC-MOQ-CMD: - moquery -c 13extRsPathL3OutAtt
2018-10-26 10:27:09,284 DEBUG    ftriage:      misc:780 Identify a new node: spine-1 podid: 1 of type: External-Spine
```

Ftriage in 4.2 Example

```
ftriage route -ii LEAF:101 -sip 10.1.1.1 -dip 10.1.2.1
```

```
ftriage route -ii LEAF:103,104 -sip 10.1.2.1 -dip 10.1.1.1
```

Linux utility

ACI is based on full linux system

- Everything you know about linux and tools in linux is likely applicable for ACI ...
- Tcpdump/sed/awk/grep/while/watch/top

Awk / sed /grep

- List all vlan use anywhere as encap in the fabric... from :

```
pod2-apic2# moquery -c fvIfConn | egrep "dn.*vlan-"
```

```
dn          : uni/epp/fv-[uni/tn-Services/ap-App1/epg-Server]/node-101/dyatt-[topology/pod-1/protpaths-101-102/pathep-[ESX01-vpc]]/conndef/conn-[vlan-1049]-[0.0.0.0]
```

- You can get :

```
admin@pod2-apic3:~> moquery -c fvIfConn | egrep "dn.*vlan-" | awk '{print $3}' | sed '.*vlan-//g' | sed 's/]-.*/g' | uniq | sort -n | tr '\n' ''  
101 102 102 103 112 120 211 240 850 851 852 853 860 870 1000 1004 1004 1070 1071  
1072 1072 1073 1074 1074 1075 1134 1134 1137 1138 1139 1140 3891 3962
```

While loop

```
While true ; do date ; show .... ; show.... ; sleep x ; done > /bootflash/test-while.txt
```

Example:

```
pod2-leaf1# while true ; do date ; show ip route vrf L3:L3 | egrep -A1 "10.34.11.0" ;  
vsh -c 'show ip bgp vrf L3:L3' | egrep "10.34.11.0" ; sleep 1 ; done
```

```
Mon Apr 18 19:31:16 UTC 2016
```

```
10.34.11.0/24, ubest/mbest: 1/0
```

```
*via 10.33.10.2%L3:L3, [20/0], 6d21h, bgp-101, external, tag 100
```

```
*>e10.34.11.0/24          10.33.10.2                      0 100 99 i
```

```
Mon Apr 18 19:31:26 UTC 2016
```

```
10.34.11.0/24, ubest/mbest: 1/0
```

```
*via 10.33.10.2%L3:L3, [20/0], 6d21h, bgp-101, external, tag 100
```

```
*>e10.34.11.0/24          10.33.10.2                      0 100 99 i
```

Watch

```
pod2-leaf1# watch 'show interface ether 1/33 counters'
```

```
Every 2.0s: show interface ether 1/33 counters
Mon Apr 18 19:35:19 2016
```

Port	InOctets	InUcastPkts
Eth1/33	1079097526	655923
Port	InMcastPkts	InBcastPkts
Eth1/33	14091741	8
Port	OutOctets	OutUcastPkts
Eth1/33	66282372	28423
Port	OutMcastPkts	OutBcastPkts
Eth1/33	752235	8623

Tech support

Tech support

- They are huge ! Few Gig for each fabric devices (APIC or switch)
- They do contain Log of almost everything:
 - DME transaction between APICs, APIC and leaves/spine , leaves to NXOS process including all MO created, modified or deleted
 - On leaf NXOS sh tech with all nxos traces
 - On leaf, Persistent storage info (boot log info, epld log, install log, ...)

Getting show tech – Method 1

Techsupport local on **any device (apic or leave)**

DO NOT USE that one unless there is no other choice.

This is not as complete as the 2nd option

```
admin@pod2-apic1:~> techsupport local
```

```
Running bash commands
```

```
Completed 1 of 10 commands
```

```
...
```

Getting show tech – Method 2

Policy trigger Tech support – Always use that if possible.
Can be triggered via GUI/API

The screenshot shows the Cisco ACI Controller UI interface. The top navigation bar includes tabs for System, Tenants, Fabric, VM Networking, L4-L7 Services, Admin (circled in red), and Operations. Below the navigation bar is a secondary toolbar with links for AAA, Schedulers, Historical Record Policies, Firmware, External Data Collectors, Config Rollbacks, Import/Export (circled in red), and a refresh icon.

The main content area displays the "TechSupport Export Policy - default" configuration. On the left, a sidebar titled "Import/Export" lists options like Quick Start, Import Policies, Rollback Policies, Export Policies, and TechSupport. Under TechSupport, a tree view shows nodes such as default, On-demand, default, supN, supN, supNode103, supNode104, supNode201, supNode202, and ts_exp_pol. The "On-demand/default" node is selected, and a context menu is open, with the "Collect Tech Supports" option highlighted and circled in red.

The right panel shows the "Properties" section for the "default" policy. It includes fields for Name (set to "default"), Export to Controller (unchecked), Include pre-upgrade logs (checked), Export Destination (set to "bru-fs"), Scheduler (set to "select a value"), Specify TechSupport Time Range (unchecked), and Category (set to "All").

Getting show tech – Method 2

Node ID	Status	Detail Status	Collection Time	Export Location	Logs Location
Collected at: 2016-03-08 03:30 (3 out of 3 successful)					
1	green checkmark	Task completed.	2016-03-08 03:30	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic1_ssid-1_2016-03-08T02-03CET_1of3...	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod...
2	green checkmark	Task completed.	2016-03-08 03:30	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic2_ssid-2_2016-03-08T02-03CET_1of3...	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod...
3	green checkmark	Task completed.	2016-03-08 03:30	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic3_ssid-3_2016-03-08T02-03CET_1of3...	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod...
3 nodes	green checkmark				
Collected at: 2016-04-18 18:10:56 (0 out of 3 successful)					
1	pending	Running bash comm...	2016-04-18 18:10:56	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic1_ssid-1_2016-04-18T18-10CEST_10...	
2	pending	Running bash commands	2016-04-18 18:10:56	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic2_ssid-2_2016-04-18T18-10CEST_10...	
3	pending	Running bash comm...	2016-04-18 18:10:56	scp://10.48.37.151//data/03/roland/LAB/tsexp-default_pod2-apic3_ssid-3_2016-04-18T18-10CEST_10...	
3 nodes	Pending				

Result will either by local (click on link) or send to remote destination (if preconfigure)

Name	Node ID	Status	Detail Status	Collection Time	Export Location	Logs Location	DB Location
supNode101	101	pending	Transferred datab...	2016-04-18T...	files/2/techsupport/dbgexp_tsod-supNode101_pod2...		files/2/techsupport/dbgexp_tsod-supNode101_pod2...
ts_exp_pol	1	success	Task completed	2016-04-18T...	files/1/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...	files/1/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...	files/1/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...
ts_exp_pol	2	success	Task completed	2016-04-18T...	files/2/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...	files/2/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...	files/2/techsupport/dbgexp_tsod-ts_exp_pol_pod2-a...
ts_exp_pol	3	success	Task completed	2016-04-18T...	files/3/techsupport/dbaexp_tsod-ts_exp_pol_pod2-a...	files/3/techsupport/dbaexp_tsod-ts_exp_pol_pod2-a...	files/3/techsupport/dbaexp_tsod-ts_exp_pol_pod2-a...
ts_exp_pol	1	failed	Previous techsupp...	2016-04-18T...			
ts_exp_pol	2	failed	Previous techsupp...	2016-04-18T...			

Getting show tech – Method 2

Policy trigger Tech support – Always use that if possible.
Can be triggered by cli only from APIC

```
admin@pod2-apic2:~> techsupport
Usage: techsupport switch <nodeid> ([status] | [remotename <fname>])
      techsupport controllers [status]
      techsupport controllers remotename <fname>
      techsupport local
      techsupport all [status]
      techsupport all remotename <fname>
      techsupport remote (list|name) [<fname>] (delete|[(<host> <remoteport> <protocol> <username>
<remotepath>) ])
admin@pod2-apic2:~> techsupport controllers
Triggering techsupport for APIC using policy ts_exp_pol, setting filters to default value
Triggered on demand tech support successfully for controllers, will be available at: /data/techsupport on
the controller.
Use 'status' option with your command to check techsupport status
admin@pod2-apic2:~> techsupport switch 101
Triggering techsupport for Switch 101 using policy supNode101, setting filters to default value
Triggered on demand tech support successfully for node 101, will be available at: /data/techsupport on the
controller.
Use 'status' option with your command to check techsupport status
admin@pod2-apic2:~>
```

What's in Tech support on switch

- 3 files :
 - Part 1 – contains /mnt/pss and some logs. Mandatory for booting/upgrade issue
 - Part 2 – Full DB
 - Part 3 – Logs – DME logs, NXOS show tech including log – ACI specific process log (epm, epmc, eltm, eltmc,...)
 - Sh tech nxos is located in /var/sysmgr/tmp_logs/svc_ifc_techsup_nxos.tar

What's in Tech support on APIC

- 3 files :
 - Part 1 – contains pre upgrade log + some general info (disk usage, acidiag , version, audit and even log,...)
 - Part 2 – Full DB
 - Part 3 – Logs – DME logs

What / when ?

- Get tech support Always the 3 APIC
- Not necessarily all leaves / depending on the problem pattern...
- For Datapath issue any switch possibly involved is crucial (epm trace) – typically Border leaves and involved Server leaf for example

When ?

- AS quickly as possible after an issue , log on some process may wrap very quick.