VPLS, PPB, EVPN and VxLAN Diagrams

Contents

1. VPLS Signalling: An overview of how VPLS is signalled to create the pseudowires and how the different labels are chosen.
This based on the following document: *VPLS with BGP Signalling - Cisco TAC Document ID 116121*

2. VPLS Issues: Common issues experienced within a VPLS setup.

3. PPB Switching: A look at the path a packet will take through a PBB Switched network, including the different labels and identifiers used. This is partially based on http://www.tatacommunications.com/vpn/PBBknowledgeCenter/BRKSPG-2203.pdf. Any images marked with O are taken from the document, I claim no credit for their creation.

4. EVPN Overview: Shows the operation and principles involved in EVPN. This is partially based on https://conference.apnic.net/data/37/2014-02-24-apricot-evpn-presentation_1393283550.pdf. Any images marked with O are taken from the document, I claim no credit for their creation.

5. EVPN Operation: More detailed notes on the processes involved in EVPN.

6. PBB-EVPN: Some of the basic processes involved when the above two technologies work together (MAC learning and advertisements)

7. Inter-operation: A conceptual diagram showing how PPB-EVPN and VLPS technologies could interrelated in a Service Provider core.

8. VxLAN: A very brief overview of a VxLAN packet.

by Steven Crutchley

VPLS - BGP Signalling Operation

Based on VPLS with BGP Signalling - Cisco TAC Document ID 116121

If A BGP Router Reflector runs software that does not support RFC 4761, but does have support for RFC 4762, the special BGP ***neighbor x.x.x.x prefix-length-size 2*** configuration command is needed on the Route Reflector so it can reflect the BGP updates used for RFC 4761.

RFC 4762: VPLS
Discovery = BGP (auto)
Signalling = LDP

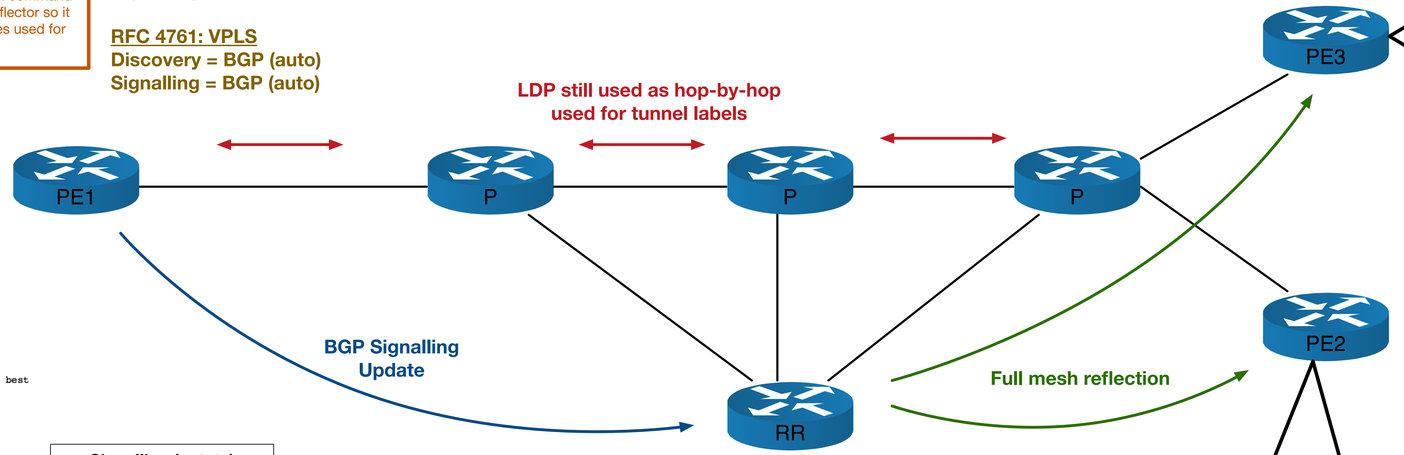
RFC 4761: VPLS
Discovery = BGP (auto)
Signalling = BGP (auto)

```
router bgp 1

l2vpn context ONE
 vpn id 100
 autodiscovered bgp singaling bgp
  ve id 1001
  ve range 50
  route-target export 32:64
  route-target import 32:64

mpls label range 10000 20000
```

```
PE1#show bgp l2vpn vpls rd 1:100 ve-id 1001 block-offset 1000
BGP routing table entry for 1:100:VEID-1001:Blk-1000/136, version 3
Paths: (1 available, best #1, table L2VPN-VPLS-BGP-Table)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  Local
    0.0.0.0 from 0.0.0.0 (10.100.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      AGI version(0), VE Block Size(50) Label Base(10000)
      Extended Community: RT:1:100 RT:32:64 L2VPN L2:0x0:MTU-1500
      rx pathid: 0, tx pathid: 0x0
```



If the process of selecting a label fails PE1 will notice, when it gets an update from PE3. Knowing its own advertised label range, it will see that the VE-ID will fall outside of the ***VBO <= VE-ID < (VBO + VBS)*** test. So in reaction, PE1 will send a new update with new VBO and LB values. It should now accommodate the VE-ID of PE3. PE3 may do the same in the opposite direction. Both advertised blocks will show in ***show bgp*** commands.

Blocks of contiguous VE-IDs are less likely to require second BGP messages to be sent.

```
PE1#show bgp l2vpn vpls rd 1:100 ve-id 1001 block-offset 1000
BGP routing table entry for 1:100:VEID-1001:Blk-1000/136, version 2
Paths: (1 available, best #1, table L2VPN-VPLS-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 1
  Local
    0.0.0.0 from 0.0.0.0 (10.100.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      AGI version(0), VE Block Size(50) Label Base(10000)
      Extended Community: RT:1:100 RT:32:64 L2VPN L2:0x0:MTU-1500
      rx pathid: 0, tx pathid: 0x0

PE1#show bgp l2vpn vpls rd 1:100 ve-id 1001 block-offset 10000
BGP routing table entry for 1:100:VEID-1001:Blk-10000/136, version 4
Paths: (1 available, best #1, table L2VPN-VPLS-BGP-Table)
  Not advertised to any peer
  Refresh Epoch 1
  Local
    0.0.0.0 from 0.0.0.0 (10.100.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      AGI version(0), VE Block Size(50) Label Base(10053)
      Extended Community: RT:1:100 RT:32:64 L2VPN L2:0x0:MTU-1500
        L2VPN L2:0x0:MTU-1500
      rx pathid: 0, tx pathid: 0x0
```

```
router bgp 1

l2vpn context ONE
 vpn id 100
 autodiscovered bgp singaling bgp
  ve id 1002
  ve range 50
  route-target export 32:64
  route-target import 32:64

mpls label range 3000 600000
```

Problem Statement

VPLS needs point-to-mutlipoint PWs. PEs within one VPLS Realm could be manually configured or discovered using BGP. But targeted LDP would still be needed for signalling. This diagram shows how to use BGP for signalling (RFC4761).

iBGP is used because of its full mesh requirement with Router Reflectors. There would be two methods to send updates:
1. Send one update per PW. But this goes to all PE routers and only one of them can use this information (the PE that is the other end of the PW in question)

2. To avoid a high level of updates, one local PE router sends a set/block of local VC labels to all remote PE routers. Each remote PE picks a VC label in a unique fashion, so that no other PE picks the same one. There must be enough labels and they must not be wasted.

This diagram describes this second method.

Signalling (octets)		
ID	PE Indentity	BGP sender - afi/safi 25/65. ***l2 router-id <id>***
NLRI	Length (2)	
	RD (8)	identity of the VPLS domain. If not configured format is ***ASN:vpn_id***
	VE ID (2)	(VE-ID) must be a unique to each PE to identify it within the VPLS domain
	VE Block Offset (2)	(VBO) gap used if more than one block needs to be sent. VBO = RND (VE-ID/VBS) * VBS
	VE Block Size (2)	(VBS) size of the block set (default 10). ***ve range***
	Label Base (3)	(LB) first free label in the block
Communities	Extended Comm Type (2)	0x800A
	Encapsulation Type (1)	Encapsulation Type (1)
	Control Flags (1)	0-5 must be zero. 6 = C for control word. 7 = S for sequencing
	Layer 2 MTU (2)	
	Reservered (2)	
	Additional RTs (import, export…)	imports and exports from an L2VPN like MPLS L3VPN

RND = the maximum value out of "the division result, rounded down" or 1.

To select VC label from block sent by PE1, it must determine if VBO is within range of its configuration.
Note the VBO will be very close to the LB. This is essentially the range the PE1 is offering PE2.

VBO <= VE-ID < (VBO + VBS)

If this passes, VC label is determined using:

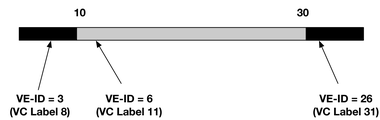
LB + VE-ID - VBO

Examples
VBS of 20, VBO of 5 and LB of 10. VBO + VBS = 25.
VE-ID of 6 succeeds. 5<=6<25
Label = 10+6-5=11

VE-ID of 26 fails. 26>25
Label = 10+26-5=31 (above block)

VE-ID of 3 fails. 3<5
Label = 10+3-5=8 (below block)

The successful label is used as the remote label in the output (10002 in the example shown)



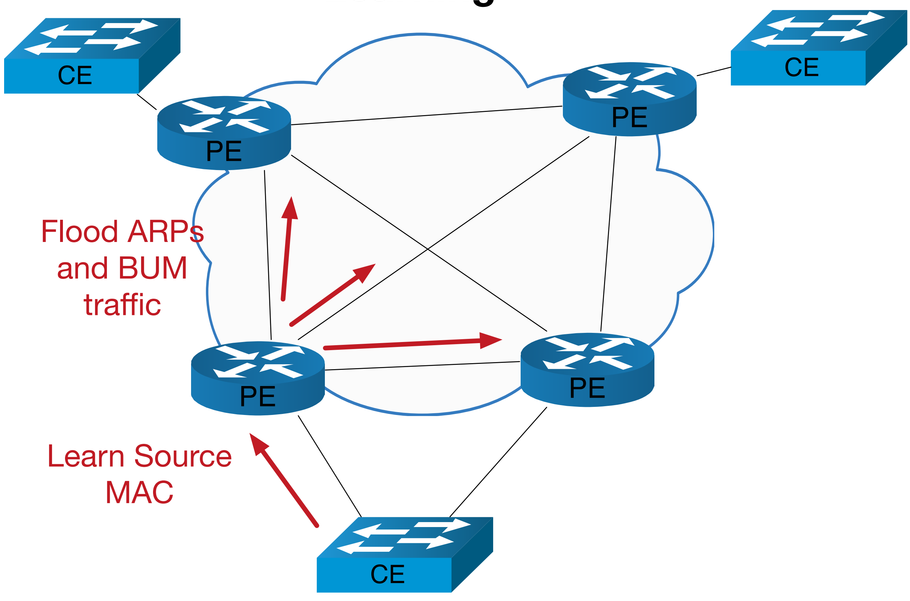

```
PE2#show l2vpn vfi name one
Legend: RT=Route-target, S=Split-horizon, Y=Yes, N=No

VFI name: one, state: up, type: multipoint, signaling: BGP
  VPN ID: 100, VE-ID: 1002, VE-SIZE: 50
  RD: 1:100, RT: 1:100
  Bridge-Domain 100 attachment circuits:
  Pseudo-port interface: pseudowire100001
  Interface          Peer Address    VE-ID  Local Label  Remote Label     S
  pseudowire100002   10.100.1.1      1001   3101         10002            Y
```

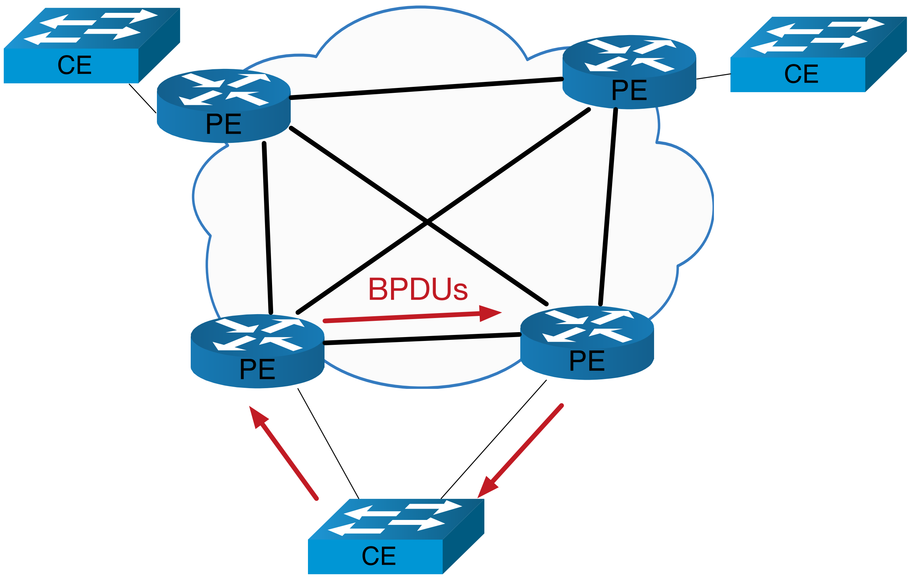
The same process is done in the opposite direction

VPLS Issues

MAC Learning

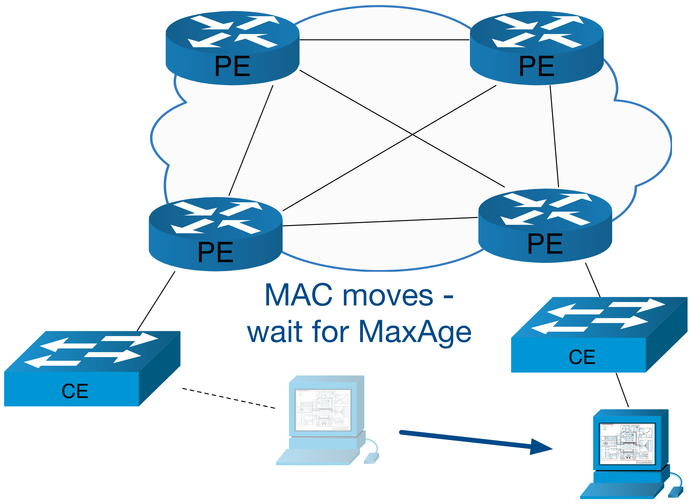


Multihoming

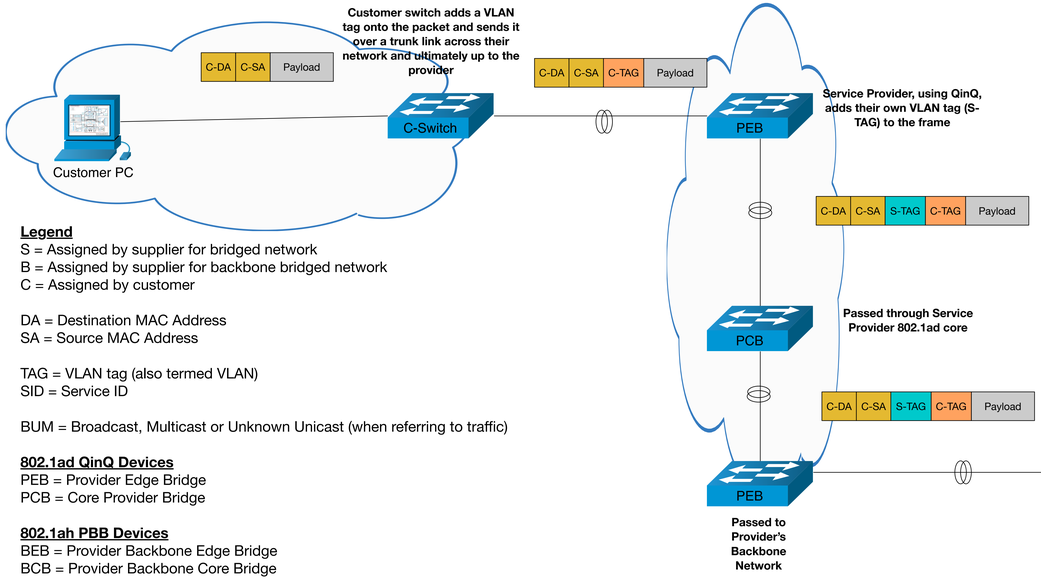


Multi homed CEs can get their own packet back from the core

MAC Move



Packet Path through Switched PBB Network



Legend
S = Assigned by supplier for bridged network
B = Assigned by supplier for backbone bridged network
C = Assigned by customer

DA = Destination MAC Address
SA = Source MAC Address

TAG = VLAN tag (also termed VLAN)
SID = Service ID

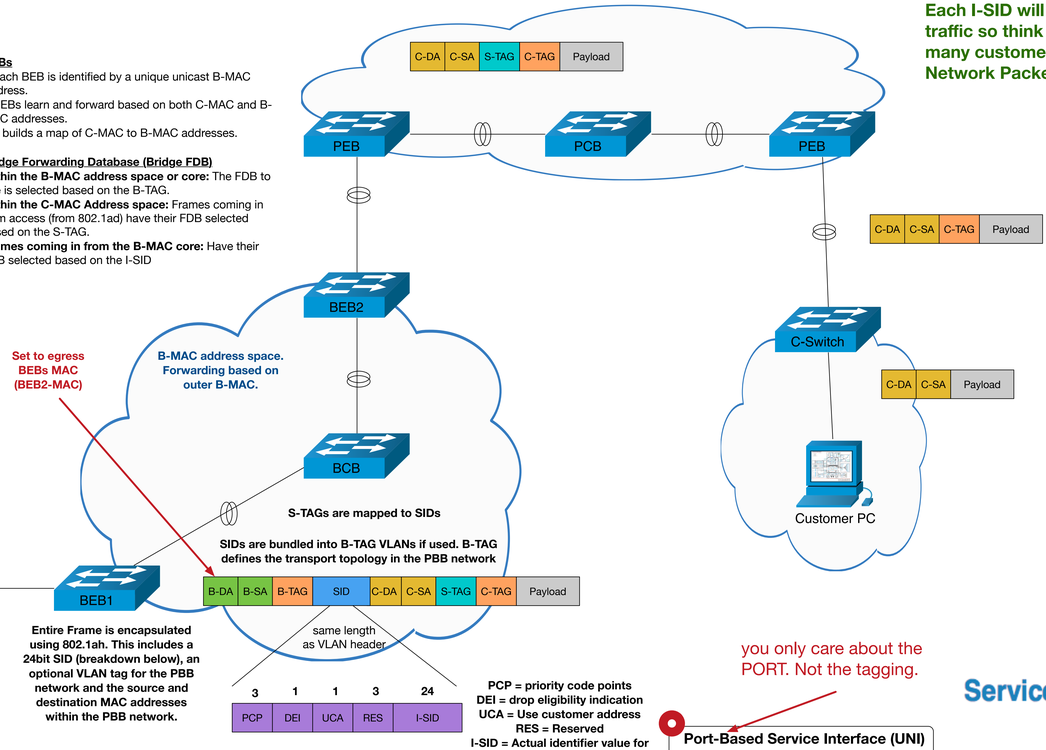
BUM = Broadcast, Multicast or Unknown Unicast (when referring to traffic)

802.1ad QinQ Devices
PEB = Provider Edge Bridge
PCB = Core Provider Bridge

802.1ah PBB Devices
BEB = Provider Backbone Edge Bridge
BCB = Provider Backbone Core Bridge

BEBs
> Each BEB is identified by a unique unicast B-MAC Address.
> BEBs learn and forward based on both C-MAC and B-MAC addresses.
> It builds a map of C-MAC to B-MAC addresses.

Bridge Forwarding Database (Bridge FDB)
Within the B-MAC address space or core: The FDB to use is selected based on the B-TAG.
Within the C-MAC Address space: Frames coming in from access (from 802.1ad) have their FDB selected based on the S-TAG.
Frames coming in from the B-MAC core: Have their FDB selected based on the I-SID

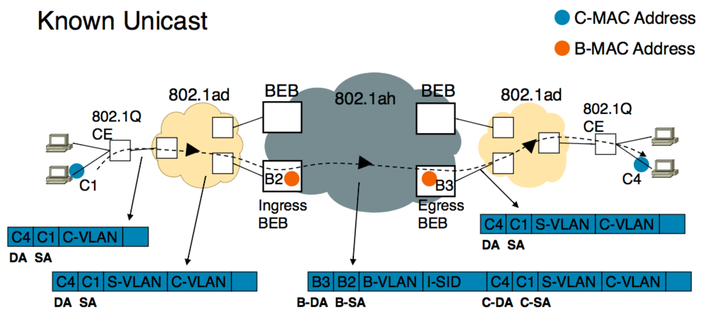


PCP = priority code points
DEI = drop eligibility indication
UCA = Use customer address
RES = Reserved
I-SID = Actual identifier value for the SID (Service Instance Identifier)

Each I-SID will have a multicast group for BUM traffic so think how this will affect things if many customers map to one I-SID (See BUM Network Packet Flow diagrams below)

Looking at S-TAG Service Interfaces (below) as an example, this depends on how customers are mapped to S-TAGs or SIDs. It could be one customer per S-TAG in the 802.1ad realm. If the provider doesn't want to separate them in the core you could use S-TAG port mode. Or give them each their own setup and use S-TAG Mode. Or some customers could use multiple S-TAGs (in which case you might what S-TAG bundle mode). Or maybe each customer *does* only have one S-TAG each and you want to give a fixed number of internal topologies rather than one each (also bundle mode) … similar to mapping VLANs to MSTP instances…

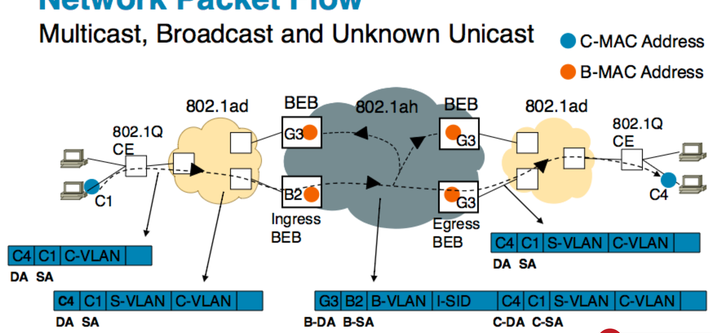
Network Packet Flow
Known Unicast



- Ingress BEB encapsulates frame with PBB header
  - B-MAC DA is set to egress BEB's MAC address (learnt via reverse traffic)
  - B-MAC SA set to ingress BEB's MAC address
  - I-SID determined based on S-VLAN & B-VLAN determined based on I-SID
- Egress BEB strips off PBB encapsulation

(In these diagrams the term TAG has been replaced with VLAN)

Network Packet Flow
Multicast, Broadcast and Unknown Unicast



- Ingress BEB encapsulates frame with PBB header
  - B-MAC DA is set to B-MAC multicast group address
  - B-MAC SA set to ingress BEB's MAC address
  - I-SID determined based on S-VLAN & B-VLAN determined based on I-SID
- One or multiple egress BEBs listen in to the group address

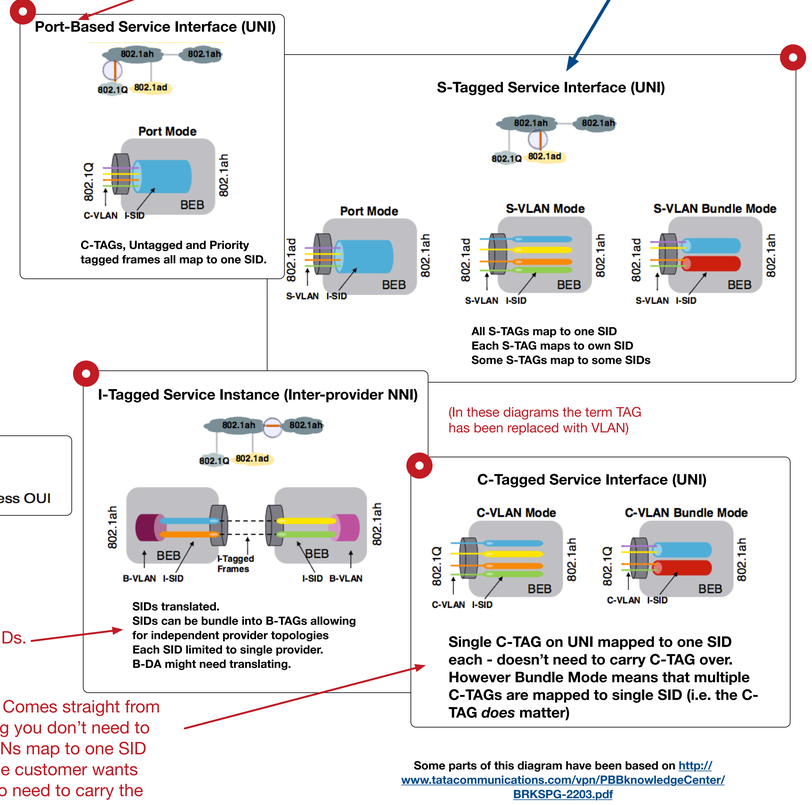
00-1E-83 | I-SID
Backbone Service Instance Group address OUI

Looking one level deeper just at I-SIDs.
I-TAG = SID + C-DA + C-SA

In this instance there ***is no*** 802.1ad network. Comes straight from customer VLANs. If it is a one to one mapping you don't need to worry about the VLAN tag but if multiple VLANs map to one SID (either because it is multiple customers or one customer wants different treatment for multiple VLANs) you do need to carry the VLANs.

you only care about the PORT. Not the tagging.

Service Interfaces



Some parts of this diagram have been based on http://www.tatacommunications.com/vpn/PBBknowledgeCenter/BRKSPG-2203.pdf

EVPN

EVPN ⟷ **L3VPN**
EVI ⟷ VRF

Multihomed devices can be…
> Single-Active: with one active PE
> All-Active: with multiple active PEs (will need split horizon and designated forwarding)


Device (PC, Switch, Router etc)

CE1

EVI 1

PE4

PE1

P

P

P

PE3

PE2

CE3

EVI 1

EVI 1

RR

MPLS or VXLAN control plane

LDP still used as hop-by-hop for tunnel labels

Data Plane learning can be static or dynamic

PE advertises MAC addresses and next hops from CEs using MBGP

Full mesh reflection

LAG (with All Active mode)

Overview
AFI = 25 (L2VPN), SAFI = 70 (EVPN).
ECMP from mutlihomed CEs is possible.
EVI is a VPN instance (like a VRF for L3VPN).
ESI is a link that connects the CE to the PEs.

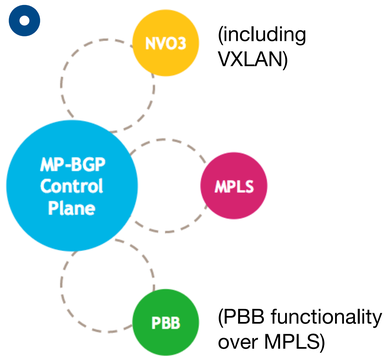
L2 and L3 services in one VPN.
Multiple Data Plane encapsulation models (MPLS or VXLAN).
ARP or ND proxy (PE responds on behalf of client)
No more flood-and-learn. Pre-Signalled FDB used instead.
You can control who learns what MAC (using policies).

This update will be one of the route types shown below

BGP Update (octets)	
RD (8)	Each EVI (just like a VRF) has an RD (?)
ESI (10)	Ethernet Segment Identifier
Ethernet Tag ID (4)	Broadcast domain (VLAN) for the EVPN
MAC Length (1)	
MAC Address (6)	48-bit MAC address
IP Length (1)	
IP Address (0, 4 or 16)	0 for no IP, 4 for IPv4 and 16 for IPv6
MPLS Label 1 (3)	MPLS Label for EVI (?)
MPLS Label 2 (0 or 3)	Label for split horizon BUUM traffic (?)
Ext comms…	

Route Type	Route Description	Route Usage
1	Ethernet Auto-Discovery (A-D) Route	Endpoint Discovery, Aliasing, Mass-Withdraw
2	MAC Advertisement Route	MAC/IP Advertisement
3	Inclusive Multicast Route	BUM Flooding Tree
4	Ethernet Segment Route	Ethernet Segment Discovery, DF Election
5	IP Prefix Route	IP Route Advertisement

EVPN Data Planes

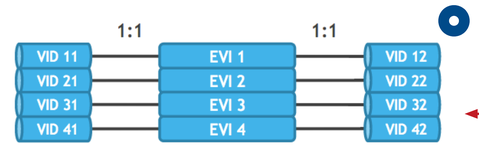


Extended Community Type	Extended Community Description	Extended Community Usage
0x06/0x01	ESI Label Extended Community	Split Horizon Label
0x06/0x02	ES-Import Route Target	Redundancy Group Discovery
0x06/0x00	MAC Mobility Extended Community	MAC Mobility
0x03/0x030d	Default Gateway Extended Community	Default Gateway

A VLAN WILL MAP TO AN EVI MUCH LIKE A VLAN (S-TAG or C-TAG) MAPPED TO AN I-SID

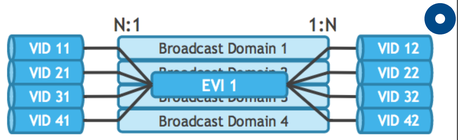
Services Overview

VLAN Based - one to one
1:1 mapping of VLAN to EVI.
VLAN translation allowed.
Single bridge domain per EVI.
Ethernet tag in route is set to 0.



VLAN Bundle - EVI doesn't care about the VLAN and multiple VLANs map to it
Multiple to one mapping of VLAN to EVI.
Still single bridge domain for each EVI.
MACs need to be unique across VLANs.
No VLAN translation.
Ethernet tag in route is set to 0.

VLAN Aware - EVI cares about what the VLAN is
Multiple to one Mapping of VLAN to EVI.
Multiple broadcast domains.
One bridge domain per VLAN.
VLAN translation allowed (look at left then right VIDs).
Ethernet tag is set to configured tag (VLAN).



One EVI per VLAN (possibly indicating each customer is represented by one VLAN. OR each customers VLAN gets an EVI). When carried across an EVI the "Ethernet Tag ID" isn't needed to differentiate.

Mapping multiple VLANs to one EVI… the only catch is that duplicate MACs could cause issues. VLANs are *not* carried across.

Mapping multiple VLANs to one EVI… but the VLAN is cared about, so you have one broadcast domain per EVI (e.g. the Ethernet tag is *not* zero) - possibly one customer with multiple VLANs.

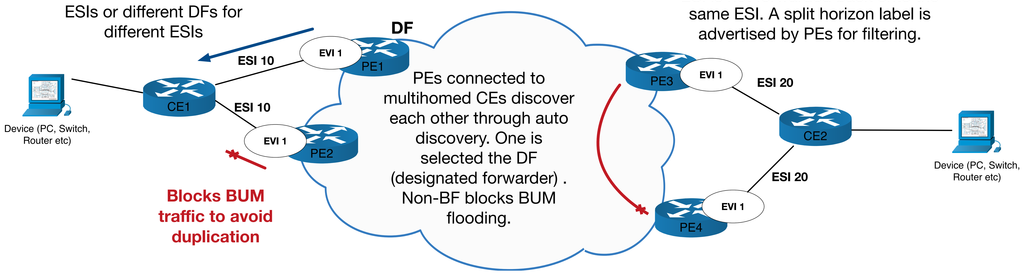
Some parts of this diagram have been based on https://conference.apnic.net/data/37/2014-02-24-apricot-evpn-presentation_1393283550.pdf

EVPN Operation

All-Active MULTIHOMING and SPLIT HORIZON

Can have same DFs for all ESIs or different DFs for different ESIs

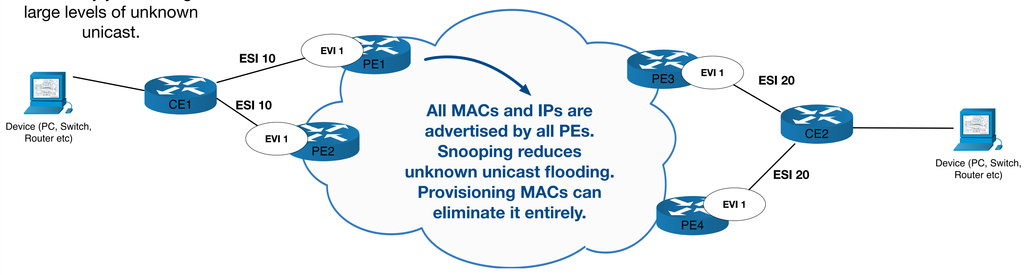
Split horizon = BUM traffic from one ESI is not forwarded back onto the same ESI. A split horizon label is advertised by PEs for filtering.



Blocks BUM traffic to avoid duplication

ARP/ND Proxy

You could have spoofed or untrusted sources… additionally you could get large levels of unknown unicast.



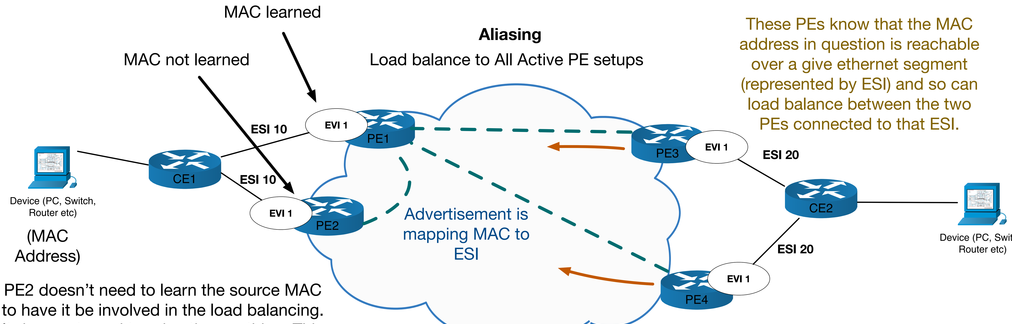
Aliasing

MAC learned

MAC not learned

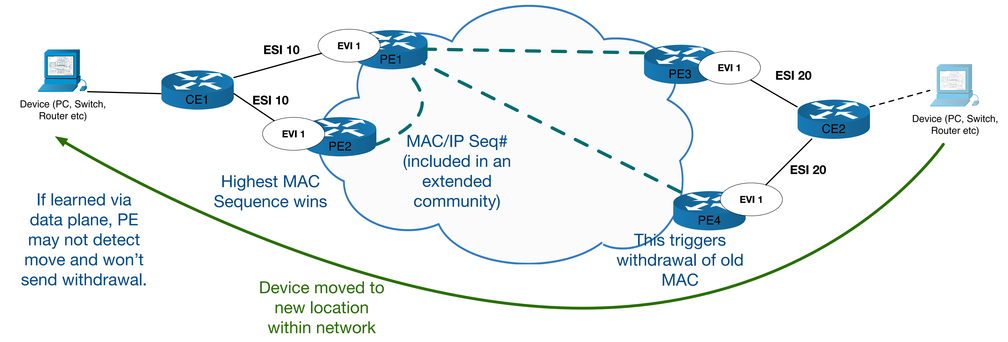
Load balance to All Active PE setups

These PEs know that the MAC address in question is reachable over a give ethernet segment (represented by ESI) and so can load balance between the two PEs connected to that ESI.

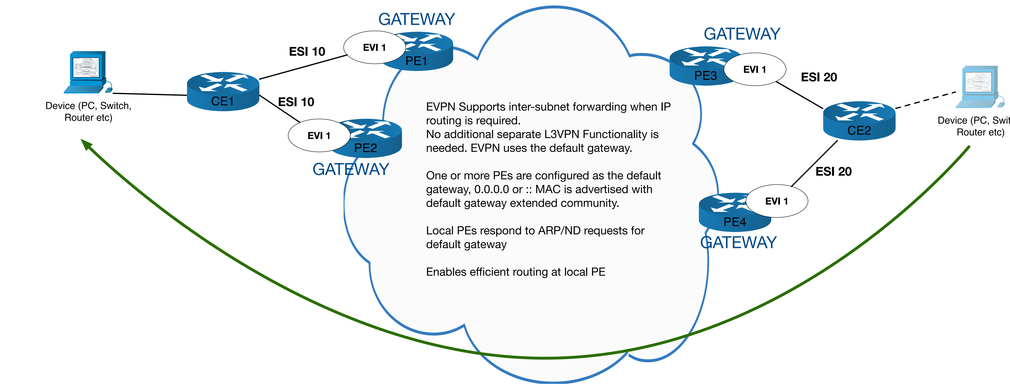


PE2 doesn't need to learn the source MAC to have it be involved in the load balancing. It does not need to advertise anything. This setup could also be used for Active-Standby

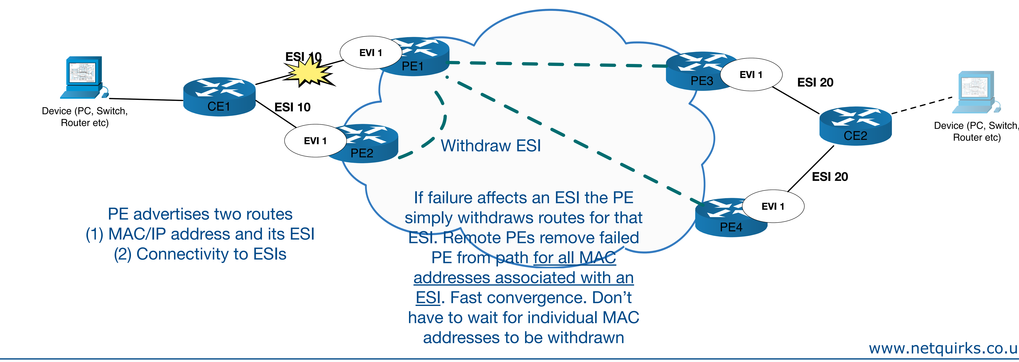
MAC Mobility



Default Gateway Inter-subnet Forwarding



MAC Mass-Withdrawal



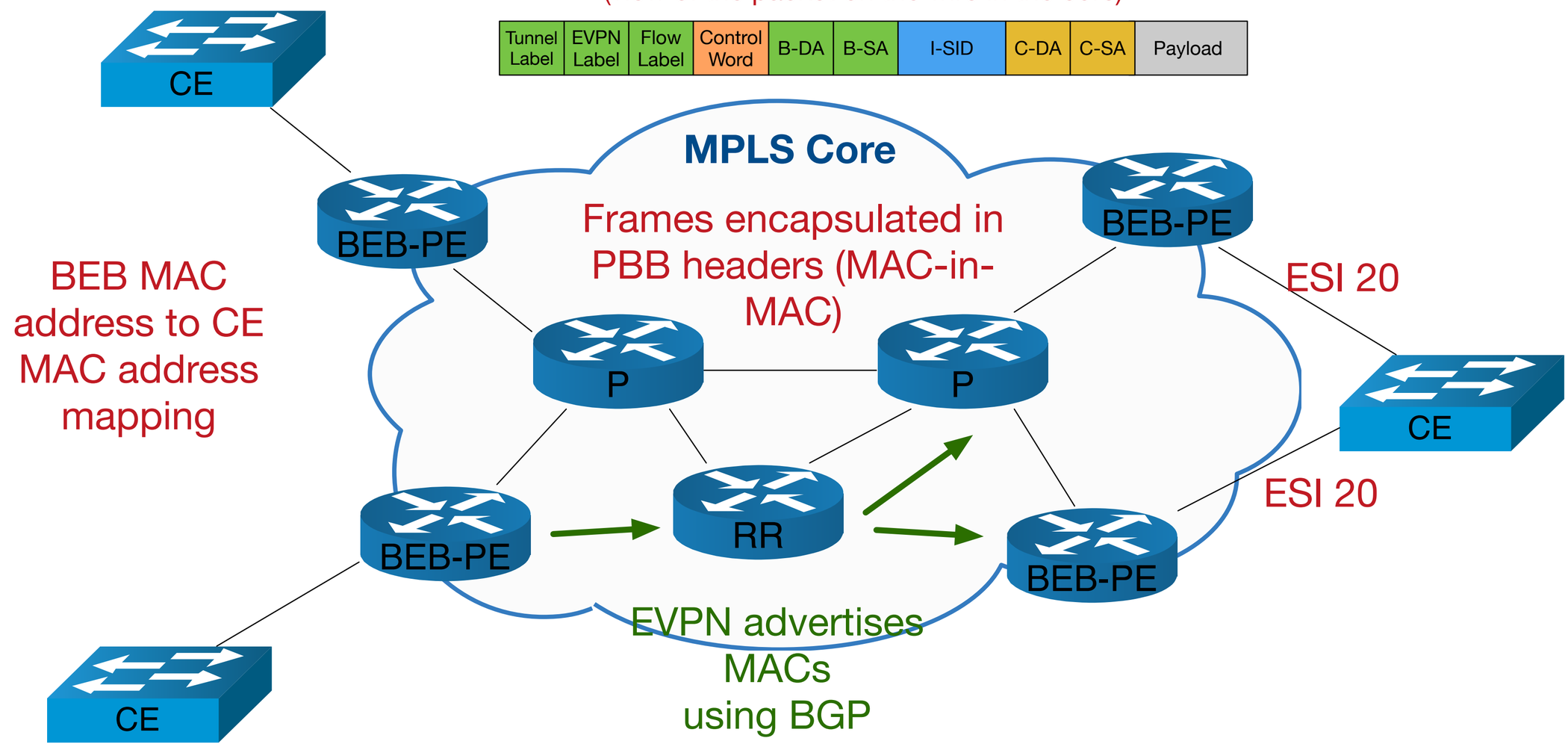
PE advertises two routes
(1) MAC/IP address and its ESI
(2) Connectivity to ESIs

If failure affects an ESI the PE simply withdraws routes for that ESI. Remote PEs remove failed PE from path for all MAC addresses associated with an ESI. Fast convergence. Don't have to wait for individual MAC addresses to be withdrawn

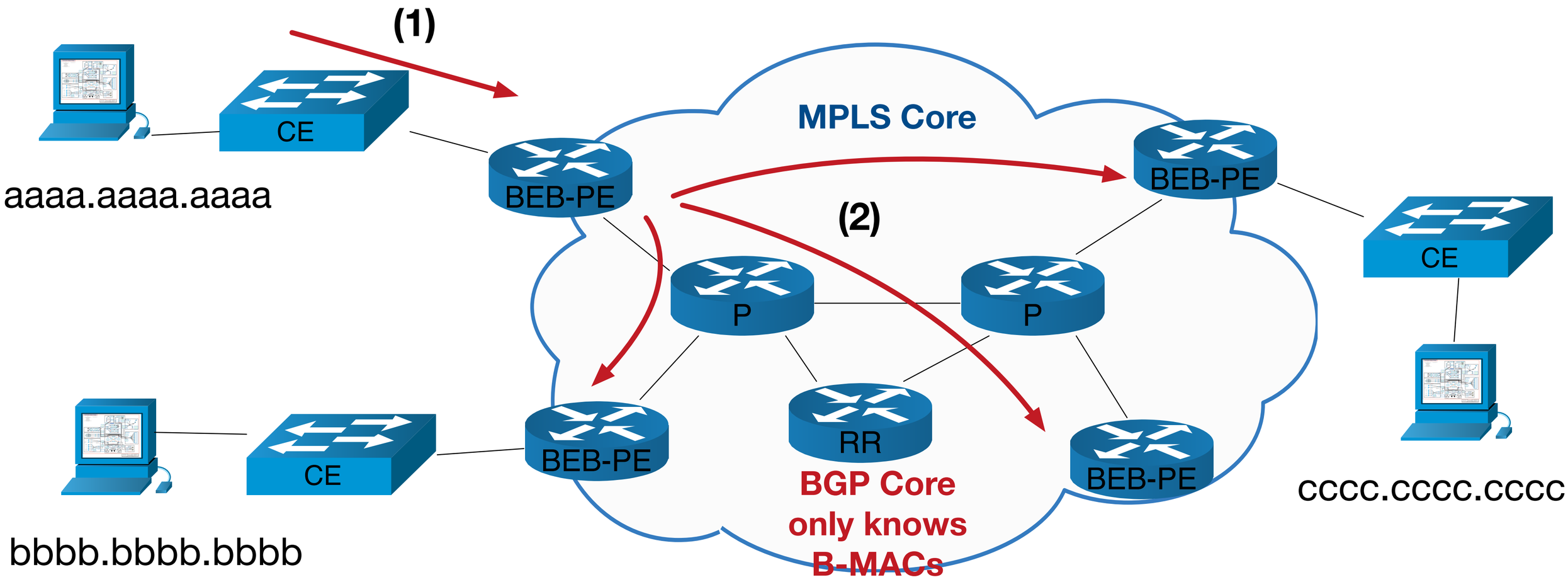
PBB-EVPN Overview

(view of the packet on the wire in the core)

Tunnel Label	EVPN Label	Flow Label	Control Word	B-DA	B-SA	I-SID	C-DA	C-SA	Payload

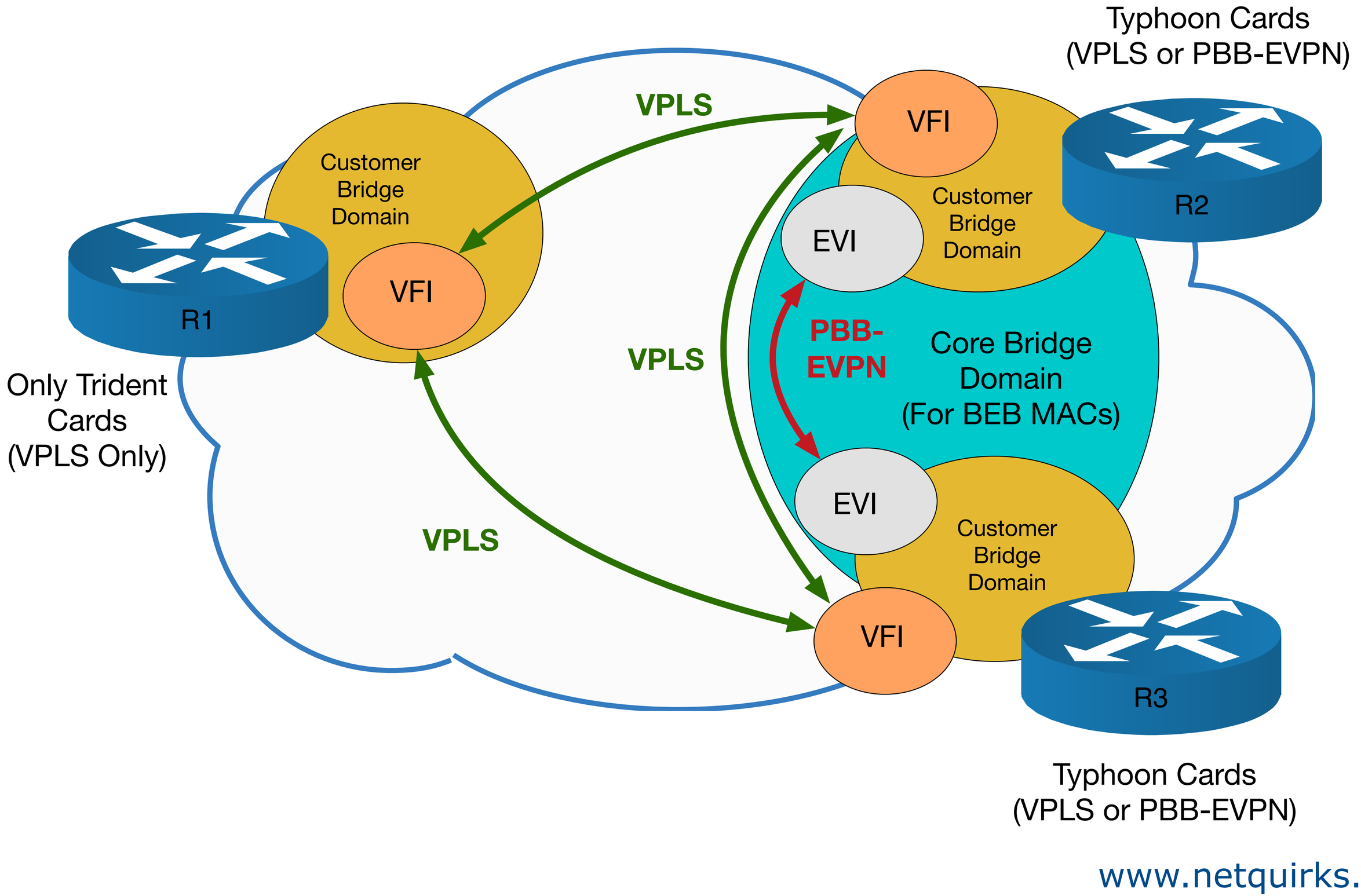


MAC Learning



(1) Packet goes to BEB1: Source MAC learning
(2) Doesn't know destination: Broadcast ARP. All remote PEs learn B-MAC that aaaa.aaaa.aaaa maps to. In this way a B-MAC to C-MAC table is built.

Inter-operation

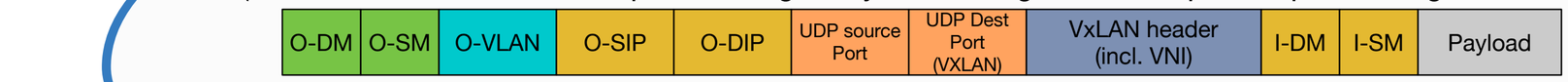


VxLAN Overview

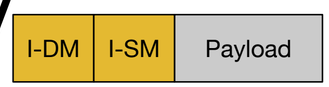
VxLAN is a Layer 2 overlay scheme on a Layer 3 network. Each overlay is termed a VXLAN segment.
Each segment has a 24 bit identifier called a VNI (VxLAN Network Identifier). A VNI is an outer header that encapsulates the inner MAC. VxLAN Tunnel End Points hide the VxLAN infrastructure from hosts and could be on physical or virtual switches or servers. Multicast carries BUM traffic (destination is multicast group for VNI segment)**. VTEP IP to VM MAC learning needs to take place (control plane learning like BGP with EVPN).** VTEPs must not fragment VxLAN packets.

DM = Destination MAC
SM = Source MAC
DIP = Destination IP
SIP = Source IP
I = Inner
O = Outer

(sections not to scale. Some part missing - only shown to give most important parts and general idea)



Look up VNI that this host is associated with. Is destination MAC on same segment and is there a mapping?



Upon receipt, VTEP will validate VNI and determine whether a host in that VNI matches inner MAC. If so it decapsulates it and sends it on.