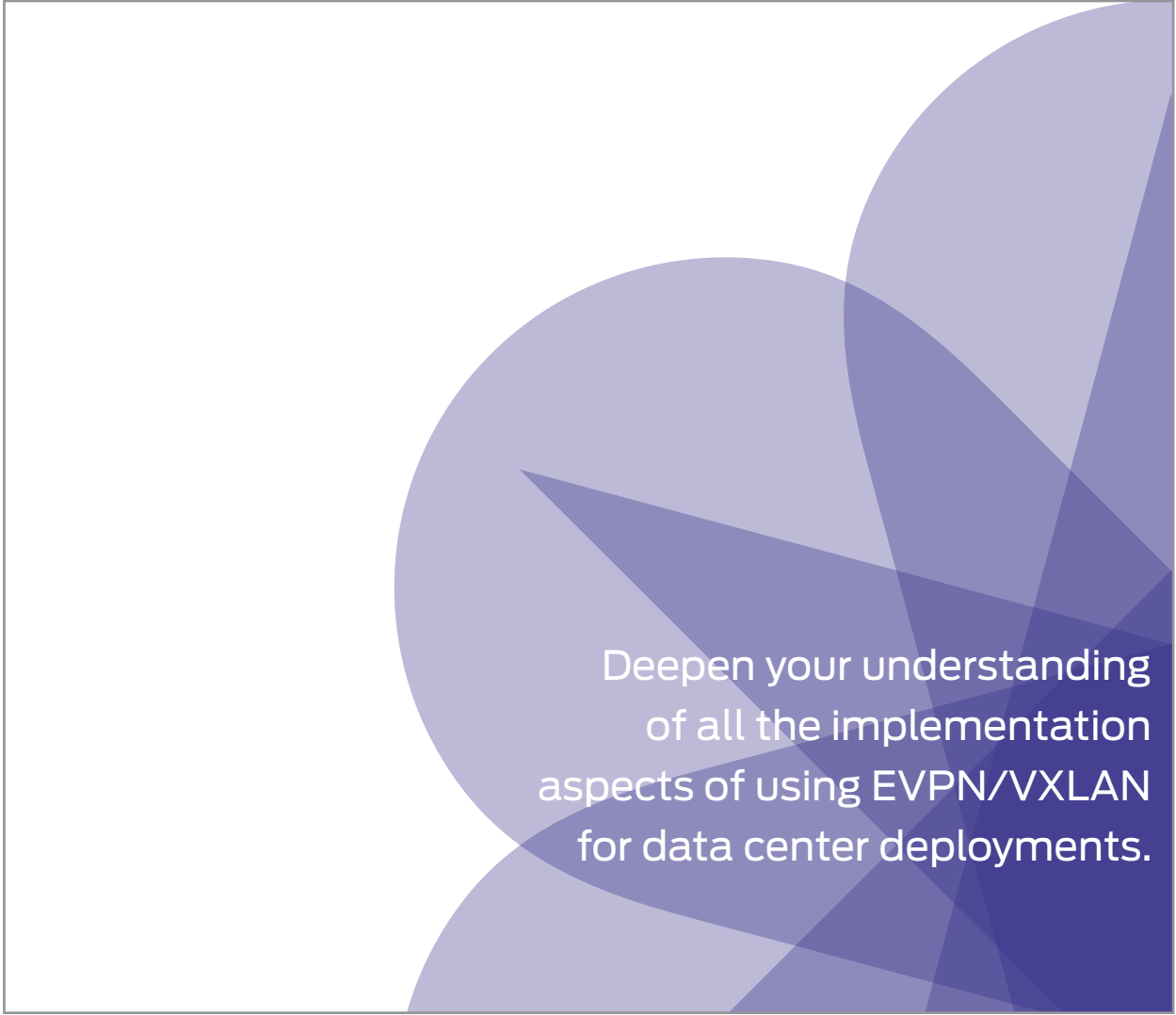


Data Center Technologies

THIS WEEK: DATA CENTER DEPLOYMENT WITH EVPN/VXLAN



Deepen your understanding
of all the implementation
aspects of using EVPN/VXLAN
for data center deployments.

By Deepti Chandra

THIS WEEK: DATA CENTER DEPLOYMENT WITH EVPN/VXLAN

This Week: Data Center Deployment with EVPN/VXLAN provides readers with a structured understanding of EVPN/VXLAN technology concepts and how they are implemented. The book includes detailed configuration examples, verification commands, and packet captures to demonstrate different traffic flows *within* a data center as well as for Data Center Interconnect (DCI). *This Week: Data Center Deployment with EVPN/VXLAN* contains five full-length case studies complete with illustrations, configurations, and field-tested insights from a senior data center architect.

"Expert guides on the practical applications of EVPN are few and far apart. Deepti brings you up close and personal to this powerful technology applied to the most common data center use cases. Grab your thinking cap and get ready to learn."

– Aldrin Isaac, Sr. Dir., Switching Solution Architecture, Juniper Networks, Co-Author of RFC7432

"Choosing the right set of options to achieve optimal network design in data center deployment requires deep understanding of EVPN/VXLAN technology as well as real world deployment experience. Deepti Chandra has worked with many experts at Juniper to come up with a concise set of guidelines and configuration examples that will help network architects build their data center fabrics quickly. The configuration examples in this book help define best-practice data center deployment."

– Sachin Natu, Sr. Dir., PM, Routing Applications, Juniper Networks

"This book is a valuable resource for those looking to explore or implement EVPN/VXLAN. The various deployment and configuration options are not only explained in great detail, but also backed up with validation in the lab."

– Victor Ganjian, POC Lab Engineer, Juniper Networks, JNCIP-DC,
Author of *Day One: Using Ethernet VPNs for Data Center Interconnect*

"Deepti Chandra shares her extensive knowledge and insights in this no-nonsense, practical guide for engineers and network architects alike."

– Wen Lin, Distinguished Engineer, Juniper Networks

"A thorough, well-organized, and insightful reference that doesn't miss a single detail. It's the new Bible for data center deployment!"

– Disha Chopra, Senior Manager, Product Line Manager, Juniper Networks

Published by Juniper Networks Books

www.juniper.net/books



JUNIPER
NETWORKS

This Week: Data Center Deployment with EVPN/VXLAN

By Deepti Chandra

Chapter 1	
Network Virtualization and Overlays	5
Chapter 2	
Building a Data Center – Routing at the Leaf Layer	51
Chapter 3	
Building a Data Center – Routing at the Spine Layer	99
Chapter 4	
Data Center Interconnect (DCI) – OTT DCI (L3VPN-MPLS Core)	157
Chapter 5	
Data Center Interconnect (DCI) – OTT DCI (Public Internet)	183
Chapter 6	
Data Center Interconnect (DCI) – Layer 3 DCI (L3VPN-MPLS Core)	213
Appendix	
Multicast/VXLAN Packet Walkthrough	232
Understanding OVSDb	234
OVSDb/VXLAN Packet Walkthrough	235
EVPN Service Interfaces	236
Building a Data Center - Routing at the Leaf Layer (Spine Devices Servicing Each POD)	245

© 2017 by Juniper Networks, Inc. All rights reserved.
 Juniper Networks and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo and the Junos logo, are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Published by Juniper Networks Books

Author: Deepti Chandra

Technical Reviewers: Aldrin Isaac, Bing Sun, Disha Chopra, Matt Mellin, Michael Pergament, Ryan Bickhart, Sachin Natu, Sachin Vador, Selvakumar Sivaraj, Victor Ganjian, Wen Lin

Editor in Chief: Patrick Ames

Copyeditor and Proofer: Nancy Koerbel

ISBN: 978-1-941441-57-2 (print)

Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-58-9 (ebook)

Version History: v1 August 2017

2 3 4 5 6 7 8 9 10

About the Author

Deepti Chandra is currently a Product Manager for the Routing Applications Group at Juniper Networks in Sunnyvale, California. She has multiple years of hands-on experience in versatile roles with pre-sales, operations and engineering, working closely with Data Center and Service Provider customers. Deepti holds M.S. degree in Telecommunications from University of Maryland, College Park.

Authors' Acknowledgments

A big thank you to all the technical reviewers for their time and valuable input that has helped me immensely to structure and improve the massive content of this book. Thank you, Aldrin Isaac, Disha Chopra, and Sachin Natu for your time and contributions that have been of great value in ironing out the details. A special acknowledgment and thanks to Wen Lin and Selvakumar Sivaraj: I have learned so much from you and I will always be grateful. I would also like to acknowledge and thank Ryan Bickhart, Bing Sun, Michael Pergament, and Matt Mellin who have provided support with all the questions I had in turning this into a field reference. I would also like to thank Mark Goedde for his diligence and support without which all the lab resources for the testing that went into this book would not have been possible. Thank you to Patrick Ames and his team for their patience and hard work, helping turn this into reality. Also, a special thank you to Victor Ganjian for his initial encouragement that motivated me to write this book, and also to Russell Kelly, Damien Garros and Eric Cheung Young Sen for their help during the initial concept phase. A big shout-out to Sachin Vador who has been instrumental in brainstorming and contributing to a lot of these case study scenarios. And finally, I can't thank my loving family enough, especially my Mom, for their continued inspiration and encouragement which continues to be my motivation for everything I do.

This book is available in a variety of formats. For more information see www.juniper.net/dayone.

Welcome to *This Week*

This Week books are an outgrowth of the extremely popular *Day One* book series published by Juniper Networks Books. *Day One* books focus on providing just the right amount of information that you can do, or absorb, in a day. *This Week* books explore networking technologies and practices that in a classroom setting might take several days to absorb. Both book series are available from Juniper Networks at: www.juniper.net/dayone.

Audience

This book is focused on EVPN-VXLAN and aims to provide the reader with a structured understanding of EVPN-VXLAN technology concepts and how they are implemented with detailed configuration examples, verification commands, and packet captures to demonstrate different traffic flows within a data center, as well as for DCI. Data Center Deployment with EVPN-VXLAN will help anyone starting to learn more about data center technology, from more experienced readers who are trying to explore different EVPN options for DCI solutions and traffic optimization, to engineers who need to understand the building blocks of designing and automating their data center deployments.

What You Need to Know Before Reading This Book

Before reading this book, you should be familiar with the administrative functions of the Junos operating system, including the ability to work with operational commands and to read, understand, and change Junos configurations. There are several books in the *Day One* library on learning Junos, at www.juniper.net/dayone.

This book also makes a few assumptions about you, the reader:

- You have a strong understanding of IP networking and experience with BGP and L3/L2 VPN technologies.

After Reading This Book

When you're done with this book, you'll be able to:

- Understand the fundamentals and implementation aspects of using EVPN-VXLAN for data center deployments.
- Design next generation data center architectures.
- Show detailed configuration examples, verification commands, and packet walk-through examples.

Terminology

Below is a brief summary of the terminology that is common throughout this book:

- *Leaf*: Rack switch (for example, Top of Rack or TOR) to which physical or virtual endpoints can connect to (for example, compute, storage or networking resources).
- *Spine*: Connect rows of racks providing reachability between leaf switches.
- *POD*: By definition, a POD is a modular defined set of compute and network resources that can be replicated to satisfy growth and expansion requirements. Here, a POD consists of leaf and spine devices serving a common pool of compute, storage, or network resources.

- *Fabric/Super Spine*: Provide connectivity between different spine devices thus connecting all PODs.
- *Service Node*: Refers to the devices that service each POD by directing intended traffic from the IP fabric to the service block.
- *Service Block*: Physical or virtual service appliances for example, firewall, load-balancer, etc.

Data center traffic flows are broadly categorized as:

- *East-West (E-W) traffic*: Comprises communication between applications hosted on physical servers or virtual machines that exist within a data center (Intra-DC) or across data centers (Inter-DC). Inter-DC traffic quite often addresses disaster recovery or resource optimization requirements across geographically dispersed data centers.
- *North-South (N-S) traffic*: Data center applications can be accessed from the WAN or the Internet, in which communication needs to be established between endpoints within a data center (southbound) and those out of the data center (northbound).

E-W traffic requirements can be either Layer 2 (intra-subnet) or Layer 3 (inter-subnet) in nature while N-S traffic requirements are commonly Layer 3.

NOTE A common question that arises on assessing deployment models is where should the Layer 3 gateway functionality be placed in a fabric: leaf or spine. Chapters 2 and 3, delve deeper into achieving E-W and N-S traffic requirements with both variations (routing at leaf or spine) and also show different options for firewall placement to achieve policy enforcement for tenant traffic.

Chapter 1

Network Virtualization and Overlays

Contents

<i>Network Overlay and Underlay</i>	<i>7</i>
<i>Relationship Between Network Overlay and Underlay</i>	<i>8</i>
<i>Understanding the Need for VXLAN</i>	<i>13</i>
<i>VXLAN Essentials</i>	<i>14</i>
<i>VXLAN Variants</i>	<i>15</i>
<i>Hardware VTEP Gateways</i>	<i>16</i>
<i>Understanding EVPN</i>	<i>17</i>
<i>EVPN-VXLAN Packet Walkthrough</i>	<i>22</i>
<i>Summary</i>	<i>49</i>

Virtualization is essentially the abstraction of physical resources to support multitenancy, allowing for fully isolated and distributed workload environments. Different computing resources like operating systems, storage devices, or network components can be virtualized to offer better scalability and flexibility. In modern data center architectures, server virtualization has allowed organizations to use consolidated compute infrastructures. As a result of this consolidation of software applications, operating systems and hardware platforms, fewer physical compute resources are needed providing huge cost benefits and many others advantages.

With server virtualization, server hypervisor provides software abstraction for compute resources, by replicating physical server attributes in software, allowing virtual machines to be created by using these attributes. Using the same hardware resources, multiple different operating systems instances hosted in virtual machines (guest OS) can be created. Network virtualization, where the virtualization layer functionally is equivalent to a network hypervisor, provides software abstraction for network resources by replicating network attributes (like routing, switching, firewall/micro segmentation, load balancing, etc.), allowing virtual networks to be created by using these attributes. Using the same physical network resources, multiple virtual networks to support multiple tenants can be created.

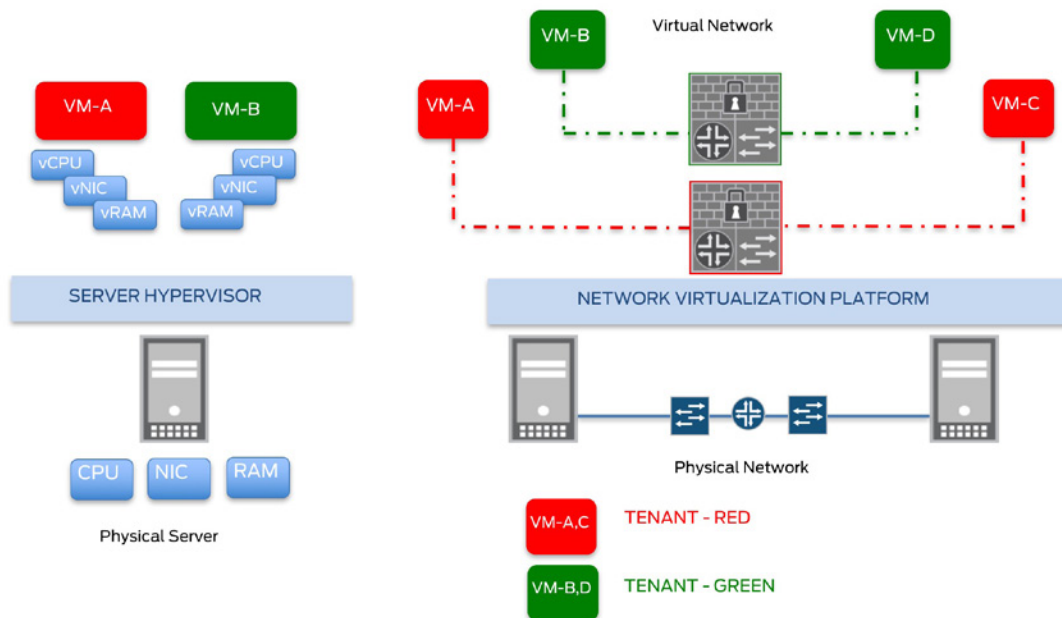


Figure 1.1 Difference Between Server and Network

However, data centers with server virtualization without any network virtualization have to face many network challenges. Network virtualization offers many benefits to ease integration of virtual and physical networks. Some of these include: automation to ease provisioning, elasticity to support dynamic changes and fully utilize deployed resources, security through complete isolation of network segments, and centralized management of both virtual and network infrastructure.

Network Overlay and Underlay

Network overlays and virtualized network services are important components that constitute network virtualization.

Virtual LANs (VLANs) have traditionally allowed for multi-tenancy by abstracting the LAN segment for network partitioning. Each VLAN corresponds to a unique IP subnet (separate broadcast domain). In legacy data center designs a VLAN is logically equivalent to a tenant or a service, so for example, accommodating 10 tenants would be equivalent to provisioning 10 different VLANs. VLAN trunks over the physical network infrastructure connect servers to allow for application communications that reside on virtual machines or containers.

With increasing application demands – be it for cloud computing or big data initiatives – scalability, elasticity, and ease of provisioning in hybrid (physical and virtual) environments have become critical data center requirements. With the use of VLANs, network resources become a bottleneck in realizing these requirements.

Outlined here are some challenges seen with legacy non-virtualized data center architectures that do not use any overlays:

- *Lack of flexibility and rigid provisioning:* Provisioning new tenants or making modifications to existing ones requires changes on both the virtual and physical infrastructure (starting from the server hypervisor through all ports on underlying network devices to add/change the respective VLAN). This requires collaboration between different teams that further impacts business agility.
- *Limited scale:* VLAN-ID field is 12 bits yielding a maximum of 4094 VLANs to create Layer 2 segments. This scale that can be achieved with VLANs is no longer sufficient to meet the increasing demands in a multi-tenant environment.
- *Restricted stability:* The use of VLAN trunks requires all devices to maintain state. MAC scalability is affected due to limits on the number of MAC addresses that can be learned per device. Unpredictable amounts of BUM traffic when the MAC scale limit is surpassed, further impacts resiliency.
- *Inefficient resource utilization:* Loop avoidance mechanisms like STP in a pure Layer 2 network, blocks redundant links, providing only a single active path, thereby not utilizing all available bandwidth resources. This also affects traffic distributions since multiple redundant paths cannot be leveraged for load balancing.
- *Support for Layer 2 extension and DCI (Data Center Interconnect):* To provide for a distributed workload environment while enabling business continuity, high availability, redundancy, and disaster recovery, multiple data centers are required for geographical diversity. Data centers are often interconnected over Wide Area Networks (WAN) to allow communication between application elements distributed across multiple data centers. Layer 2 extension refers to the ability to stretch Layer 2 across the WAN to allow Layer 2 connectivity between these dispersed applications. This Layer 2 connectivity across DCI is often required to support workload mobility requirements like VM migration (Layer 2 extension is needed since both source and destination data centers need to support the same IP subnet and extend VLANs for VM live migration). Other common use cases include geo-clustering to achieve disaster recovery (data centers hosting cluster members need to support same IP subnet and extend VLANs for clustered servers), high availability clusters (Layer 2 extension to facilitate heartbeat exchange and database replication) and supporting business continuity during server farm growth. In deployment scenarios where Layer 2 islands are separated by a Layer 3 core,

overlays can provide Layer 2 extension between data centers across the WAN which legacy architectures that use only VLANs cannot support.

Relationship Between Network Overlay and Underlay

The overlay network virtualizes the underlying physical infrastructure by creating logical networks over the physical underlay network. It decouples the offered service from the underlying transport and lifts the necessity for the transit devices to maintain state, which improves scalability. Maintenance of state is now confined only to overlay endpoints responsible for virtual tunnel encapsulation/decapsulation (for example, server hypervisor or TOR). Transit devices can have smaller forwarding tables as they forward traffic using the outer encapsulation or transport header that contains underlay addresses only. This decoupling offers the added advantage of faster virtual network provisioning without the need to configure the traversed physical devices.

The underlying infrastructure is transparent to application services which use the virtual tunnels provided by the overlay to communicate with each other. The underlay provides a service to the overlay by simply transporting encapsulated tenant data between compute resources. In other words, underlay is responsible for tunnel endpoint reachability while complexity of virtual tunnel management is handled by the overlay. If the underlay is structured for scale and resiliency, it can provide optimal traffic paths that the overlay can leverage for better traffic management and distribution.

The generic packet structure depicted in Figure 1.2 is only a logical representation of the packet headers mapped to network overlay and underlay. Subsequent sections will elaborate on what these specific fields for the stack of nested headers are, based on the overlay and underlay in use.

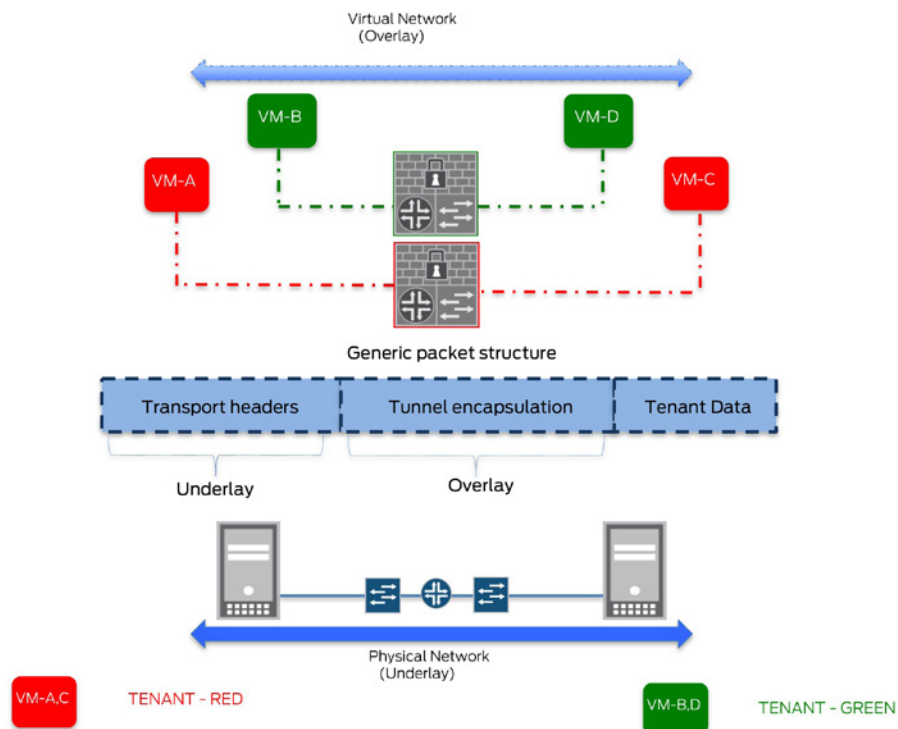


Figure 1.2

Relationship Between Network Overlay and Underlay

Types of Network Underlay

A data center fabric based on Clos architectures is defined as a matrix of tightly woven interconnections between network devices that resembles a fabric. It allows for a flattened architecture with equidistant endpoints and a non-blocking core to support low latency and high bandwidth for both East-West and North-South traffic. Data center fabrics are a popular choice for network underlay.

These could be broadly categorized as:

- *Ethernet fabric*: Ethernet fabrics allow you to manage the entire fabric as a single device. Examples of Juniper Ethernet fabric solutions are QFabric, Virtual Chassis Fabric (VCF), and Junos Fusion. The advantages they offer over the traditional multitier access-aggregation core networks include: no need for STP (as link state routing is used for loop avoidance allowing for a non-blocking architecture), support for multipath and load balancing, single pane of glass management, plug and play operation, and distributed forwarding with no control plane protocols. Disadvantages include fabric scale out limitations, as the system cannot scale beyond its defined capacity and not being an open standard.
- *MPLS fabric*: MPLS serves as the underlay to interconnect all devices in the Clos fabric. Physical architecture concepts that apply to an IP fabric also apply to an MPLS fabric. MPLS fabrics can take advantage of FRR and traffic engineering, amongst the rich set of MPLS features, which are well-known and widely deployed. However, MPLS underlay would need the network devices to be MPLS capable. Juniper products have good support for the same, but that might not be the case with all other network/server vendor equipment. MPLS signaling protocols like LDP, RSVP, or BGP-LU can be used to provide MPLS underlay.

LDP/RSVP offers the advantages of being well known, granular LSP statistics and reservation. Using MPLS fabric within a data center deployment would require a mesh of LDP/RSVP LSPs between the TORs (assuming these are overlay endpoints), which could result in limited control plane scale due to the maintenance of per-LSP state on transit devices. Other associated disadvantages include the increased overhead of managing multiple protocols and lack of deterministic labels. MSDCs (Massive Scalable Data Centers) prefer BGP (eBGP) as the only routing protocol due to the scale and stability benefits it offers (draft-ietf-rtgwg-bgp-routing-large-dc-11). Additionally, it also offers the advantages of reduced failure domains by confining L2 broadcast domains, better multipath, and loop-detection mechanisms. BGP-LU (labeled unicast) can also serve as transport by mapping infrastructure address (server reachability) to MPLS labels (within an AS or across multiple AS).

Alternatively, Segment Routing or SPRING (Source Packet Routing In Networking) can be used for label distribution. Both IGP (ISIS – sub TLVs and OSPF – opaque LSAs) and BGP (BGP-LU SPRING extensions) support SPRING. SPRING uses source routing to put the forwarding state in the packet itself. It allows a node to specify the optimal path to be traversed, based on application requirements instead of the shortest path. Label stack is equivalent to an ERO and is used by the ingress node to explicitly specify the path to be traversed. SPRING offers the advantages of being an ECMP-aware, deterministic, label computation that offers ease of troubleshooting and integration with an external controller, scalable control plane state, and lesser operational overhead. More state would be needed to be added to the packet as MPLS labels if more granularity with regard to the path to be traversed is required. In this case, label stack depth support might become a concern.

- **IP fabric:** IP serves as the underlay to interconnect all devices in the Clos fabric, resulting in an open standard, flat forwarding, horizontally scalable, and non-blocking spine/leaf architecture. *Leaf* and *spine* are roles assigned to fabric network devices that are connected in a hierarchical design (3-stage Clos) and transport overlay tunneled traffic based on routing the outer IP header. Leaf devices provide connectivity to access while spine devices act as aggregation points. Common multi-stage IP fabric models include 3-stage Clos (three tier: stages 1 and 3 (leaf), stage 2 (spine)), and 5-stage Clos (three tier: stages 1 and 5 (leaf), stage 2 and 4 (spine), and stage 3 (fabric/super spine to provide connectivity between the spine switches)). See Figure 1.3.

Being an open-standard allows for cross-vendor support. IP Clos fabrics are horizontally scalable since more leaf or spine devices can be added to accommodate for capacity increase without any major network redesign to support increasing East-West traffic. Each leaf device is connected to every spine device with routed links between leaf/spine devices. As a result, the *blast radius* is considerably reduced due to the localized failure domains, since failure of any one node does not bring the entire fabric down. Multiple paths (ECMP) are now available between any two leaf devices through the multiple connected spine devices available, enabling redundancy, resiliency and efficient traffic load-balancing over all available paths. Being L3 fabrics, these are inherently loop free. However, there is no single pane of glass management. This can be overcome through the use of automation tools or frameworks that can accommodate operational changes and simplify the provisioning process.

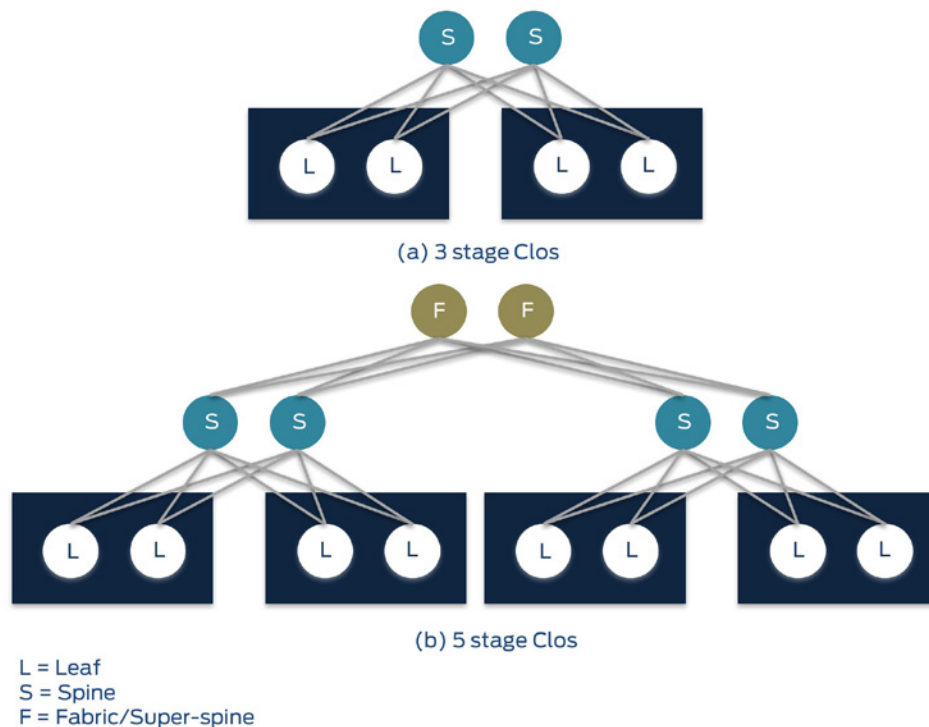


Figure 1.3 3-stage/5-stage Clos

- Popular choices of routing protocols to provide for the control plane of an IP fabric include:
 - **IGP (OSP/ISIS):** Link state protocols like OSPF/ISIS use flooding and with increase in scale, LSDB sizes and unnecessary churns might become a bottleneck.

However, though areas can help alleviate flooding scope, a contiguous Layer 2/ backbone hierarchical design requirement can introduce design complexity. Though these are easy to configure, there is operational overhead in managing multiple protocols to connect to the WAN, since BGP is a preferable option used in the core. Also, traffic engineering capabilities are limited (common use cases like shifting traffic away from a spine node during a maintenance window becomes challenging with IGP-based fabrics).

- *iBGP*: iBGP requires the need to have a full mesh of peering sessions which has limited scalability due to increased state and configuration/operational overhead. BGP route reflection can help mitigate this, but based on BGP path selection algorithm, route reflectors will advertise only the single best path to its clients. In this situation, add-path functionality can provide ECMP with iBGP. This imposes additional feature requirements. Hierarchical route reflection can provide better performance with increased scale, but again, this adds to design complexity.

- *eBGP*: eBGP has better inherent loop avoidance (AS_PATH) and multipath capabilities. Each leaf/spine node can be in a separate AS resulting in distributed routing domains. Use of 4-byte AS can prevent the exhaustion of AS numbers utilized by the fabric devices. It has been built to scale to support large number of peers and prefixes along with the ease of reduced protocol interactions if BGP is used within the DC and across the WAN. Better traffic engineering capabilities make migration of traffic easier through BGP attribute manipulation (for example, to move traffic away from a spine under maintenance as-path-prepend or communities can be used). There is configuration complexity involved with the proper configuration of policies and peers, however, automation tools can facilitate provisioning for the same.

Types of Network Overlay

Overlay networks create communication channels between tenant applications/services through the creation of logical/virtual tunnels. These can support L2 (L2 overlays – extend L2 network so VMs can share the same IP subnet) and L3 (L3 overlays – share L3 networks) transport between services while hiding the same from the underlying network infrastructure. Tunnel encapsulation consists of a tenant ID which is a function of the overlay network and acts as a demultiplexor used to distinguish between different traffic streams carried over the tunnel, each stream representing a different service.

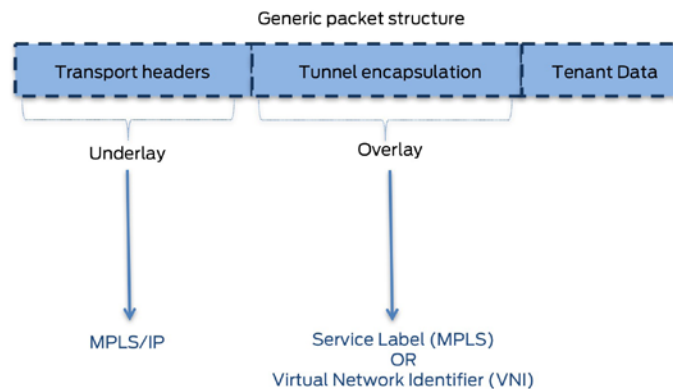


Figure 1.4

Generic Packet Structure Fields for Overlay/Underlay

Each communication channel consists of a control plane to exchange tenant application topology information (MAC addresses/IP routes) and data plane encapsulation to encapsulate and forward traffic between overlay tunnel endpoints across the virtual tunnel.

Examples of data plane encapsulation include MPLSoMPLS, MPLSoGRE, MPLSoUDP, and VXLAN:

- **MPLSoMPLS:** This data plane encapsulation employs an MPLS aware underlay. The label stack is composed of the inner service MPLS label and outer transport MPLS label. Though there are traffic engineering advantages with MPLS, there is operational overhead associated with the same, particularly for usage in data centers.
- **MPLSoGRE:** Useful when the transit devices are not MPLS aware. Inner service MPLS label is encapsulated in GRE and transported over a physical IP infrastructure. Load balancing capabilities are limited since network device implementations need to understand GRE key field being used in a load-balancing context.
- **MPLSoUDP:** Similar to MPLSoGRE, this is an IP based encapsulation for MPLS packets that uses a UDP header instead. Most existing routers in IP networks are already capable of distributing IP traffic over ECMP and/or LAG, based on the hash of the five-tuple of UDP and TCP packets (therefore, source IP address, destination IP address, source port, destination port, and protocol). By encapsulating the MPLS packets into an UDP tunnel and using the source port of the UDP header as an entropy field (hash of the payload), the existing load-balancing capability can be leveraged to provide fine-grained ECMP load balancing of MPLS traffic over IP underlay networks.
- **VXLAN:** Can be used to transport data packets over an underlying IP networks that provides connectivity between VXLAN Tunnel End Points (VTEPs). VXLAN uses a MAC in UDP encapsulation to extend Layer 2 segments, thus providing Layer 2 overlay over a Layer 3 network. Due to scalability and complexity concerns with the multicast flooding mechanisms used for MAC learning, OVSDb and EVPN are commonly deployed as control planes (more details in subsequent sections). Since there is no intrinsic support for security with VXLAN (or UDP or GRE), IPsec can be used to ensure data privacy when the underlying IP transport is untrusted.

Examples for control plane include OVSDb, EVPN:

- **OVSDb:** OVSDb is a management plane protocol that defines data and how it is exchanged, but not what to do with it. Primarily designed for centralized controller-based management functions to support programmability, it can configure the hardware devices and import/export dynamically-learned addresses. Intelligence is contained in the database schema. It lacks the ability to make independent route decisions to influence forwarding state. EVPN on the other hand is based on a control plane routing protocol like MP-BGP and can offer a distributed control plane or operate with a centralized controller. BGP has inherent support for scalability and attributes that constitute a policy-based framework. This makes it possible to realize a myriad of rich features.
- **EVPN:** MAC learning between PEs is achieved using the control plane. It offers flexible choices for data plane encapsulation and MP-BGP control plane (different EVPN route types) supports the exchange of MAC/IP addresses. Support for all-active multihoming allows for better utilization of multiple CE-PE links for traffic distribution and high availability. Automated DF election, discovery of multi-homed PEs and auto-discovery of neighbors in the same VPN allow for easy and simplified

provisioning. Fast convergence, Aliasing, MAC Mass withdraw, MAC mobility, support for advertising /32 host routes enabling traffic optimization and VM mobility (VMTO), ARP suppression, and distributed anycast gateway support are some of the many EVPN benefits that will be elaborated upon later in this chapter.

Understanding the Need for VXLAN

VXLAN (Virtual eXtensible Local Area Network) is a Layer 2 overlay technology over a Layer 3 underlay infrastructure. It provides a means to stretch the Layer 2 network by providing a tunneling encapsulation using MAC addresses in UDP (MAC in UDP) over an IP underlay. It is used to carry the MAC traffic from the individual VMs in an encapsulated format over a logical “tunnel.” Outlined below are some of the requirements associated with the data center deployments addressed by VXLAN:

- *Higher scalability for multi-tenancy:* A key characteristic of Layer 2 data center networks is their use of Virtual LANs (VLANs) to provide broadcast isolation. A 12-bit VLAN ID is used to identify a Layer 2 segment and is used in the Ethernet data frames to divide the larger Layer 2 network into multiple broadcast domains. In the case when the VMs in a data center are grouped according to their VLAN, thousands of VLANs might be required to partition the traffic according to the specific group to which the VM may belong. With the growing adoption of virtualization, requirements for multi-tenant environments accelerate the need for larger VLAN limits. As a result, this upper limit of 4094 is seeing pressure and is inadequate in such situations. VXLAN addresses this limitation by using a 24-bit VXLAN Network Identifier (VNI) to identify a Layer 2 segment providing an increased namespace of up to 16M (2²⁴) identifiers. This alleviates the bottleneck associated with VLANs as with VXLAN up to 16M isolated networks for that many tenants can now be supported over the same shared IP underlay.
- *Optimal resource utilization:* STP accomplishes loop avoidance by blocking redundant paths. Apart from resiliency due to multipathing not being available with the STP model, it also poses a challenge for network operators who are paying for additional ports and links but are unable to use the same. VXLAN can take advantage of the IP fabric underlay, which eliminates the need for disabling network links to avoid loops and helps achieve multipath through ECMP. Furthermore, the UDP source port entropy ensures efficient load balancing of VXLAN traffic these across equal cost paths.
- *Workload mobility and flexible deployment:* Cloud computing involves on-demand elastic provisioning of resources for multi-tenant environments over the same physical infrastructure. A POD typically consists of one or more racks of servers with associated network and storage connectivity. Cross-pod expansion is a common use case wherein tenants may start off on a POD and due to expansion, require servers/VMs on other pods, especially in the case when tenants on the other pods are not fully utilizing all their resources. VXLAN provides for stretched Layer 2 domains connecting the individual servers/VMs to support seamless movement and flexible placement of virtualized resources across data center network.
- *Rapid provisioning:* To support agile business needs, provisioning of applications needs to be quick and easy. VXLAN as an overlay allows for the same by avoiding any configuration changes to the underlying physical infrastructure.

VXLAN Essentials

Below is a summary of some essential VXLAN components:

- **VXLAN segment:** Each VXLAN overlay network is called a VXLAN segment. Hosts (e.g. VMs) that are part of the same VXLAN segment (same broadcast domain) can communicate with each other. A Layer 3 gateway is needed for inter-VNI communication.
- **VXLAN Network Identifier (VNI):** Each VXLAN segment is identified by a 24-bit identifier called VNI. This supports high-scale network segmentation by allowing for up to 16M VXLAN segments to co-exist within the same administrative domain. A VNI provides traffic isolation for each VXLAN segment. A host is identified by the unique combination of its MAC address and VNI. This allows for overlapping addresses to exist across segments since the VNI defines the scope of the inner Layer 2 frame originated by the end host. As a result, different VNIs translate to different domains and traffic does not cross over between network segments.
- **VXLAN Tunnel Endpoint (VTEP):** The Layer 2 frame originated by the end-host is encapsulated in a VXLAN header that includes the VNI used to identify the VXLAN segment this end-host belongs to. Tunnels are further used to append outer header encapsulation comprising of UDP, IP, and Ethernet headers to transport the end-host traffic over the underlying physical network. These tunnel endpoints or Network Virtualization Edges (NVEs) that are responsible for mapping end-hosts to the appropriate VXLAN segments (local VLAN to VXLAN VNI mapping) and VXLAN related tunnel encapsulation and decapsulation are called VTEPs. VTEPs can exist on any network device (multi-vendor environment) or compute element (e.g. server hypervisor). Each VTEP is identified by a unique IP address (lo0 address). The physical underlay network is responsible for providing IP reachability between the VTEPs to ensure end-to-end traffic delivery. VTEPs perform two important functions namely: 1) Remote VTEP discovery to create a mapping of end-host MAC addresses and corresponding VTEPs for the VXLAN segments it is responsible for; and, 2) VXLAN encapsulation of end-host traffic to be transported transparently over the underlay network and further decapsulation of received packets for delivery to the right end-hosts. VTEPs can switch packets in the same VNID or same broadcast domain (L2 gateways) or can route packets across different VNIDs or different subnets (L3 gateways).
- **VXLAN encapsulation:** VXLAN encapsulation adds an additional 50-54B of overhead as a result of appending UDP, IP, and Ethernet headers to the frame generated by the end-host. As a result, it is important that the IP underlay network support Jumbo frames to account for this overhead, which means MTU values would need to reflect the same across the entire path traversed by the encapsulated packet. The UDP header added populates its source port value by computing a hash of the original inner frame contents (L2/L3/L4 headers). This enables flow entropy to ensure efficient load-balancing of VXLAN traffic over the ECMP paths offered by the physical underlay. UDP checksum is set to 0x0000. If the checksum is not set to 0x0000 by the source VTEP, then the receiving VTEP should verify the checksum. If incorrect, the frame must be dropped and not decapsulated. The outer IP header consists of source and destination VTEP IP addresses and these values are used for the packet to be routed by the underlay. The underlay network devices do not look into the VXLAN header, neither do they maintain any tunnel state and as such are decoupled from the overlay.

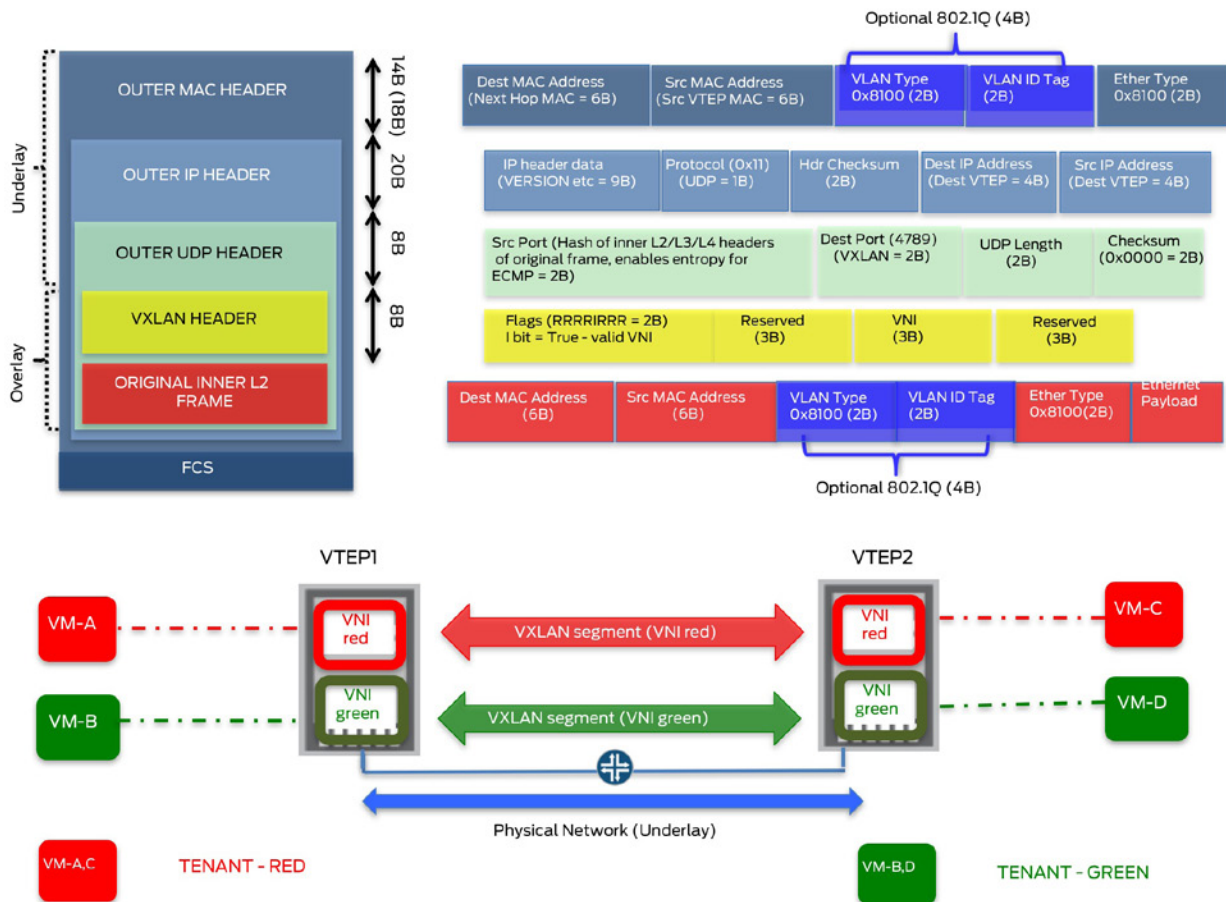


Figure 1.5 VXLAN Frame Format and Terminology

VXLAN Variants

A control plane protocol defines how speakers form relationships, exchange topology information, and make best path decisions. This in turn influences data plane behavior which determines how data packets are transported over a network path. The current VXLAN standard does not define a control plane. As a result, exchanging topology information or address learning is done in the data plane, similar to Layer 2 flood and learn. When a VTEP learns the MAC address of a local end-host (local learning), it creates an entry in its forwarding table. In the absence of a control plane protocol, it does not advertise this information to any of the remote VTEPs. Each VXLAN speaker needs to perform its own independent address learning. When a VTEP receives a VXLAN packet (remote learning), it creates an association tracking the remote MAC, VNI it was learned for and remote VTEP it was learned from, so that it knows how to forward traffic destined to this remote MAC in the future. For multi-destination traffic (Broadcast, Unknown Unicast and Multicast), it resorts to a flooding mechanism. VXLAN variants differ in their capability on how they enable MAC learning and handle BUM traffic.

- **Multicast/VXLAN:** Remote MAC learning is done in the data plane. Each VXLAN segment or VNID is mapped to a multicast group. VTEPs that have end-hosts in a given VNID will join the same multicast group (IGMP). Multicast groups are thus

used to limit the flooding scope of BUM traffic to only interested VTEPs. For unknown unicast or broadcast, the VTEP will flood the VXLAN packet by sending it only to the multicast group associated with the given VNID or VXLAN segment. Multicast needs to be enabled in the underlay to maintain distribution trees for multicast groups (PIM). With underlay replication, packets are replicated at the farthest point in the physical topology, making it efficient for heavy multicast traffic. However, this necessity for the underlay to run multicast can be a significant operational overhead, especially in environments that do not run multicast today and would have to do so only to support VXLAN. Though multicast provides some efficiency in utilizing resources, it does pose scalability challenges. While VXLAN can support 16M logical Layer 2 networks, it is not possible to support 1:1 mapping between that many VNIDs and multicast groups. If a given multicast group has to support more than one VNID, it can lead to unnecessary flooding wherein uninterested VTEPs will still have to do some processing to drop traffic from the unwanted VNIDs.

- **Unicast (Static) VXLAN:** Like Multicast/VXLAN, remote MAC learning is done in the data plane. Through the use of ingress replication, the source VTEP will replicate a BUM packet locally and send a unicast copy to each of the other VTEPs participating in the same VXLAN segment. Multiple unicast VTEPs can be configured for a given VNI. Static configuration, while simpler to manage, along with providing granular control over the flooding scope, inhibits auto-discovery of dynamic topology changes. Moreover, additional processing at the headend node might not make this an ideal choice for heavy multicast traffic.
- **OVSDB/VXLAN:** OVSDB provides control plane functionality for VXLAN, i.e., remote MAC learning is now done in the control plane. OVSDB is usually used in conjunction with an overlay controller. BUM traffic replication is handled by the controller, which uses OVSDB to enable MAC address learning and handles VTEP provisioning. More on OVSDB is available in the Appendix.
- **EVPN/VXLAN:** Like OVSDB, EVPN is also a control plane technology for VXLAN. It uses MP-BGP to enable remote MAC learning in the control plane and can be used with or without a controller. More details on the same are available in the *Understanding EVPN* section on the next page.

NOTE Refer to the Appendix for more information on a Multicast/VXLAN packet walkthrough and for more information on understanding OVSDB.

Hardware VTEP Gateways

Most data center environments consist of a combination of virtualized and non-virtualized compute resources. Non-virtualized or Bare-Metal Servers (BMSs) are not capable of any server or network virtualization (single tenant). They do not support VXLAN and use VLANs for network segmentation. These are generally used by high-performance applications that require dedicated resources. There is often a need to allow for communication between virtual and physical workloads. Some common examples include interaction for multi-tier applications (Web/Application tiers could be virtual workloads and Database tier could be residing on a BMS), Physical-to-Virtual (P-V) migration, VMs need to communicate with firewall or load-balancer appliances deployed on BMSs. Apart from connecting virtual workloads (as shown in Figure 4), VTEP gateways can also handle the VLAN to VXLAN bridging to connect physical and virtual workloads.

Based on where the Network Virtualization Endpoint (NVE) resides, VTEPs can be categorized as listed in Table 1.1.

Table 1.1 Characteristics of VTEPs

	Software VTEP Gateways	Hardware VTEP Gateways
Description	When VTEP functionality is enabled in software (VXLAN enabled hypervisor) on a server. These are often used in conjunction with a SDN controller (e.g. Juniper Contrail or VMware NSX).	When VTEP functionality is enabled on a hardware network device. These can be used with or without a SDN controller and in terms of VXLAN functionality are equivalent to Software VTEPs. Hardware VTEPs typically use a control plane protocol like OVSDB or EVPN to interact with the controller in use. (For more information on integration with VMware NSX please refer to: https://www.juniper.net/documentation/en_US/release-independent/solutions/information-products/pathway-pages/solutions/qfx5100-nsx-impl-guide.pdf .)
Scale and Performance	Though software gateways can be more cost effective, the maximum throughput that can be achieved is limited to the capacity of the physical server. When traffic requirements are not very high, this makes them a good choice for smaller deployments.	Hardware VTEP gateways, being network devices, offer high-density, high-performance and better scale. For larger deployments, if there are a large number of BMSs that need to communicate with the virtualized workloads, or if there is need to handle large traffic volumes, then a high throughput hardware gateway offers a better solution. Based on the desired deployment requirements, hardware VTEP gateways can be used to provide physical-physical (BMS-BMS), virtual-virtual (VM-VM) or virtual-physical (VM-BMS) connectivity.
Redundancy	With software VTEP gateways, the end-hosts/VMs are co-located with the NVE/VTEP in the same physical server and the links between them are virtual. Multihoming is typically not possible and if possible it is purely local to the physical server and is not visible to any other VTEP residing on a different server. As such, redundancy mechanisms are limited.	With the use of EVPN as the control plane, hardware VTEP gateways can support different modes of redundancy like All-Active or Single-Active multihoming.

Understanding EVPN

Because the design goal for Network Virtualization Overlay (NVO) is millions of instances per common physical infrastructure, the scaling properties of the control plane for NVO are extremely important. EVPN, and the extensions described herein, are designed with this level of scalability in mind:

- *Control plane learning and ARP suppression:* EVPN distinguishes between how MAC addresses are learned depending on whether a CE device is either directly connected to a PE device, or whether the CE device is behind another PE device. When a CE device is directly connected to a PE, traditional IEEE MAC learning techniques on the data plane can be used. This is called *local learning*. When a CE device is connected to a remote PE device the original PE device needs to learn the CE device's MAC address via the remote PE device. This is called *remote learning*. Unlike traditional bridging, MAC learning between PEs in EVPN occurs in the

control plane and not data plane. They are not learned through the data frames themselves, but rather in the control plane using MP-BGP. As broadcast domains increase, ARP traffic increases. This EVPN functionality results in a reduction of flooding for unknown unicast traffic. Most end-hosts send Gratuitous ARP (GARP) requests when they come online, which allows for local learning by EVPN PEs and further distributes the learned MAC and IP addresses to other PEs using the MP-BGP control plane for remote learning. MP-BGP NLRI (MAC/IP advertisement routes) are used to distribute MAC/IP addresses between PEs and as such reachability is advertised, prior to traffic to those destinations. EVPN also allows for the capability to suppress ARP flooding by activating the Proxy-ARP functionality. When an end host initiates an ARP request for another remote end-host, it is intercepted by the EVPN PE which performs a Proxy-ARP lookup on the requested IP address. If it finds a match it sends an ARP reply on behalf of the remote end host, without unnecessary flooding of ARP requests that consumes network bandwidth.

- *Active/Active (A/A) Multi-homing and Aliasing:* When a customer device or network is connected to two or more PEs via a set of Ethernet links (referred to as an Ethernet segment) and all multi-homed PEs can forward known unicast traffic to/from that Ethernet segment for a given VLAN, then such multi-homing is referred to as *All Active*. EVPN support for A/A multihoming provides for better site availability and load-balancing since all links are active and utilized. Multi-homed PEs are configured with the same LACP parameters for a given ESI, appearing to form a single system from the CE perspective. This allows the CE to send different flows to both PEs for the same CE-VID (VLAN-ID). When multiple EVPN PEs are connected to the same Ethernet segment (e.g. connected to the same server via a LAG), it is possible that only one PE learns MAC addresses from the end host (ARP traffic could be hashed to a single link in the LAG). As a result, remote PEs receive MAC/IP advertisement routes for these addresses from a single PE, even though multiple PEs are connected to the same Ethernet segment. Aliasing allows remote PEs to effectively load-balance traffic among all multi-homed PEs for a given Ethernet segment (same ESI). Ethernet A-D routes are used to convey to remote PEs that MAC addresses are reachable from all multi-homed PEs for a given Ethernet segment, even though reachability is not advertised by all of them.
- *Fast Convergence and Mass Withdraw:* EVPN defines a new mechanism to recover from PE-CE link failures, as well as multi-homed PE node failures, that quickly communicates to the remote PEs the need to update their forwarding tables. When an EVPN PE loses connectivity to a given Ethernet segment, it withdraws the corresponding Ethernet A-D route (A-D per ES). This leads to all remote peers immediately purging from the FIB, all MAC addresses learned from the failed peer on this ESI, instead of waiting for all MAC address routes to be withdrawn, thus enabling fast convergence. As a result, network reconvergence upon failure is independent of the number of MAC addresses learned by the PE. This is of importance in the context of virtualization applications that are fueling an increase to the volume of MAC addresses to be handled by the network.
- *MAC Mobility:* To support workload mobility and migrations, EVPN supports MAC Mobility (MAC move). It refers to the possibility of an end host (defined by MAC address) to move from one Ethernet segment to another. In a MAC move scenario, a given MAC address could appear to be reachable via multiple Ethernet segments, since there could be multiple MAC/IP advertisement routes. To help PEs track and process mobility events correctly, MAC/IP advertisement routes are tagged with the MAC mobility extended community attribute with a sequence number. EVPN also supports the capability of associating MAC addresses to a specific

Ethernet segment (sticky MAC i.e. static MAC addresses to not be subjected to MAC moves). MAC mobility for such sticky MAC addresses is disallowed and traffic for the same from any other Ethernet segment is discarded. This helps limit malicious traffic (spoofed addresses) into the network for that MAC address.

- *Ease of provisioning*: Ease of provisioning becomes paramount with the expansion of enterprise and data center deployments. EVPN presents a single VPN technology solution for both Layer 3 and Layer 2 VPNs to achieve deployment simplification. In addition to the automated discovery of PEs belonging to a given VPN instance, EVPN also supports auto discovery of PEs belonging to a given multihomed group as also automated Designated Forwarder (DF) election among multi-homed PEs to prevent looping of BUM traffic. Attributes needed for setting up the service, for example Route Distinguisher, Route Target, etc., can be automatically derived without the need for explicit configuration. Existing and well-understood architecture concepts like BGP route filtering and constrained route distribution to limit route distribution to interested PEs, and BGP route reflection/hierarchical route reflection to address scale considerations with full-mesh peering between PEs, can be applied to build EVPN solutions.
- *Distributed anycast gateway*: EVPN supports integrated routing and bridging, allowing for both intra-subnet (Layer 2) and inter-subnet (Layer 3) forwarding. Distributed anycast gateway functionality provides for optimal forwarding since end hosts can use their local gateway to send traffic outside of their IP subnet. IRB interfaces which act as Layer 3 gateways for a tenant (VNI) can be distributed across different PEs.

The default gateway extended community is needed in the MAC/IP advertisement routes of IRB interfaces when different MAC addresses are configured on IRB interfaces across PEs for a given VLAN and it becomes necessary to distinguish these as gateway addresses. This is referred to as *default gateway synchronization*, which communicates to a receiving PE that it must forward traffic on behalf of the advertising PE, therefore, packets with destination MAC address set to that of the original PE's IRB interface will still be accepted and processed by the new PE as if they have been destined to the new PE's IRB MAC address. The default-gateway extended community helps achieve automatic synchronization across PEs when IRB MAC addresses for a given VLAN are different. However, a drawback to this approach is observed during PE node failure situations when all routes from the failed PE, including its IRB interface MAC/IP route that had been advertised with the default gateway community, are withdrawn, breaking any IRB MAC equivalency. In this scenario, when traffic from the moved VM with destination MAC address is set to that of the original PE's IRB interface, it is not processed by the new PE. The VM now needs to re-ARP for the default gateway MAC address breaking traffic flows.

However, when the same MAC address is configured on all IRB interfaces across PEs for a given VLAN, then they have been manually synchronized. JUNOS platforms are capable of using the `virtual-gateway-address` knob which configures the same anycast IP/MAC address on all necessary gateways to achieve gateway redundancy. Seamless workload mobility can now be achieved since there is no longer a need for the end host to send an ARP request to re-learn the gateway address, as this is identically provisioned and synchronized. The IRB interface address now has two components – unicast address and anycast address. The IRB anycast address is the virtual-gateway address, which is the same IP/MAC across all PEs for a given subnet/vlan and is used to achieve the default gateway functionality and redundancy. The IRB unicast address has a unique IP/MAC, which is used to handle ping and ARP requests. Layer 3 EVPN PE advertises two EVPN MAC routes for the corresponding

IP addresses associated with the IRB interface.

Consider a CE connected to two PEs – PE1 and PE2 – with the same IRB IP address on both PEs. In this case, when a ping is initiated from PE2 to the CE, it might not receive an ICMP reply back because the CE could have sent the response to the other active PE, PE1 (with the same IRB IP), resulting in ping failures. As such, asymmetric data paths for the ARP request and response can be avoided when the IRB interface arp's for a host's destination MAC address. This is useful where L3 PEs have to interoperate with L2 PEs (for example, leaf devices in an IP fabric which cannot provide L3 gateway functionality). L2 EVPN PEs advertise only MAC addresses but not IP and MAC addresses (MAC+IP) Type 2 routes. In this case, for inter-subnet traffic, to forward packets to an end host the L3 EVPN PE initiates an ARP request with the IRB interface's own IP/MAC address. The host learns the IRB interface's IP/MAC binding and adds or refreshes the ARP entry based on the ARP request packet from the L3 gateway. The ARP response from the host can arrive on any of the connected multi-homed L2 PE devices, and it has to have a destination MAC (DMAC) of the IRB interface's own MAC address. So the ARP request is unicast back to the L3 gateway device that initiated the ARP request. This effectively means that L3 gateway devices rely on ARP/NDP to discover and install the MAC/IP bindings, which further more means each IRB must have its own interface MAC address to identify itself (for ARP) and virtual MAC address for gateway functions. Chapter 2 will delve into details on relevant configuration knobs and their usage in EVPN-VXLAN deployment scenarios.

- **Traffic Optimization:** EVPN provides for optimized delivery of inbound and outbound traffic. Host MAC-IP synchronization refers to the process of a PE that has learned the MAC/IP binding (ARP snooping), to transmit another MAC advertisement route that contains both the MAC and IP addresses for an end host. The receiving PE installs the MAC address into the EVI (MAC-VRF) and the IP address into the corresponding IP-VRF. L2 and L3 aware services can be stretched between VMs within and across data centers. This capability further extends support for VMTO (Virtual Machine Traffic Optimization), which prevents traffic tromboning on VM migration. A VM that has moved might not have flushed its ARP table because it continues to send inter-subnet packets to its previous default gateway, which may no longer be the optimal or even local default gateway after the move. As a result, outbound traffic from the VM to an external destination might trombone via the previously configured default gateway. This egress trombone effect can be eliminated using the distributed anycast gateway functionality (`virtual-gateway-address`), which ensures that every router within the VLAN has an active instance of the default gateway. For a given VLAN/subnet, the same default gateway (same IP/MAC address) is realized simultaneously on appropriate devices, thereby ensuring the default gateway for the end hosts is always at the closest point. Traffic tromboning could also be observed in the inbound direction when an external destination across the WAN is trying to reach an end-host that has moved from DC-1 to DC-2 (same IP subnet as Layer 2 is stretched). Initial reachability could have been established by means of DC-1 and DC-2 gateways advertising summary routes. An end host across the WAN could prefer the advertisement from DC-1 gateway for a given IP subnet. Since it does not have host routes or visibility into the destination host's exact location, to reach a host that has moved from DC-1 to DC-2 traffic from the remote end host could trombone via DC-1.

- *Support for different data planes:* EVPN decouples the control and data plane. This decoupling allows for a consistent signaled forwarding database to be created across the network using control plane learning instead of relying on flooding and learning. MP-BGP provides control plane functionality, which is the same over any data plane encapsulation.

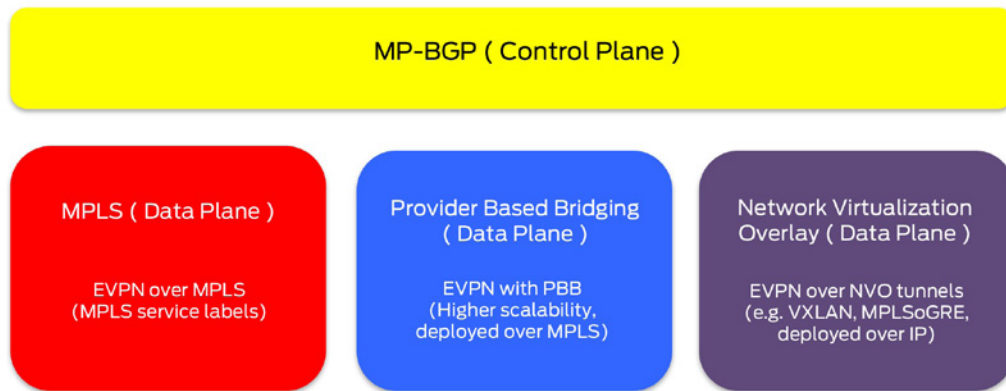


Figure 1.6 EVPN Data Plane Encapsulation Options

A brief summary of essential EVPN concepts is illustrated in Figure 1.6. Please refer to the Appendix for more information on EVPN service interfaces and EVPN Route Types.

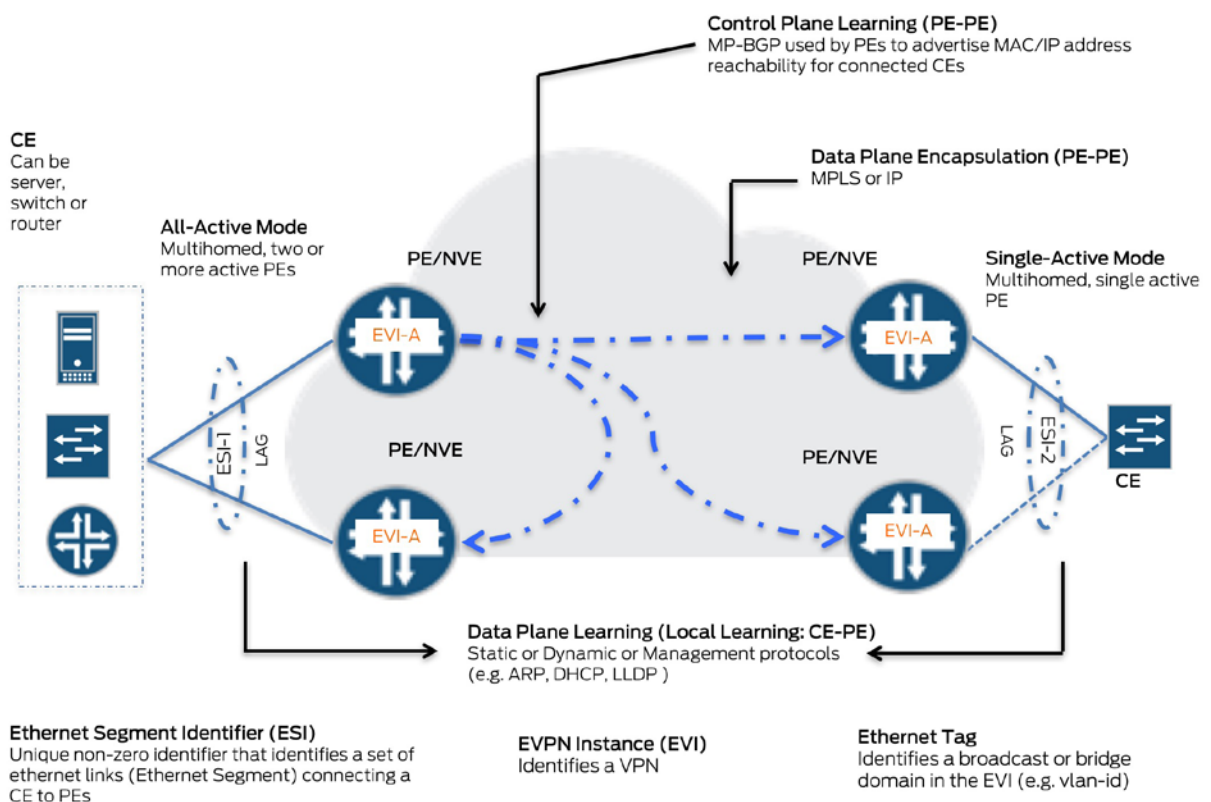


Figure 1.7 EVPN Conceptual Overview

EVPN-VXLAN Packet Walkthrough

This section provides an overview packet walkthrough and verification of baseline EVPN functions.

Testbed

QFX10002-36Q devices are used to build the Leaf and Spine layers. QFX5100-48S-6Q devices are used to build the VC (2 QFX5100s) and the Access Switch (1 QFX 5100). End-hosts have been simulated using IXIA tester ports.

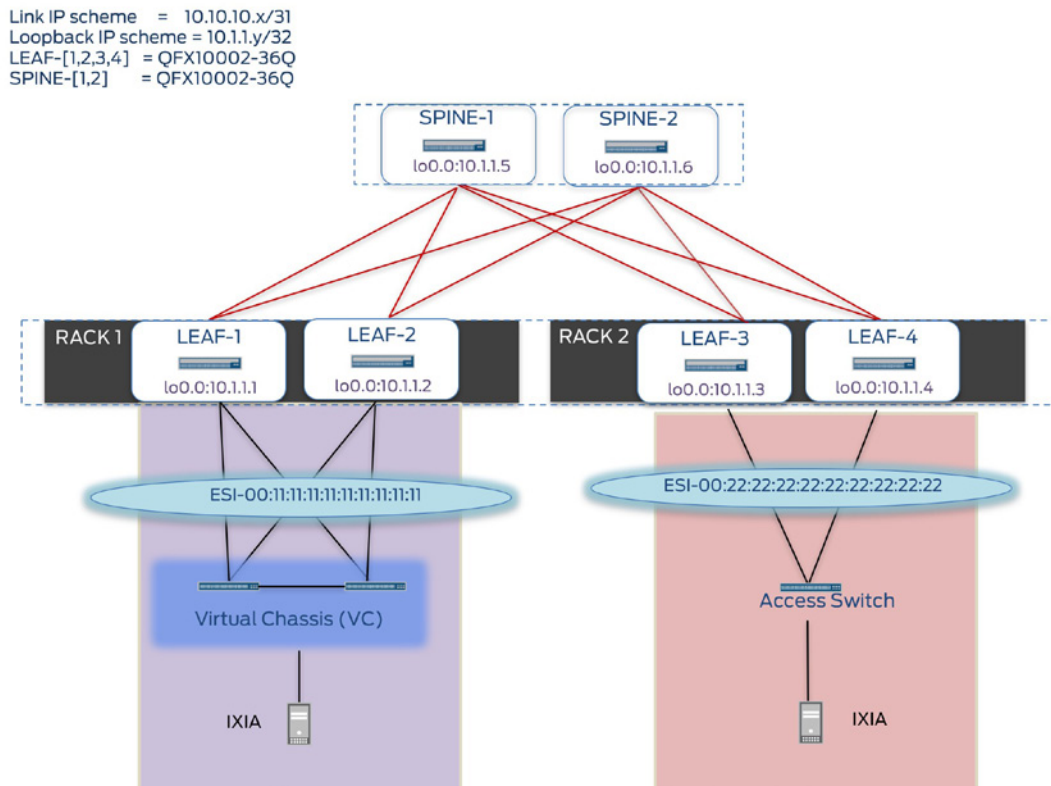


Figure 1.8 EVPN-VXLAN Physical Topology

Design Highlights

- Leaf nodes function as L2/L3 gateways (consolidated L2 and L3 gateway functionality in a single device).
- VTEPs are located on each leaf node. Full mesh of VXLAN tunnels exist between all leaf nodes.
- IP underlay is provided by a single ISIS Layer 2 domain.
- Spine nodes are EVPN-unaware: they only participate in the underlay (ISIS configuration only).
- VLAN-aware EVPN service.

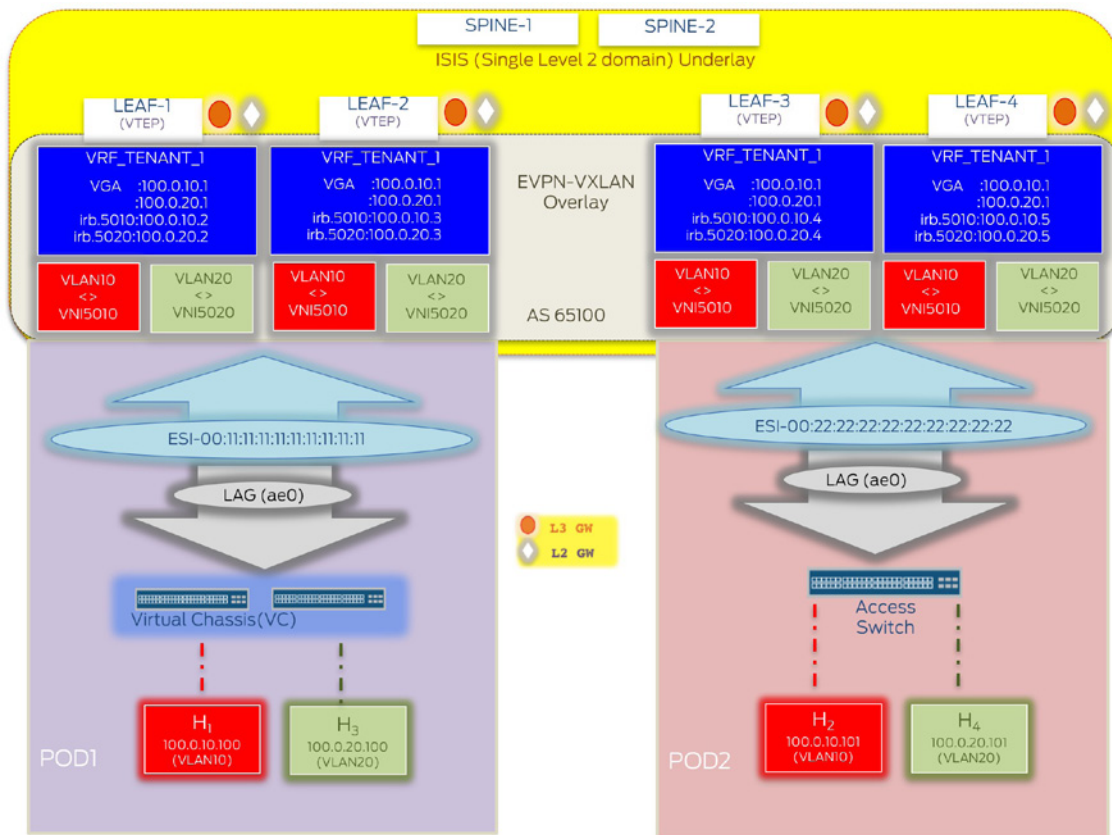


Figure 1.9 EVPN-VXLAN Logical Topology

- MTU has been set on all physical interfaces to account for VXLAN encapsulation.
- CEs (VC and TOR) are multi-homed to the redundant pair of leaf nodes (all-active mode).
- Intra-subnet communication can be achieved using Type 2 routes only. In this example, Type 2 routes have been used to set up inter-subnet communication as well.

Verification

This section summarizes the verification of baseline functional components and commands used for the same. For brevity, outputs are shown for Leaf-1 and have been truncated to highlight relevant information.

Underlay

ISIS provides IP loopback reachability for all EVPN (leaf and spine) nodes to establish VXLAN tunnels and BGP sessions.

```
jnpr@LEAF-1> show route table inet.0 | match "10.1.1.[1-6]/32"
10.1.1.1/32      * [Direct/0] 20:02:00
10.1.1.2/32      * [IS-IS/18] 19:56:37, metric 20
10.1.1.3/32      * [IS-IS/18] 19:56:37, metric 20
10.1.1.4/32      * [IS-IS/18] 19:56:37, metric 20
10.1.1.5/32      * [IS-IS/18] 20:01:13, metric 10
10.1.1.6/32      * [IS-IS/18] 19:56:37, metric 10
```

Overlay (Data Plane – VXLAN)

Verify state of all established VXLAN tunnels on VTEPs (leaf nodes).

For each remote VTEP, a vtep.x logical interface (IFL) is created dynamically. Each remote VTEP is reachable via a unicast tunnel represented by the vtep.x IFL. For example, in the following output for LEAF-1, vtep.32769 is the IFL created to represent the VXLAN tunnel from the source (10.1.1.1 – vtep.32768) to 10.1.1.4. The loopback on each VTEP, here lo0.0, has been used as the tunnel endpoint address.

```
jnpr@LEAF-1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
Interface index: 642, SNMP ifIndex: 511
Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed: Unlimited
Device flags: Present Running
Link type: Full-Duplex
Link flags: None
Last flapped: Never
Input packets: 0
Output packets: 0

Logical interface vtep.32768 (Index 551) (SNMP ifIndex 556)
Flags: Up SNMP-Traps Encapsulation: ENET2
VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.1.1.1,
L2 Routing Instance: default-switch, L3 Routing Instance: default
Input packets: 0
Output packets: 0

Logical interface vtep.32769 (Index 566) (SNMP ifIndex 538)
Flags: Up SNMP-Traps Encapsulation: ENET2
VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.4,
L2 Routing Instance: default-switch, L3 Routing Instance: default
Input packets: 79803896
Output packets: 80399681
Protocol eth-switch, MTU: Unlimited
Flags: Trunk-Mode

Logical interface vtep.32770 (Index 567) (SNMP ifIndex 531)
Flags: Up SNMP-Traps Encapsulation: ENET2
VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.3,
L2 Routing Instance: default-switch, L3 Routing Instance: default
Input packets: 85167340
Output packets: 85812491
Protocol eth-switch, MTU: Unlimited
Flags: Trunk-Mode

Logical interface vtep.32771 (Index 568) (SNMP ifIndex 528)
Flags: Up SNMP-Traps Encapsulation: ENET2
VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.2,
L2 Routing Instance: default-switch, L3 Routing Instance: default
Input packets: 3022
Output packets: 827
Protocol eth-switch, MTU: Unlimited
Flags: Trunk-Mode
```

Verify established VXLAN tunnels use ECMP over underlay paths.

A new RIB called ::vxlan.inet.0 holds the routes to remote VTEPs and uses inet.0 to for route resolution in accordance with Multi-Topology Routing (indicated by : in front of the RIB name).

```
jnpr@LEAF-1> show route table ::vxlan.inet.0
::vxlan.inet.0: 10 destinations, 10 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```



```

10.1.1.1/32      *[Direct/0] 1d 11:21:13
                  > via lo0.0
10.1.1.2/32      *[Static/1] 1d 11:19:04, metric2 20
                  to 10.10.10.8 via et-0/0/1.0
                  > to 10.10.10.18 via et-0/0/5.0
10.1.1.3/32      *[Static/1] 1d 11:19:37, metric2 20
                  to 10.10.10.8 via et-0/0/1.0
                  > to 10.10.10.18 via et-0/0/5.0
10.1.1.4/32      *[Static/1] 1d 11:19:54, metric2 20
                  to 10.10.10.8 via et-0/0/1.0
                  > to 10.10.10.18 via et-0/0/5.0

```

In the next output, from LEAF-1, VXLAN tunnels to remote VTEPs, 10.1.1.2, 10.1.1.3, and 10.1.1.4 are represented by VTEP IFLs vtep.32771, vtep.32770, and vtep.32769, respectively. Each remote VTEP is reachable via a composite next-hop that points to unicast next-hop composed of ECMP next-hops in the underlay indicated by the unicast entries following the unicast entry.

```
jnpr@LEAF-1> show route forwarding-table table default-switch extensive
```

```
Routing table: default-switch.evpn-vxlan [Index 3]
```

```
Destination: vtep.32769
```

```
Route type: interface
```

```
Route reference: 0
```

```
Route interface-index: 566
```

```
Multicast RPF nh index: 0
```

```
Flags: sent to PFE
```

```
Next-hop:
```

```
Next-hop type: composite
```

```
Index: 1754
```

```
Reference: 8
```

```
Next-hop type: indirect
```

```
Index: 524286
```

```
Reference: 3
```

```
Next-hop type: unicast
```

```
Index: 524297
```

```
Reference: 10
```

```
Next-hop: 10.10.10.8
```

```
Next-hop type: unicast
```

```
Index: 1753
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/1.0
```

```
Weight: 0x0
```

```
Next-hop: 10.10.10.18
```

```
Next-hop type: unicast
```

```
Index: 1775
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/5.0
```

```
Weight: 0x0
```

```
Destination: vtep.32770
```

```
Route type: interface
```

```
Route reference: 0
```

```
Route interface-index: 567
```

```
Multicast RPF nh index: 0
```

```
Flags: sent to PFE
```

```
Next-hop:
```

```
Next-hop type: composite
```

```
Index: 1759
```

```
Reference: 8
```

```
Next-hop type: indirect
```

```
Index: 524288
```

```
Reference: 3
```

```
Next-hop type: unicast
```

```
Index: 524297
```

```
Reference: 10
```

```
Next-hop: 10.10.10.8
```

```
Next-hop type: unicast
```

```
Index: 1753
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/1.0
```

```
Weight: 0x0
```

```
Next-hop: 10.10.10.18
```

```
Next-hop type: unicast
```

```
Index: 1775
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/5.0
```

```
Weight: 0x0
```

```
Destination: vtep.32771
```

```
Route type: interface
```

```
Route reference: 0
```

```
Route interface-index: 568
```

```
Multicast RPF nh index: 0
```

```
Flags: sent to PFE
```

```
Next-hop:
```

```
Next-hop type: composite
```

```
Index: 1766
```

```
Reference: 8
```

```
Next-hop type: indirect
```

```
Index: 524294
```

```
Reference: 3
```

```
Next-hop type: unicast
```

```
Index: 524297
```

```
Reference: 10
```

```
Next-hop: 10.10.10.8
```

```
Next-hop type: unicast
```

```
Index: 1753
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/1.0
```

```
Weight: 0x0
```

```
Next-hop: 10.10.10.18
```

```
Next-hop type: unicast
```

```
Index: 1775
```

```
Reference: 7
```

```
Next-hop interface: et-0/0/5.0
```

```
Weight: 0x0
```

Verify VTEP and VNI association

The output below shows the remote VTEP information learned by the source VTEP and the VXLAN bridge domains each of the remote VTEPs is serving. All VTEPs are interested in VNIs 5010 and 5020:

```
jnpr@LEAF-1> show ethernet-switching vxlan-tunnel-end-point source
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx
<default>                0   10.1.1.1      100.0  0
L2-RTT                   Bridge Domain  VNID    MC-Group-IP
default-switch            bd5010+10      5010    0.0.0.0
default-switch            bd5020+20      5020    0.0.0.0

jnpr@LEAF-1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx
<default>                0   10.1.1.1      100.0  0
RVTEP-IP                IFL-Idx  NH-Id
10.1.1.2                568      1766
  VNID                    MC-Group-IP
  5010                    0.0.0.0
  5020                    0.0.0.0
RVTEP-IP                IFL-Idx  NH-Id
10.1.1.3                567      1759
  VNID                    MC-Group-IP
  5020                    0.0.0.0
  5010                    0.0.0.0
RVTEP-IP                IFL-Idx  NH-Id
10.1.1.4                566      1754
  VNID                    MC-Group-IP
  5020                    0.0.0.0
  5010                    0.0.0.0
```

Overlay (Control Plane – EVPN)

IBGP sessions are established between all leaf nodes with only the “evpn” NLRI.

VTEP peer discovery with the use of EVPN is control-plane based. As outlined in the configuration section above, only the source vtep (here 10.0.0 address) has been configured. To exchange any reachability information, VTEPs first need to establish BGP peering for the overlay. Remote VTEP peers are recognized when a local VTEP receives EVPN NLRI from them. To ensure that communication is established only between authorized peers, BGP authentication can be enabled:

```
jnpr@LEAF-1> show bgp summary
Groups: 1 Peers: 3 Down peers: 0
Table Tot Paths Act Paths Suppressed History Damp State Pending
bgp.evpn.0 76 76 0 0 0 0 0
Peer AS InPkt OutPkt OutQ Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped
10.1.1.2 65100 4908 4910 0 0 1d 12:49:58 Estab1
  bgp.evpn.0: 26/26/26/0
  default-switch.evpn.0: 25/25/25/0
  __default_evpn__.evpn.0: 1/1/1/0
10.1.1.3 65100 4916 4911 0 0 1d 12:50:31 Estab1
  bgp.evpn.0: 26/26/26/0
  default-switch.evpn.0: 26/26/26/0
  __default_evpn__.evpn.0: 0/0/0/0
10.1.1.4 65100 4913 4914 0 0 1d 12:50:48 Estab1
  bgp.evpn.0: 24/24/24/0
  default-switch.evpn.0: 24/24/24/0
  __default_evpn__.evpn.0: 0/0/0/0

jnpr@LEAF-1> show bgp neighbor 10.1.1.2 | match NLRI
NLRI for restart configured on peer: evpn
NLRI advertised by peer: evpn
NLRI for this session: evpn
NLRI that restart is negotiated for: evpn
NLRI of received end-of-rib markers: evpn
NLRI of all end-of-rib markers sent: evpn
```



```
jnpr@LEAF-1> show bgp neighbor 10.1.1.3 | match NLRI
NLRI for restart configured on peer: evpn
NLRI advertised by peer: evpn
NLRI for this session: evpn
NLRI that restart is negotiated for: evpn
NLRI of received end-of-rib markers: evpn
NLRI of all end-of-rib markers sent: evpn
```

```
jnpr@LEAF-1> show bgp neighbor 10.1.1.4 | match NLRI
NLRI for restart configured on peer: evpn
NLRI advertised by peer: evpn
NLRI for this session: evpn
NLRI that restart is negotiated for: evpn
NLRI of received end-of-rib markers: evpn
NLRI of all end-of-rib markers sent: evpn
```

```
jnpr@LEAF-1> show bfd session
```

Address	State	Interface	Detect Time	Transmit Interval	Multiplier
10.1.1.2	Up		1.050	0.350	3
10.1.1.3	Up		1.050	0.350	3
10.1.1.4	Up		1.050	0.350	3

3 sessions, 3 clients
Cumulative transmit rate 8.6 pps, cumulative receive rate 8.6 pps

Verify exchange of Ethernet Segment (Type 4) routes used for ES Discovery to enable multi-homing, DF Election, and Split Horizon/Local Bias.

Type 4 route is accepted from a peer that is connected to the same Ethernet Segment (es-import-target community). The same ESI value is set on redundant leaf nodes for multihoming. The output below shows the Type 4 route from LEAF-2 accepted by LEAF-1 since they share the same ESI value and matching es-import-target community. LEAF-3 and LEAF-4 do not share the same ESI value as LEAF-1 and LEAF-2 and hence will discard these Type 4 routes, which allows only PEs connected to the same ES to discover each other. Valid Type 4 routes are imported into the global EVPN table (bgp.evpn.0) and the secondary table (__default_evpn__.evpn.0) that accepts routes matching on the es-import RT as can be seen by the action of the route instance policy. Although this case study shows only a pair of redundant PEs multi-homed to the access device, support is not limited to only two. Unlike MC-LAG, EVPN can provide a solution that horizontally scales across multiple PEs.

```
jnpr@LEAF-1> show route instance __default_evpn__ detail
__default_evpn__:
Router ID: 0.0.0.0
Type: non-forwarding evpn State: Active
Route-distinguisher: 10.1.1.1:0
Vrf-import: [ __vrf-import-__default_evpn__-internal__ ]
Vrf-import-target: [ es-import-target:11-11-11-11-11 ]
Fast-reroute-priority: low
Tables:
__default_evpn__.evpn.0: 5 routes (5 active, 0 holddown, 0 hidden)
```

```
jnpr@LEAF-1> show policy __vrf-import-__default_evpn__-internal__
Policy __vrf-import-__default_evpn__-internal__:
Term unnamed:
  from community __vrf-community-__default_evpn__-import-internal__
  [es-import-target:11-11-11-11-11 ]
  then accept
Term unnamed:
  then reject
```

```
jnpr@LEAF-1> show route community-name __vrf-community-__default_evpn__-import-internal__
bgp.evpn.0: 76 destinations, 76 routes (76 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
4:10.1.1.2:0::1111111111111111:10.1.1.2/304
    *[BGP/170] 1d 18:50:08, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
```

```

> to 10.10.10.8 via et-0/0/1.0
  to 10.10.10.18 via et-0/0/5.0

__default_evpn__.evpn.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

4:10.1.1.1:0::1111111111111111:10.1.1.1/304
    *[EVPN/170] 1d 18:50:13
    Indirect
4:10.1.1.2:0::1111111111111111:10.1.1.2/304
    *[BGP/170] 1d 18:50:08, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    > to 10.10.10.8 via et-0/0/1.0
      to 10.10.10.18 via et-0/0/5.0

jnpr@LEAF-1> show route table bgp.evpn.0 extensive | find ^4:
4:10.1.1.2:0::1111111111111111:10.1.1.2/304 (1 entry, 0 announced)
    *BGP Preference: 170/-101
    Route Distinguisher: 10.1.1.2:0
    Next hop type: Indirect, Next hop index: 0
    Address: 0xa5bf570
    Next-hop reference count: 52
    Source: 10.1.1.2
    Protocol next hop: 10.1.1.2
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    State: <Active Int Ext>
    Local AS: 65100 Peer AS: 65100
    Age: 1d 17:57:49 Metric2: 20
    Validation State: unverified
    Task: BGP_65100_65100.10.1.1.2
    AS path: I
    Communities: encapsulation0:0:0:0:vxlan es-import-target:11-11-11-11-11-11
    Import Accepted
    Localpref: 100
    Router ID: 10.1.1.2
    Secondary Tables: __default_evpn__.evpn.0
    <snip>

```

For EVPN using VXLAN encapsulation, split horizon/local bias is implemented by not forwarding BUM traffic towards the access device on a shared ES, if a Type 4 route for the ESI identifying that ES is received from VTEP source IP of BUM traffic. For each ES, DF is elected, which is responsible for forwarding BUM traffic towards the multi-homed access.

```

jnpr@LEAF-1> show evpn instance designated-forwarder esi 00:11:11:11:11:11:11:11:11
Instance: default-switch
  Number of ethernet segments: 10
  ESI: 00:11:11:11:11:11:11:11:11
  Designated forwarder: 10.1.1.1

jnpr@LEAF-1> show evpn instance backup-forwarder esi 00:11:11:11:11:11:11:11:11
Instance: default-switch
  Number of ethernet segments: 10
  ESI: 00:11:11:11:11:11:11:11:11
  Backup forwarder: 10.1.1.2

jnpr@LEAF-1> show evpn instance extensive
Instance: default-switch
  Route Distinguisher: 10.1.1.1:100
  Encapsulation type: VXLAN
  <snip>
  Number of local interfaces: 1 (1 up)
    Interface name ESI Mode Status
    ae0.0 00:11:11:11:11:11:11:11:11 all-active Up
  <snip>
  Number of ethernet segments: 10
  ESI: 00:11:11:11:11:11:11:11:11
  Status: Resolved by IFL ae0.0
  Local interface: ae0.0, Status: Up/Forwarding
  Number of remote PEs connected: 1

```

```

      Remote PE      MAC label Aliasing label Mode
      10.1.1.2      5010      0              all-active
Designated forwarder: 10.1.1.1
Backup forwarder: 10.1.1.2
ESI: 00:22:22:22:22:22:22:22:22
Status: Resolved
Number of remote PEs connected: 2
      Remote PE      MAC label Aliasing label Mode
      10.1.1.3      5010      0              all-active
      10.1.1.4      5020      0              all-active
Router-ID: 10.1.1.1
Source VTEP interface IP: 10.1.1.1

```

Now let's verify exchange of Inclusive Multicast (Type 3) routes from each peer.

Used for BUM traffic replication to remote peers on a per VNI basis, ingress replication (overlay) uses unicast VXLAN tunnels for packet replication to all remote peers. Valid Type 3 routes are imported into the global EVPN table (bgp.evpn.0) and the secondary table (default-switch.evpn.0) that accepts routes matching on the configured policy:

```

jnpr@LEAF-1> show route table bgp.evpn.0 | match ^3:
3:10.1.1.2:100::5010::10.1.1.2/304
3:10.1.1.2:100::5020::10.1.1.2/304
3:10.1.1.3:100::5010::10.1.1.3/304
3:10.1.1.3:100::5020::10.1.1.3/304
3:10.1.1.4:100::5010::10.1.1.4/304

jnpr@LEAF-1> show route table bgp.evpn.0 extensive
3:10.1.1.4:100::5010::10.1.1.4/304 (1 entry, 0 announced)
  *BGP      Preference: 170/-101
            Route Distinguisher: 10.1.1.4:100
            PMSI: Flags 0x0: Label 0x1392: Type INGRESS-REPLICATION 10.1.1.4
            Next hop type: Indirect, Next hop index: 0
            Address: 0xa5bdf50
            Next-hop reference count: 14
            Source: 10.1.1.4
            Protocol next hop: 10.1.1.4
            Indirect next hop: 0x2 no-forward INH Session ID: 0x0
            State: <Active Int Ext>
            Local AS: 65100 Peer AS: 65100
            Age: 17:04      Metric2: 20
            Validation State: unverified
            Task: BGP_65100_65100.10.1.1.4
            AS path: I
            Communities: target:1:5010 encapsulation0:0:0:0:vxlan
            Import Accepted
            Localpref: 100
            Router ID: 10.1.1.4
            Secondary Tables: default-switch.evpn.0

```

A flood list is created for each VLAN. VNI RTs carried by Type 3 routes confine BUM traffic to only interested VTEPs. In the output above VTEP-4 (10.1.1.4/vtep.32770), is advertising Type 3 for only VNI 5010 (relevant configuration for VNI 5020 has been deactivated on LEAF-4). As shown below, the local ae interface and relevant VTEPs are part of the flood domain for each VLAN. Here, all VTEPs are interested in VNI 5010 (VLAN 10) and VNI 5020 (VLAN 20), except for LEAF-4, which is only interested in VNI 5010 (VLAN 10):

```

jnpr@LEAF-1> show interfaces extensive vtep | match "vxlan endpoint|logical interface"
Logical interface vtep.32768 (Index 551) (SNMP ifIndex 556) (HW Token 4294967295) (Generation 136)
  VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.1.1.1, L2 Routing Instance: default-
switch, L3 Routing Instance: default
Logical interface vtep.32769 (Index 560) (SNMP ifIndex 538) (HW Token 4294967295) (Generation 157)
  VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.3, L2 Routing Instance: default-
switch, L3 Routing Instance: default
Logical interface vtep.32770 (Index 561) (SNMP ifIndex 531) (HW Token 4294967295) (Generation 158)
  VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.4, L2 Routing Instance: default-
switch, L3 Routing Instance: default

```

Logical interface **vtep.32771** (Index 566) (SNMP ifIndex 528) (HW Token 4294967295) (Generation 159)
 VXLAN Endpoint Type: Remote, **VXLAN Endpoint Address: 10.1.1.2**, L2 Routing Instance: default-switch, L3 Routing Instance: default

jnpr@LEAF-1> **show ethernet-switching flood vlan-name bd5010 extensive**

Name: default-switch

CEs: 1

VEs: 3

VLAN Name: bd5010

<snip>

Flood route prefix: 0x30003/51

Flood route type: FLOOD_GRP_COMP_NH

Flood route owner: __all_ces__

Flood group name: __all_ces__

Flood group index: 1

Nexthop type: comp

Nexthop index: 1758

Flooding to:

Name	Type	NhType	Index
__ves__	Group	comp	1756

Composition: flood-to-all

Flooding to: <<< Flood list includes all remote VTEPs for VNI 5010

Name	Type	NhType	Index
vtep.32769	CORE_FACING	venh	1754
vtep.32770	CORE_FACING	venh	1759
vtep.32771	CORE_FACING	venh	1766

Flooding to:

Name	Type	NhType	Index
__all_ces__	Group	comp	1771

Composition: split-horizon

Flooding to:

Name	Type	NhType	Index
ae0.0	CE	ucst	1676

jnpr@LEAF-1> **show ethernet-switching flood vlan-name bd5020 extensive**

Name: default-switch

CEs: 1

VEs: 3

VLAN Name: bd5020

<snip>

Flood route prefix: 0x30004/51

Flood route type: FLOOD_GRP_COMP_NH

Flood route owner: __all_ces__

Flood group name: __all_ces__

Flood group index: 1

Nexthop type: comp

Nexthop index: 1764

Flooding to:

Name	Type	NhType	Index
__all_ces__	Group	comp	1773

Composition: split-horizon

Flooding to:

Name	Type	NhType	Index
ae0.0	CE	ucst	1676

Flooding to:

Name	Type	NhType	Index
__ves__	Group	comp	1767

Composition: flood-to-all

Flooding to:

Name	Type	NhType	Index
vtep.32769	CORE_FACING	venh	1754
vtep.32771	CORE_FACING	venh	1761

<<< Flood list includes all remote VTEPs for VNI 5020 (except VTEP-4 i.e. vtep.32770)

Verify exchange of Ethernet Auto-

Discovery (Type 1) routes (Per ES and Per EVI) used for Aliasing and MAC Mass Withdraw

jnpr@LEAF-1> **show route table bgp.evpn.0 | match ^1:**

```

1:10.1.1.2:0::1111111111111111::FFFF:FFFF/304
1:10.1.1.2:0::050000fe4c0000139200::FFFF:FFFF/304
1:10.1.1.2:0::050000fe4c0000139c00::FFFF:FFFF/304
1:10.1.1.2:100::1111111111111111::0/304
1:10.1.1.3:0::2222222222222222::FFFF:FFFF/304
1:10.1.1.3:0::050000fe4c0000139200::FFFF:FFFF/304
1:10.1.1.3:0::050000fe4c0000139c00::FFFF:FFFF/304
1:10.1.1.3:100::2222222222222222::0/304
1:10.1.1.4:0::2222222222222222::FFFF:FFFF/304
1:10.1.1.4:0::050000fe4c0000139200::FFFF:FFFF/304
1:10.1.1.4:0::050000fe4c0000139c00::FFFF:FFFF/304
1:10.1.1.4:100::2222222222222222::0/304

jnpr@LEAF-1> show route table bgp.evpn.0 extensive
<<< AD per ES 1:10.1.1.2:0::1111111111111111::FFFF:FFFF/304 (1 entry, 0 announced)
*BGP Preference: 170/-101
  Route Distinguisher: 10.1.1.2:0
  Next hop type: Indirect, Next hop index: 0
  Address: 0xa5beeb0
  Next-hop reference count: 22
  Source: 10.1.1.2
  Protocol next hop: 10.1.1.2
  Indirect next hop: 0x2 no-forward INH Session ID: 0x0
  State: <Active Int Ext>
Local AS: 65100 Peer AS: 65100
Age: 55:12 Metric2: 20
Validation State: unverified
Task: BGP_65100_65100.10.1.1.2
AS path: I
Communities: target:1:100 encapsulation0:0:0:0:vxlan
esi-label:all-active (label 0)
Import Accepted
Route Label: 1
Localpref: 100
Router ID: 10.1.1.2
Secondary Tables: default-switch.evpn.0
<<< AD per EVI 1:10.1.1.2:100::1111111111111111::0/304 (1 entry, 0 announced)
*BGP Preference: 170/-101
  Route Distinguisher: 10.1.1.2:100
  Next hop type: Indirect, Next hop index: 0
  Address: 0xa5beeb0
  Next-hop reference count: 22
  Source: 10.1.1.2
  Protocol next hop: 10.1.1.2
  Indirect next hop: 0x2 no-forward INH Session ID: 0x0
  State: <Active Int Ext>
Local AS: 65100 Peer AS: 65100
Age: 55:12 Metric2: 20
Validation State: unverified
Task: BGP_65100_65100.10.1.1.2
AS path: I
Communities: target:1:100 encapsulation0:0:0:0:vxlan
Import Accepted
Localpref: 100
Router ID: 10.1.1.2
Secondary Tables: default-switch.evpn.0
<snip>

```

Aliasing allows a peer to signal reachability to an ES (AD per EVI), even when it has not learned any MAC addresses from that segment, thereby allowing remote peers to load-balance traffic to all peers connected to an ES. Test case: LEAF-1 and LEAF-2 are multihomed to the same ES. LEAF-2 advertises Type 2 route for Host MAC/IP (H1 = 00:00:1e:63:c8:7c/100.0.10.100 – VNI 5010) to LEAF-1, LEAF-3 and LEAF-4. Even though, LEAF-3 and LEAF-4 do not learn the MAC address for H1 from LEAF-1, traffic for this host on VNI 5010 is load balanced to both LEAF-1 and LEAF-2.

```

jnpr@LEAF-2> show evpn database extensive mac-address 00:00:1e:63:c8:7c

```

```

Instance: default-switch
VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
  Source: 00:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
  Local origin: ae0.0
  Timestamp: Jan 16 00:11:47 (0x587c8043)
  State: <Local-MAC-Only Local-To-Remote-Adv-Allowed>
  IP address: 100.0.10.100
  Local origin: ae0.0
  L3 route: 100.0.10.100/32, L3 context: VRF_TENANT_1 (irb.5010)

jnpr@LEAF-1> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch
VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
  Source: 00:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
  Remote origin: 10.1.1.2
  Timestamp: Jan 16 00:11:49 (0x587c8045)
  State: <Remote-To-Local-Adv-Done>
  IP address: 100.0.10.100
  Remote origin: 10.1.1.2

jnpr@LEAF-3> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch
VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
  Source: 00:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
  Remote origin: 10.1.1.2
  Timestamp: Jan 16 00:11:49 (0x587c8045)
  State: <Remote-To-Local-Adv-Done>
  IP address: 100.0.10.100
  Remote origin: 10.1.1.2

jnpr@LEAF-4> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch
VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
  Source: 00:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
  Remote origin: 10.1.1.2
  Timestamp: Jan 16 00:11:49 (0x587c8045)
  State: <Remote-To-Local-Adv-Done>
  IP address: 100.0.10.100
  Remote origin: 10.1.1.2

jnpr@LEAF-1> show route table bgp.evpn.0 evpn-mac-address 00:00:1e:63:c8:7c
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c/304
  *[BGP/170] 00:00:23, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    > to 10.10.10.8 via et-0/0/1.0
    to 10.10.10.18 via et-0/0/5.0
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c::100.0.10.100/304
  *[BGP/170] 00:00:23, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    to 10.10.10.8 via et-0/0/1.0
    > to 10.10.10.18 via et-0/0/5.0

jnpr@LEAF-3> show route table bgp.evpn.0 evpn-mac-address 00:00:1e:63:c8:7c
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c/304
  *[BGP/170] 00:00:30, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    > to 10.10.10.12 via et-0/0/1.0
    to 10.10.10.22 via et-0/0/5.0
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c::100.0.10.100/304
  *[BGP/170] 00:00:30, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    to 10.10.10.12 via et-0/0/1.0

jnpr@LEAF-4> show route table bgp.evpn.0 evpn-mac-address 00:00:1e:63:c8:7c
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c/304
  *[BGP/170] 00:00:33, localpref 100, from 10.1.1.2
    AS path: I, validation-state: unverified
    > to 10.10.10.14 via et-0/0/1.0
    to 10.10.10.24 via et-0/0/5.0
2:10.1.1.2:100::5010::00:00:1e:63:c8:7c::100.0.10.100/304

```

H1 MAC points to ESI next-hop that resolves over VTEPs on LEAF-1 and LEAF-2 (LEAF-3 output below, same observed on LEAF-4):

Traffic from LEAF-3 and LEAF-4 (H2 > H1 1000 flows @ 1000 pps) is being load-balanced to both LEAF-1 (500 pps) and LEAF-2 (500 pps):

IXIA Statistics (Traffic from LEAF-3 and LEAF-4 load balanced to LEAF-1 and LEAF-2 @1000pps):

With MAC Mass Withdraw, when a peer loses connectivity to an ES it withdraws its Type 1 route (AD per ES), which leads to all remote peers immediately purging from the FIB all MAC addresses learned from the failed peer on this ESI, instead of waiting for all MAC address routes to be withdrawn, thus enabling fast convergence. A test case would be: LEAF-2 is advertising 60 host MAC addresses to LEAF-1, LEAF-3, and LEAF-4 (output for only LEAF-3 shown below). LEAF-3 receives Type 2 host routes and Type 1 routes from LEAF-2. LEAF-2 then loses connectivity to its ES. LEAF-2 withdraws its Type 1 route and LEAF-3 purges from the forwarding-table, all installed host MAC routes from LEAF-2.

```
jnpr@LEAF-3> show evpn database extensive
Jan 16 21:09:35
```

Instance: default-switch

VN Identifier: 5010, MAC address: 00:00:d4:37:61:aa
 Source: 00:11:11:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
 Remote origin: 10.1.1.2
 Timestamp: Jan 16 21:02:25 (0x587da561)
 State: <Remote-To-Local-Adv-Done>
 IP address: 100.0.10.31
 Remote origin: 10.1.1.2

<...>

VN Identifier: 5010, MAC address: 00:00:d4:37:62:20
 Source: 00:11:11:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
 Remote origin: 10.1.1.2
 Timestamp: Jan 16 21:04:17 (0x587da5d1)
 State: <Remote-To-Local-Adv-Done>
 IP address: 100.0.10.90
 Remote origin: 10.1.1.2

```
jnpr@LEAF-3> show route forwarding-table table default-switch | match 00:00:d4:37
Jan 16 21:11:19
00:00:d4:37:61:aa/48 user      0                indr    524291    62
<...>
00:00:d4:37:62:00/48 user      0                indr    524291    62
```

```
jnpr@LEAF-3> show log evpn-trace.log
Jan 16 21:19:19.085333 evpn_instance esi_add_macdb_esi_to_list:1927 EVPN instance ESI default-switch:00:11:11:11:11:11:11:11:11 [Intfs: 0, PEs: 2, MACs: 60, Refcount: 60, State: Resolved-By-Remote-PE] MAC 00:00:d4:37:61:d8 added to list, current number of MACs now 60
```

```
jnpr@LEAF-3> show route table bgp.evpn.0 | match ^1:10.1.1.2 | match 111
1:10.1.1.2:0::1111111111111111:FFFF/304
1:10.1.1.2:100::1111111111111111:0/304
```

[edit]

```
jnpr@LEAF-2# set interfaces ae0 disable
Jan 16 23:27:54
```

[edit]

```
jnpr@LEAF-2# commit
Jan 16 23:31:05
```

```
jnpr@LEAF-3> show log evpn-trace
Jan 16 23:31:06.143603 evpn_process_ad_rt_per_evi:2769 EVPN route (received) [Instance: default-switch, Type: ethernet AD per EVI (1), ESI: 1111111111111111, L2domain: 0, Label: 0, PE: 10.1.1.2] Deleting
Jan 16 23:31:06.143629 evpn_esi_aliasing_msg_send_to_l2ald:460 EVPN instance default-switch [VS, Refcount: 5, Intfs: 1 (1 up), IRBs: 2 (2 up), Peers: 3, Flags: 0x14802] Sent ESI Aliasing Delete with RTT 3, remote addr 10.1.1.2 ESI 1111111111111111 to L2ALD
Jan 16 23:31:06.143691 evpn_rt_unlock:1546 EVPN route (received) [Instance: default-switch, Type: ethernet AD per EVI (1), ESI: 1111111111111111, L2domain: 0, Label: 0, PE: 10.1.1.2] Decrement refcount to 0
Jan 16 23:31:06.248454 evpn_process_ad_rt_per_evi:2428 EVPN instance default-switch [VS, Refcount: 5, Intfs: 1 (1 up), IRBs: 2 (2 up), Peers: 3, Flags: 0x14802] Processing DELETE for AD route From MES 10.1.1.2 Peer 10.1.1.2 esi 1111111111111111, L2 domain 0xffffffff, label 0
Jan 16 23:31:07.140344 evpn_mac_ip_msg_send_to_l2ald:884 EVPN MAC default-switch::5010::00:00:d4:37:62:20 [Flags: 0x0] Sent MAC+IP withdraw, interface <none>, RTT 3, IP 100.0.10.90 remote NH 10.1.1.2, ESI 1111111111111111, VLAN 0, VNI 5010, flags 0x80, timestamp 0x587dc6f9 to L2ALD
jnpr@LEAF-3> show route table bgp.evpn.0 | match ^1:10.1.1.2 | match 111
jnpr@LEAF-3>
jnpr@LEAF-3> show route forwarding-table table default-switch | match 00:00:d4:37
jnpr@LEAF-3>
```

Now let's verify exchange of routes for distributed default gateway functionality.

Distributed default gateway functionality is provided here by means of the anycast address through `virtual-gateway-address` configuration. IRB interfaces, which act as L3 gateways for a VNI, are distributed across all leaf nodes. The next output for LEAF-1 shows that for VNI 5010 the IRB anycast address is assigned a VRRP MAC that is locally unique within the context of a virtual network. This VRRP MAC per IP subnet advertises connectivity to a shared

system generated ESI. Each L3 gateway will advertise Type 1 and Type 2 routes for this VRRP MAC (currently the system generated shared ESI assigned to VRRP MAC maps to the global AS (if AS configured under global routing-options hierarchy) and VNI – seen next $(65100)_{10} = (fe4c)_{16}$, $(5010)_{10} = (1392)_{16}$:

```
jnpr@LEAF-1> show route advertising-protocol bgp 10.1.1.3 extensive
* 1:10.1.1.1:0::050000fe4c0000139200::FFFF:FFFF/304 (1 entry, 1 announced)
  BGP group IBGP-EVPN type Internal
  Route Distinguisher: 10.1.1.1:0
  Route Label: 1
  Nexthop: Self
  Flags: Nexthop Change
  Localpref: 100
  AS path: [65100] I
  Communities: target:1:100 encapsulation0:0:0:0:vxlan
  esi-label:all-active (label 0)
* 2:10.1.1.1:100::5010::00:00:5e:00:01:01/304 (1 entry, 1 announced)
  BGP group IBGP-EVPN type Internal
  Route Distinguisher: 10.1.1.1:100
  Route Label: 5010
  ESI: 05:00:00:fe:4c:00:00:13:92:00
  Nexthop: Self
  Flags: Nexthop Change
  Localpref: 100
  AS path: [65100] I
  Communities: target:1:5010 encapsulation0:0:0:0:vxlan
* 2:10.1.1.1:100::5010::00:00:5e:00:01:01::100.0.10.1/304 (1 entry, 1 announced)
  BGP group IBGP-EVPN type Internal
  Route Distinguisher: 10.1.1.1:100
  Route Label: 5010
  ESI: 05:00:00:fe:4c:00:00:13:92:00
  Nexthop: Self
  Flags: Nexthop Change
  Localpref: 100
  AS path: [65100] I
  Communities: target:1:5010 encapsulation0:0:0:0:vxlan
```

```
jnpr@LEAF-1> show evpn instance extensive
ESI: 05:00:00:fe:4c:00:00:13:92:00
  Status: Resolved by IFL irb.5010
  Local interface: irb.5010, Status: Up/Forwarding
  Number of remote PEs connected: 3
    Remote PE      MAC label  Aliasing label  Mode
    10.1.1.2       5010       0                all-active
    10.1.1.4       5010       0                all-active
    10.1.1.3       5010       0                all-active
jnpr@LEAF-1> show ethernet-switching table
Vlan name  MAC address      MAC flags  Logical interface  Active source
bd5010     00:00:5e:00:01:01 DR,SD      esi.1753         05:00:00:fe:4c:00:00:13:92:00
jnpr@LEAF-1> show ethernet-switching vxlan-tunnel-end-point esi
ESI          RTT          VLNBH INH      ESI-IFL LOC-IFL  #RVTEPs
05:00:00:fe:4c:00:00:13:92:00 default-switch 1753 524289 esi.1753        3
  RVTEP-IP    RVTEP-IFL      VENH      MASK-ID  FLAGS
  10.1.1.3    vtep.32770     1751      1        2
  10.1.1.4    vtep.32771     1756      2        2
  10.1.1.2    vtep.32769     1750      0        2
```

Traffic Flows

For this case study, all devices are situated in the same data center (Intra DC) and all hosts belong to the same tenant (Intra VRF). Hosts H1 and H2 belong to VLAN 10, mapped to VNI 5010, while hosts H3 and H4 belong to VLAN 20, mapped to VNI 5020. All hosts share the same IP-VRF (VRF_TENANT_1).

```
Host 1 (H1 )      : IP1 = 100.0.10.100, MAC1 = 00:00:1e:63:c8:7c is multihomed to Leaf-1 and Leaf-2
                   (ESI-1 = 00:11:11:11:11:11:11:11:11)
Host 2 (H2 )      : IP2 = 100.0.10.101, MAC2 = 00:00:1e:63:d1:fd is multihomed to Leaf-3 and Leaf-4
```

(ESI-2 = 00:22:22:22:22:22:22:22:22:22:22)

Host 3 (H3) : IP3 = 100.0.20.100, MAC3 = 00:00:00:93:3c:f3 is multihomed to Leaf-1 and Leaf-2
(ESI-1 = 00:11:11:11:11:11:11:11:11:11)

Host 4 (H4) : IP4 = 100.0.20.101, MAC4 = 00:00:00:93:3c:f4 is multihomed to Leaf-3 and Leaf-4
(ESI-2 = 00:22:22:22:22:22:22:22:22:22:22)

Default gateway : 100.0.10.1 (virtual-gateway-address for irb.5010 L3 interface for VLAN 10)
: 100.0.20.1 (virtual-gateway-address for irb.5020 L3 interface for VLAN 20)

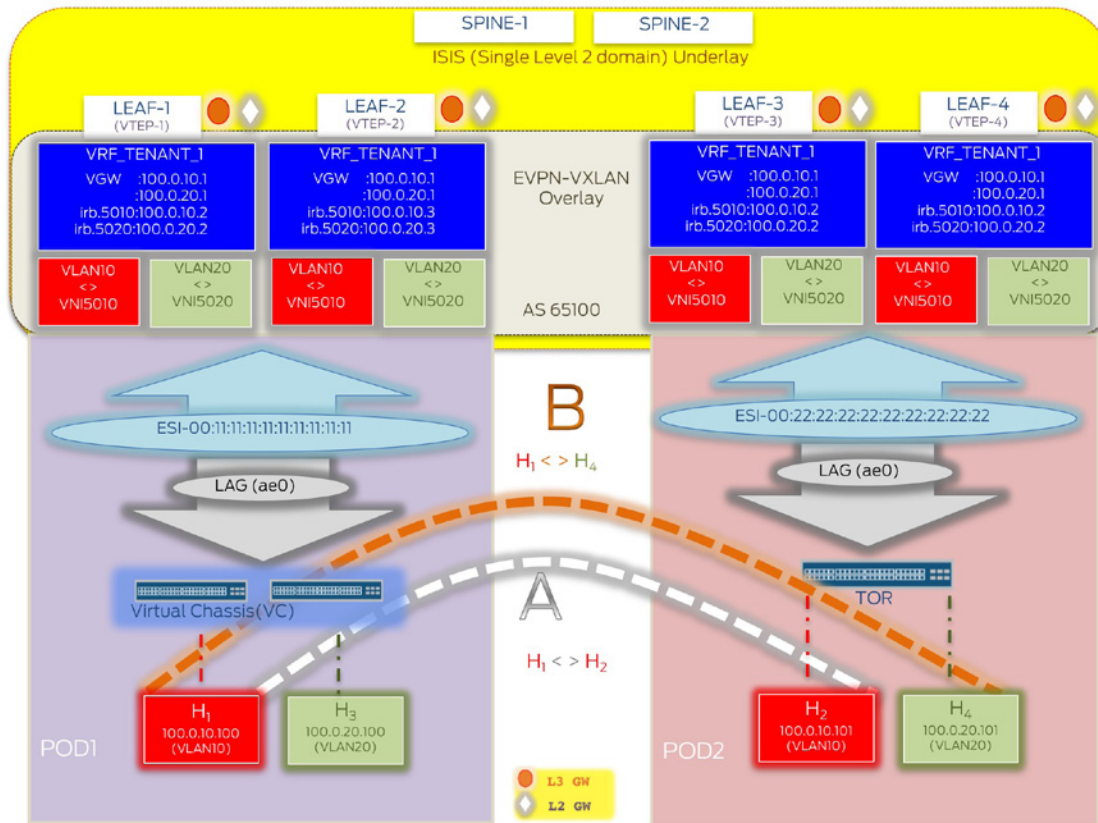


Figure 1.10 Traffic Flows

Two traffic flows between hosts in different PODs have been elaborated upon in this section. Intra – subnet, or between hosts in the same VLAN:

100.0.10.100 (H1 – VLAN 10/VNI 5010) <-> 100.0.10.101 (H2 – VLAN 10/VNI 5010)

And, Inter – subnet, or between hosts in different VLANs:

100.0.10.100 (H1 – VLAN 10/VNI 5010) <-> 100.0.20.101 (H4 – VLAN 20/VNI 5020)

MAC learning (Control Plane)

Once relevant EVPN configuration is done, even before any end-hosts are learned locally, Type 1, 3, and 4 routes are exchanged between the EVPN PEs.

1. H1 (IP1 = 100.0.10.100) needs to send traffic to H2 (IP2 = 100.0.10.101) but does not know its MAC address. As a result, H1 sends an ARP request to find the MAC address of H2 associated with IP2 (Destination MAC = FF:FF:FF:FF:FF:FF). This ARP request packet is hashed on the LAG bundle by VC (CE-1) to be received by Leaf-2.

```
jnpr@Leaf-2> monitor traffic interface ae0 no-resolve
16:09:51.219480 In arp who-has 100.0.10.101 tell 100.0.10.100
```

3. **Local Learning:** Leaf-2 reflects in the EVPN database that MAC1 for H1 has been learned from local interface ae0 (flagged as DL in the MAC table).

```
jnpr@Leaf-2> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch
```

```
jnpr@Leaf-2> show ethernet-switching table vlan-id 10 00:00:1e:63:c8:7c
```

```
Ethernet switching table : 2 entries, 2 learned
Routing instance : default-switch
```

Vlan name	MAC address	MAC flags	Logical interface	Active source
bd5010	00:00:1e:63:c8:7c	DL	ae0.0	

```
jnpr@Leaf-2> show route advertising-protocol bgp 10.1.1.1 evpn-mac-address 00:00:1e:63:c8:7c extensive
```

```
BGP group IBGP-EVPN type Internal
Route Distinguisher: 1.1.1.2:100
Route Label: 5010
ESI: 00:11:11:11:11:11:11:11
Nexthop: Self
Flags: Nexthop Change
Localpref: 100
AS path: [65100] I
Communities: target:1:5010 encapsulation0:0:0:0:vxlan
```

```
BGP group IBGP-EVPN type Internal
Route Distinguisher: 1.1.1.2:100
Route Label: 5010
EST: 00:11:11:11:11:11:11:11:11:11
Nexthop: Self
Flags: Nexthop Change
```

```

Localpref: 100
AS path: [65100] I
Communities: target:1:5010 encapsulation0:0:0:0:vxlan

```

```

jnpr@Leaf-2> show route advertising-protocol bgp 10.1.1.4 evpn-mac-address 00:00:1e:63:c8:7c extensive

```

```

default-switch.evpn.0: 26 destinations, 26 routes (26 active, 0 holddown, 0 hidden)

```

```

* 2:10.1.1.2:100::5010::00:00:1e:63:c8:7c/304 (1 entry, 1 announced)

```

```

BGP group IBGP-EVPN type Internal
Route Distinguisher: 1.1.1.2:100
Route Label: 5010
ESI: 00:11:11:11:11:11:11:11:11:11:11:11
Nexthop: Self
Flags: Nexthop Change
Localpref: 100
AS path: [65100] I
Communities: target:1:5010 encapsulation0:0:0:0:vxlan

```

In the MAC table for both Leaf-1 and Leaf-2, MAC1 for H1 is reflected as being learned local interface ae0. In the MAC table at Leaf-2, MAC1 is flagged as 'DL'(locally learned) as indicated in step 3. After Leaf-1 learns the Type 2 route from Leaf-2, in the Leaf-1 MAC table MAC1 is flagged as 'DR' (learned from the multihomed peer Leaf-2 and reachable via the local access interface ae0).

MAC1 from Leaf-2('Remote Origin' - no 'Local Origin') in EVPN database on Leaf-1:

```

-----
jnpr@Leaf-1> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch

```

```

VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
Source: 00:11:11:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
Remote origin: 10.1.1.2
Timestamp: Dec 31 17:19:38 (0x5868592a)
State: <Remote-To-Local-Adv-Done>

```

MAC1 flagged as 'DR' in the ethernet-switching table on Leaf-1:

```

-----
jnpr@Leaf-1> show ethernet-switching table vlan-id 10 00:00:1e:63:c8:7c

```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 2 entries, 2 learned

Routing instance : default-switch

Vlan name	MAC address	MAC flags	Logical interface	Active source
bd5010	00:00:1e:63:c8:7c	DR	ae0.0	

Remote peers Leaf-3 (and Leaf-4 – not shown for brevity) will receive Type 2 routes for H1 only from Leaf-2. Leaf-3 and Leaf-4 will associate H1 – MAC1 with the corresponding VX-LAN tunnels to Leaf-1 and Leaf-2, thus load balancing traffic to both Leaf-1 and Leaf-2.

Remote peers receive Type 2 route for H1 only from Leaf-2 (esi NH):

```

-----
jnpr@Leaf-3> show route table bgp.evpn.0 evpn-mac-address 00:00:1e:63:c8:7c

```

```

bgp.evpn.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)

```

+ = Active Route, - = Last Active, * = Both

```

2:10.1.1.2:100::5010::00:00:1e:63:c8:7c/304
*[BGP/170] 00:00:01, localpref 100, from 10.1.1.2
AS path: I, validation-state: unverified
to 10.10.10.12 via et-0/0/1.0
> to 10.10.10.22 via et-0/0/5.0

```

```

jnpr@Leaf-3> show ethernet-switching vxlan-tunnel-end-point remote mac-table
MAC flags (S - static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
SE - Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

```

```

Logical system : <default>
Routing instance : default-switch
Bridging domain : bd5010+10, VLAN : 10, VNID : 5010
MAC          MAC          Logical      Remote VTEP
address      flags        interface   IP address
00:00:1e:63:c8:7c  DR          esi.1670    10.1.1.2

```

Traffic load-balanced to both Leaf-1 and Leaf-2 (esi NH resolves to remote VTEP-1,2):

```

-----
jnpr@36Q-3> show ethernet-switching vxlan-tunnel-end-point esi
ESI          RTT          VLNBH INH    ESI-IFL    LOC-IFL    #RVTEPs
00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11 default-switch 1670 524289 esi.1670      2
RVTEP-IP      RVTEP-IFL    VENH      MASK-ID    FLAGS
10.1.1.2      vtep.32770   1671      1          2
10.1.1.1      vtep.32769   1664      0          2

```

Similarly, Host 2 (H2) MAC2 information is exchanged between all peers:

Remote MAC2 learned on Leaf-2 and Leaf-1 (not shown for brevity):

```

-----
jnpr@Leaf-2> show evpn database l2-domain-id 5010
Instance: default-switch
VLAN VNI  MAC address      Active source      Timestamp      IP address
5010 00:00:1e:63:c8:7c 00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11 Dec 31 17:42:12
5010 00:00:1e:63:d1:fd 00:22:22:22:22:22:22:22:22:22:22:22:22:22:22:22 Dec 31 17:42:13
5010 00:00:5e:00:01:01 05:00:01:87:06:00:00:13:92:00 Dec 31 16:49:54 100.0.10.1
5010 80:ac:ac:2e:ca:cc 1rb.5010          Dec 31 16:49:54 100.0.10.3

```

Once traffic starts, Leaf-1 will eventually learn MAC1 address from local interface ae0. Leaf-1 will then advertise Type 2 EVPN routes for H1. MAC1 will show up with 'DLR' MAC flag in the MAC table at both Leaf-1 and Leaf-2 (locally learned and from multi-homed peer):

MAC1 learned locally ('Local Origin') in EVPN database on Leaf-1:

```

-----
jnpr@Leaf-1> show evpn database extensive mac-address 00:00:1e:63:c8:7c
Instance: default-switch

```

```

VN Identifier: 5010, MAC address: 00:00:1e:63:c8:7c
Source: 00:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11, Rank: 1, Status: Active
Local origin: ae0.0
Remote origin: 10.1.1.2
Timestamp: Dec 31 18:09:38 (0x586864e2)
State: <Local-MAC-Only Local-To-Remote-Adv-Allowed>

```

Type 2 route for MAC1 advertised from Leaf-1 to Leaf-2,3,4:

```

-----
jnpr@Leaf-1> show route advertising-protocol bgp 10.1.1.2 | match 00:00:1e:63:c8:7c
2:10.1.1.1:100::5010::00:00:1e:63:c8:7c/304

```

```

jnpr@Leaf-1> show route advertising-protocol bgp 10.1.1.3 | match 00:00:1e:63:c8:7c
2:10.1.1.1:100::5010::00:00:1e:63:c8:7c/304

```

```

jnpr@Leaf-1> show route advertising-protocol bgp 10.1.1.4 | match 00:00:1e:63:c8:7c
2:10.1.1.1:100::5010::00:00:1e:63:c8:7c/304

```

MAC flag change from 'DR'/'DL' to 'DLR' on Leaf-1/Leaf-2:

```

jnpr@Leaf-1> show ethernet-switching table vlan-id 10 00:00:1e:63:c8:7c
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)
Ethernet switching table : 2 entries, 2 learned
Routing instance : default-switch
Vlan      MAC      MAC      Logical      Active
name      address  flags    interface    source
bd5010    00:00:1e:63:c8:7c  DLR      ae0.0

```

```

jnpr@Leaf-2> show ethernet-switching table vlan-id 10 00:00:1e:63:c8:7c

```

```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)
Ethernet switching table : 2 entries, 2 learned
Routing instance : default-switch
Vlan      MAC      MAC      Logical      Active
name      address  flags    interface    source
bd5010    00:00:1e:63:c8:7c  DLR      ae0.0

```

Traffic Forwarding (Data Plane)

For traffic from H2, lookup is done in the MAC table on Leaf-3 and Leaf-4, and resolved over VXLAN tunnels for Leaf-1 and Leaf-2. Route label for VNI 5010 is used for forwarding this traffic as VXLAN packets. At the destination VTEPs (Leaf-1 and Leaf-2), destination MAC lookup results in the packet being switched out towards H1. The process is repeated for forwarding traffic in the other direction as well.

Traffic from H2(100.0.10.101) to H1(100.0.10.100):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	100.0.10.101	100.0.10.100	UDP	78	Source port: 7344 Destination port: 2935
2	0.00100213	100.0.10.101	100.0.10.100	UDP	1500	Source port: 22034 Destination port: 52044
3	0.00200152	100.0.10.101	100.0.10.100	UDP	512	Source port: 2301 Destination port: 64302

Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
Ethernet II, Src: 00:00:1e:63:d1:fd (00:00:1e:63:d1:fd), Dst: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
802.1Q Virtual LAN
Internet Protocol Version 4, Src: 100.0.10.101 (100.0.10.101), Dst: 100.0.10.100 (100.0.10.100)
User Datagram Protocol, Src Port: 7344 (7344), Dst Port: 2935 (2935)
Data (28 bytes)

NH for H1 MAC lookup resolves to remote VTEPs - Leaf-1 and Leaf-2:

```

jnpr@Leaf-3> show ethernet-switching table vlan-id 10 00:00:1e:63:c8:7c
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)
Ethernet switching table : 2 entries, 2 learned
Routing instance : default-switch
Vlan      MAC      MAC      Logical      Active
name      address  flags    interface    source
bd5010    00:00:1e:63:c8:7c  DR       esi.1704     00:11:11:11:11:11:11:11:11:11:11:11

```

```

jnpr@Leaf-3> show ethernet-switching vxlan-tunnel-end-point esi
ESI      RTT      VLNBH INH      ESI-IFL      LOC-IFL      #RVTEPs
00:11:11:11:11:11:11:11:11:11:11:11 default-switch 1704 524291 esi.1704 2
RVTEP-IP  RVTEP-IFL  VENH  MASK-ID  FLAGS
10.1.1.1  vtep.32771 1718   1        2
10.1.1.2  vtep.32769 1665   0        2

```

```

jnpr@Leaf-3> show interfaces extensive vtep
Logical interface vtep.32769 (Index 544)
VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.2
Traffic statistics:
Input packets:      56931908      267 pps
Output packets:     55455508      254 pps
Logical interface vtep.32771 (Index 546)

```

VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.1

Traffic statistics:

Input packets: 54586419 256 pps
Output packets: 52667367 248 pps

Traffic load balanced on all input and output links:

Interface	Link	Input packets	(pps)	Output packets	(pps)
Leaf-1			Seconds: 94		Time: 18:59:24
et-0/0/1	Up	52091736	(251)	51005664	(239)
et-0/0/5	Up	50802434	(242)	55302523	(267)
et-0/0/34	Up	51077426	(499)	52575497	(483)
ae0	Up	51097882	(500)	52595295	(483)
Leaf-2			Seconds: 4		Time: 19:00:25
et-0/0/1	Up	21091687765	(263)	29300862443	(269)
et-0/0/5	Up	21062291737	(267)	29058140419	(243)
et-0/0/34	Up	65952703	(503)	66545413	(519)
ae0	Up	126574547	(504)	123359686	(519)
Leaf-3			Seconds: 4		Time: 19:02:05
et-0/0/1	Up	57322582	(269)	65313029	(252)
et-0/0/5	Up	55684243	(262)	55027966	(259)
et-0/0/34	Up	633997628	(502)	834286493	(500)
ae0	Up	634013157	(500)	834301964	(499)
Leaf-4			Seconds: 2		Time: 19:07:13
et-0/0/1	Up	69870720	(244)	55883038	(262)
et-0/0/5	Up	52735702	(264)	52319499	(247)
et-0/0/34	Up	513213141	(500)	964894105	(502)
ae0	Up	1425430620	(501)	1100461618	(504)

Inter – subnet: Between hosts in different VLANs

100.0.10.100 (H1 – VLAN 10/VNI 5010) <-> 100.0.20.101 (H4 – VLAN 20/VNI 5020)

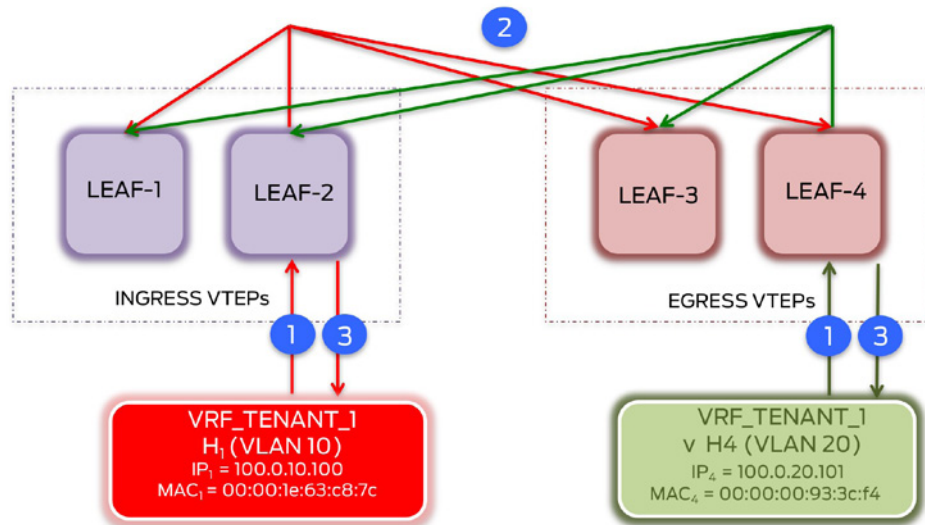


Figure 1.11 MAC learning (Control Plane)

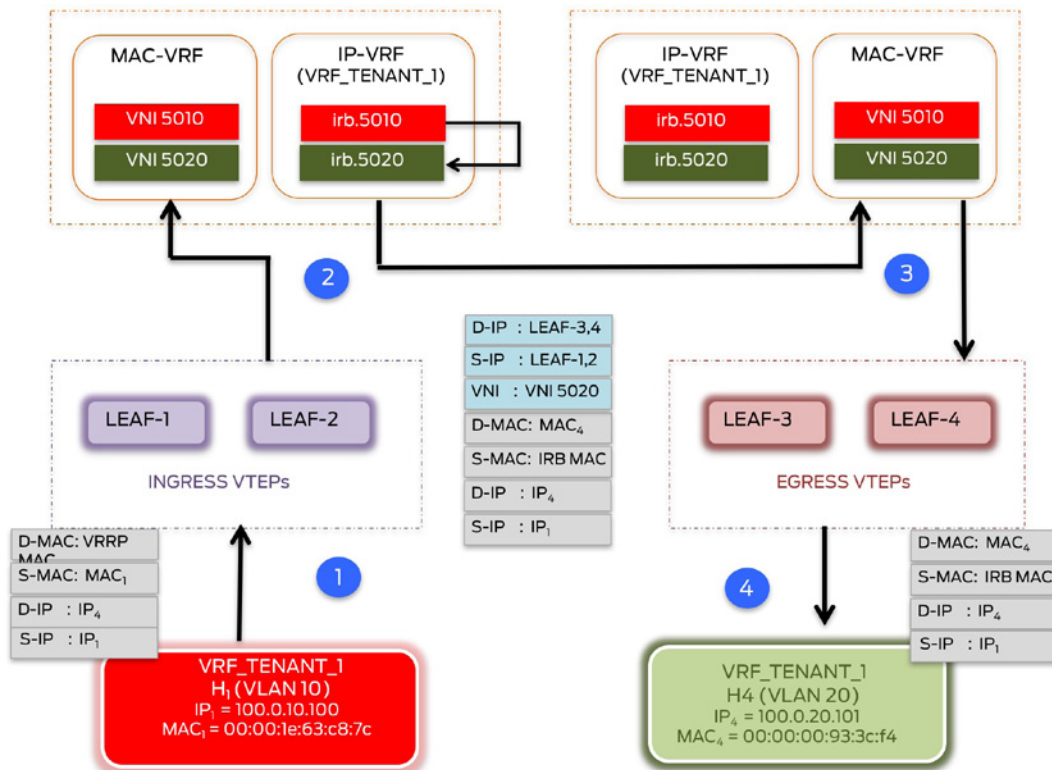


Figure 1.12 Traffic forwarding H1 > H4 (Data Plane)

MAC learning (Control Plane)

Once the EVPN configuration is done, even before any end-hosts are learned locally, Type 1, 3, and 4 routes are exchanged between the EVPN PEs. Type 2 routes for the IRB anycast address are also exchanged.

1. End-hosts in each VLAN send ARP requests for their respective gateways, which could be hashed to any one of the multi-homed PEs. H1 sends an ARP request for 100.0.10.1 received by LEAF-2 while H4 sends an ARP request for 100.0.20.1 received by LEAF-4.

2. The received ARP request triggers MAC (end-host MAC entry in Ethernet-switching table), IP (host route entry in the IP-VRF), and ARP (host ARP entry in the ARP table) updates on LEAF-2 and LEAF-4. Type 2 routes (MAC and MAC+IP) are generated with only one VNI (VNI 5010 for H1 and VNI 5020 for H4) by these L3 gateways to propagate end-host reachability information to all other PEs. EVPN route exchange is similar to that outlined for the preceding intra-subnet traffic flow.

3. The receiving L3 gateway also sends an ARP response to the initiating end-host resolving the virtual gateway IP address to VRRP MAC. If not statically configured, the chassis MAC address is used as the IRB interface MAC address (seen from the source MAC - Ethernet frame on the ARP response packet).

Here the assumption is made that gateways know the destination host's MAC address. If not known, the gateways will send out an ARP request to resolve.

ARP request from H1 received on LEAF-2:

```
-----
jnpr@LEAF-2> show chassis mac-addresses
Base address 80:ac:ac:2e:ca:cc
jnpr@LEAF-2> show interfaces irb | match address:
Current address: 80:ac:ac:2e:ca:cc, Hardware address: 80:ac:ac:2e:ca:cc
```

Time	Source	Destination	Protocol	Length	Info
1 0.00000000	00:00:1e:63:c8:7c	ff:ff:ff:ff:ff:ff	ARP	46	who has 100.0.10.1? Tell 100.0.10.100
2 0.00239300	80:ac:ac:2e:ca:cc	00:00:1e:63:c8:7c	ARP	60	100.0.10.1 is at 00:00:5e:00:01:01

```

Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 80:ac:ac:2e:ca:cc (80:ac:ac:2e:ca:cc), Dst: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
Destination: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
Source: 80:ac:ac:2e:ca:cc (80:ac:ac:2e:ca:cc)
Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN
Address Resolution Protocol (reply)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (2)
Sender MAC address: 00:00:5e:00:01:01 (00:00:5e:00:01:01)
Sender IP address: 100.0.10.1 (100.0.10.1)
Target MAC address: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
Target IP address: 100.0.10.100 (100.0.10.100)

```

ARP request from H4 received on LEAF-4:

```
-----
jnpr@LEAF-4> show chassis mac-addresses
Base address 80:ac:ac:97:15:a2
jnpr@LEAF-4> show interfaces irb | match address:
Current address: 80:ac:ac:97:15:a2, Hardware address: 80:ac:ac:97:15:a2
```

2 5.45154300	00:00:00:93:3c:f4	ff:ff:ff:ff:ff:ff	ARP	46	who has 100.0.20.1? Tell 100.0.20.101
3 5.45329100	80:ac:ac:97:15:a2	00:00:00:93:3c:f4	ARP	60	100.0.20.1 is at 00:00:5e:00:01:01

```

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c), Dst: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: 00:00:1e:63:c8:7c (00:00:1e:63:c8:7c)
Sender IP address: 100.0.10.100 (100.0.10.100)
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
Target IP address: 100.0.10.1 (100.0.10.1)

```

Traffic forwarding (Data Plane)

Summarized below is a walkthrough for traffic from H1 to H4. The same steps apply for return traffic from H4 to H1.

1. To communicate with H4, H1 sends traffic to its default gateway – irb.5010 (D-MAC = VRRP MAC). Traffic is hashed across all member links of the lag interface between the VC and the ingress VTEPs.

MAC rewrite - LEAF-2 (ingress) sets DMAC to H4 MAC for traffic towards egress VTEPs:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	100.0.10.100	100.0.20.101	UDP	1546	96-159 Len=1450
2	0.005000030	100.0.10.100	100.0.20.101	UDP	1546	97-160 Len=1450
3	0.005000020	100.0.10.100	100.0.20.101	UDP	1546	98-161 Len=1450

> Frame 1: 1546 bytes on wire (12368 bits), 1546 bytes captured (12368 bits) on interface 0
 # Ethernet II, Src: JuniperH_2e:ca:e3 (80:ac:ac:2e:ca:e3), Dst: JuniperH_7f:ba:83 (80:ac:ac:7f:ba:83)
 > Destination: JuniperH_7f:ba:83 (80:ac:ac:7f:ba:83)
 > Source: JuniperH_2e:ca:e3 (80:ac:ac:2e:ca:e3)
 Type: IPv4 (0x0800)
 Frame check sequence: 0x333f6a2e [correct]
 [FCS Status: Good]
 > Internet Protocol Version 4, Src: 10.1.1.2, Dst: 10.1.1.4
 > User Datagram Protocol, Src Port: 28699, Dst Port: 4789
 # Virtual extensible Local Area Network
 > Flags: 0x0800, VXLAN Network ID (VNI)
 Group Policy ID: 0
 VXLAN Network Identifier (VNI): 5020
 Reserved: 0
 # Ethernet II, Src: JuniperH_2e:ca:cc (80:ac:ac:2e:ca:cc), Dst: 00:00:00:93:3c:f4 (00:00:00:93:3c:f4)
 > Destination: 00:00:00:93:3c:f4 (00:00:00:93:3c:f4)
 > Source: JuniperH_2e:ca:cc (80:ac:ac:2e:ca:cc)
 Type: IPv4 (0x0800)
 > Internet Protocol Version 4, Src: 100.0.10.100, Dst: 100.0.20.101
 > User Datagram Protocol, Src Port: 96, Dst Port: 159
 > Data (1450 bytes)

4. When the VXLAN encapsulated packet is received by the egress VTEPs (LEAF-3 and LEAF-4), VNI 5020 is used to identify the MAC-VRF. The egress nodes decapsulate the VXLAN header, to perform a MAC lookup. This yields the outbound interface for the respective bridge domain to which the Ethernet frame must be forwarded, thus bridging the packet to H4. With asymmetrical routing mode, ingress VTEPs (connected to the source host) perform both routing and bridging, while the egress VTEPs (connected to the destination host) perform only bridging. Similar steps are performed for the return traffic from H4 to H1. Forwarding behavior on the egress nodes is similar to intra-subnet (Intra VNI) forwarding. Only the ingress nodes, perform all packet processing functions needed for inter-subnet forwarding and so it is called *Asymmetric IRB* (EVPN Type 2).

```

jnpr@LEAF-4> show ethernet-switching table 00:00:00:93:3c:f4
Vlan name      MAC address      MAC flags  Logical interface  Active source
bd5020         00:00:00:93:3c:f4  DL         ae0.0
  
```

IXIA Statistics - Intra DC Intra-VRF Inter-Subnet(VNI 5010 <-> VNI 5020)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Packet Loss Duration (ms)	Tx Frame Rate	Rx Frame Rate
H1 <> H4	75,748	75,748	0	0.000	0.000	2,000.000	2,000.000

Interface	Link	Input packets	(pps)	Output packets	(pps)
LEAF-1					
ae0	Up	25290342	(497)	18168916	(514)
LEAF-2					
ae0	Up	25732941	(506)	16980678	(490)
LEAF-3					
ae0	Up	16501678	(506)	9998917	(465)
LEAF-4					
ae0	Up	18318813	(495)	10054607	(537)

Asymmetric vs. Symmetric IRB models

EVPN provides integrated routing and bridging VPN solutions supporting both intra-subnet (Layer 2 extension) and inter-subnet (Layer 3 routing) communication. Asymmetric and Symmetric Integrated Routing and Bridging is a concept applicable to inter-subnet forwarding. Two operational models have been implemented that define the workflow to route traffic between different virtual networks and differ in terms of the number of lookups done by each communicating VTEP, as listed in Table 1.2.

Table 1.2 Asymmetric Versus Symmetric Functionalities

	Asymmetric IRB	Symmetric IRB
Functionality	To process a packet that is to be routed between two VNIs, ingress VTEP performs both an L3 and an L2 lookup, whereas the egress VTEP only does an L2 lookup. The number of lookups on the ingress and egress VTEPs is discrepant, i.e. all the packet processing associated with the inter-subnet forwarding semantics is confined to the ingress VTEP and hence it is called Asymmetric IRB.	To process a packet that is to be routed between two VNIs, ingress and egress VTEPs perform both a L3 and L2 lookup. The number of lookups on the ingress and egress VTEPs is the same, i.e. all the packet processing associated with the inter-subnet forwarding semantics is distributed on both the ingress and egress VTEPs and hence it is called Symmetric IRB.
Advantages	Reduced number of lookups on the receiving VTEP, since the ingress VTEP does a MAC rewrite that holds the destination host MAC address requiring the receiving VTEP to only do a MAC lookup to forward traffic to the desired end-host. As such, most of the heavy-lifting is done by the ingress VTEP. It also results in better enforcement of MAC security, since traffic to unauthorized receivers can be blocked at the ingress itself.	This does not require VTEPs to participate in all VNIs and as a result a VTEP needs to populate only the minimum desired reachability information, providing an advantage with massive scale.
Disadvantages	This requires consistent configuration across the participating VTEPs with the source and destination VNIs, as well as the need to learn reachability information for the same. For example, to route traffic between VNI A (local) and B (remote), ingress VTEP will need to be configured and populate information for VNI A and B, even though it might not have local end-hosts in VNI B. Can lead to increased configuration complexity and lookup tables impacting scalability. However policies can be used to control routes downloaded to the FIB to achieve FIB optimization. VNI scale support is also not a significant constraint.	Additional lookup is required on the receiving VTEP, since the ingress does not do any MAC rewrites to indicate the destination host, but instead makes use of a L3 VNI that provides the VRF context in which the routing lookup at the egress is done.

Inter-VXLAN routing can be achieved by using Type 2 (MAC, MAC+IP) or Type 5 routes. At the time of this writing, for devices used in this testbed (QFX10002 and MX-80), asymmetric mode is supported with Type 2 routes and symmetric mode is supported with Type 5 routes. If no L2 extension is desired for a given bridge domain, then the symmetric model using Type 5 routes is better suited. If L2 extension is desired then since Type 2 routes are already in use for intra-VXLAN communication, they can be employed for inter-VXLAN routing as well. In this case, asymmetric model with Type 2 routes does not have any additional overhead since VNIs are more likely already provisioned on participating VTEPs.

EVPN Type 5 Routes

Please refer to the Appendix for detailed information on EVPN route types. This section focuses on EVPN Type 5 routes and their role for inter-subnet connectivity.

This new route type has a differentiated role from the Type 2 route and addresses all the data center inter-subnet connectivity scenarios in which an IP Prefix advertisement is required. An overlay index is a object used in the IP Prefix route that needs a recursive route resolution on the NVE receiving the IP Prefix route, so that the NVE knows to which egress NVE it needs to forward the packets. Using this new Type 5 route, an IP Prefix may be advertised along with an overlay index that can be a GW IP address or without an overlay index, in which case the BGP next-hop will point at the egress NVE and the MAC in the Router's MAC Extended Community will provide the inner MAC destination address to be used.

Two implementation models are currently supported for Type 5 routes:

- **Pure Type 5 Model (IP-VRF-to-IP-VRF model):** This provides support for Type 5 routes without overlay next-hop and no Type 2 route for recursive route resolution. When the Type-5 route is advertised with the route MAC extended community, the Type-5 route itself can provide all the necessary forwarding information required for sending VXLAN packet in the data plane to the egress NVE. Since no Type 2 route will be used for route recursive resolution, this provisioning model is also called “IP-VRF to IP-VRF” model. So far, we have seen the usage of VNID configured per VLAN corresponding to MAC-VRF, which is carried in VXLAN packets bridged across VTEPs (Layer 2 VNI). Type 5 routes use a global unique VNI provisioned for each tenant IP-VRF that is used to identify the IP-VRF at the egress. This VNI corresponding to the IP-VRF is carried in VXLAN packets routed across VTEPs (Layer 3 VNI). A chassis MAC will be used as the inner DMAC for the VXLAN packet. The chassis MAC (carried by the Router MAC extended community) will be shared among different tenant IP-VRFs.
- **Gateway Address Model (IRB IP Next-Hop model):** This provides support for Type 5 routes with GW-IP as overlay next-hop and Type 2 route for recursive route resolution. In this case, the IP prefixes advertised by the Type-5 routes are assumed to be seated behind an IRB interface. The IRB’s IP address will be used as the GW-IP address in the Type 5 route. Furthermore, a corresponding Type 2 route will be advertised for the IRB’s MAC/IP addresses and it will be used by the Type 5 for the recursive route resolution.

The packet walkthrough described in the previous section demonstrates the workings of an asymmetric IRB model. Type 5 routes are symmetric by definition, therefore IP and MAC lookups are required at both ingress and egress NVEs. As a result, ingress VTEP does not need to learn and maintain reachability information for the destination VNI remote hosts. Subsequent sections, using Type 5 routes, focus on the Pure Type 5 model only.

Symmetric IRB Model

Figure 1.13 shows intra-VRF inter-subnet traffic flow for the topology (Figure 1.7) using symmetric IRB model with Pure Type 5 routes. LEAF-1 and LEAF-2 have local end-host only for VLAN 10 (H1 – mapped to VNI 5010), while LEAF-3 and LEAF-4 have local end-hosts only for VLAN 20 (H4 – mapped to VNI 5020). All bridge domains belong to the same tenant (Intra-VRF). An L3 VNI (VNI 1020) has been configured on all VTEPs that maps to the IP-VRF (VRF_TENANT_1) and is used for inter-VXLAN routing. EVPN Type 5 routes are exchanged to advertise summary IP routes associated with a tenant domain. These prefixes are populated in the IP-VRF. Type 5 selection for routes to be advertised can be influenced by using routing policy. Here, LEAF-1 and LEAF-2 generate Type 5 routes to advertise 100.0.10/24 while LEAF-3 and LEAF-4 advertise 100.0.20/24. These routes also carry the Router MAC BGP extended community that contains the chassis MAC address.

1. To send traffic to a host H4 in a different subnet, H1 sends an ARP request for its default gateway: irb.5010 (D-MAC = VRRP MAC).
2. Since the packet received by the ingress VTEPs is locally destined as the destination MAC address, corresponds to the IRB interface anycast address, IP lookup is done in the associated IP-VRF table. The next-hop for the destination host resolves over the L3 VNI (VNID 1020), inner packet DMAC (conveyed by the Router MAC extended community in received Type 5 routes) and VXLAN tunnel destination IP.

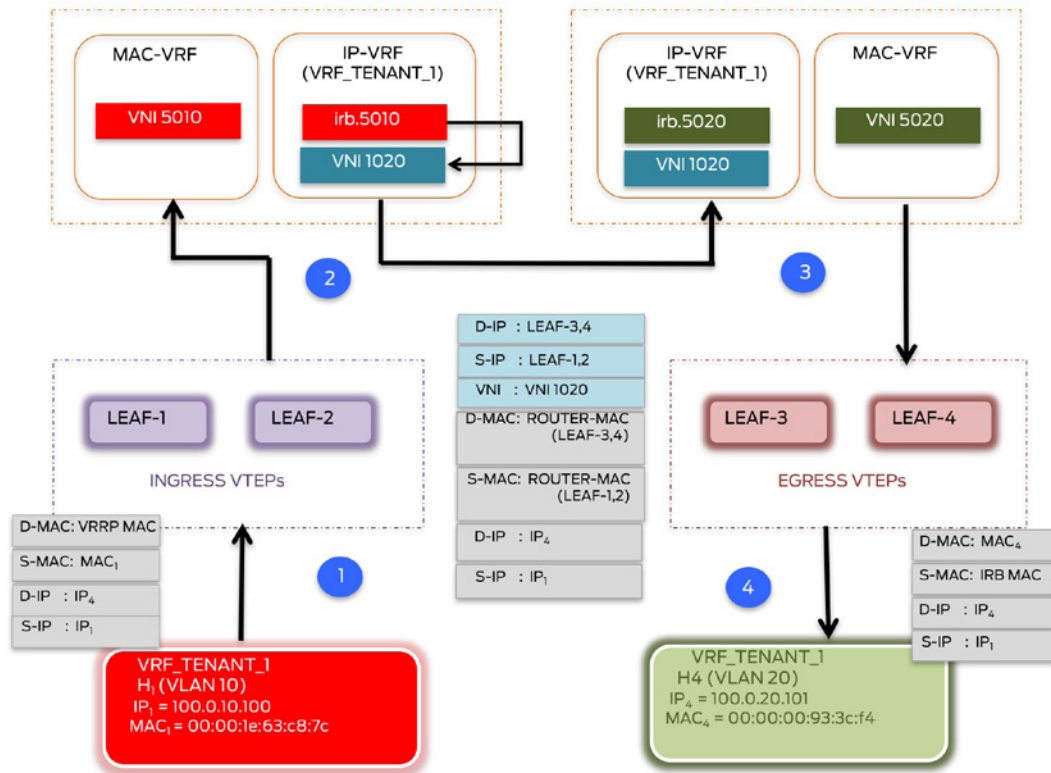


Figure 1.13 Inter-Subnet Traffic Flow (Symmetric IRB Model)

3. Ingress VTEPs rewrite the inner destination MAC address to those of the egress VTEPs as propagated in the Router MAC extended community contained in Type 5 routes received from the egress VTEPs (unique for each VTEP). Packets are encapsulated with VXLAN header with VNID set to the L3 VNI (VNID 1020) and forwarded to egress VTEPs. H₄ is not known to the ingress VTEPs via Layer 2 and hence are routed via the Layer 3 VNI. Egress VTEPs remove the VXLAN header thus decapsulating received traffic. Since the inner DMAC is local (chassis MAC), an IP lookup is done. L3 VNI helps identifying the associated IP-VRF on the egress VTEPs.

4. An IP lookup performed in the routing context concludes that the destination host resides on a local subnet associated to VNI 5020 in the MAC-VRF. Subsequent lookups in the ARP table and MAC-VRF FIB will provide the relevant forwarding information. Traffic is routed from the L3 VNI to the destined L2 VNI that maps to VLAN 20, thus forwarding the intended traffic to the destination host over the corresponding interface.

Summary

The above sections elaborated on essential EVPN and VXLAN related concepts that are fundamental to building next generation data center architectures. Subsequent chapters in this book describe the use of building blocks to construct a data center, and, further, to connect different data centers to realize some of the common user traffic requirements with an EVPNVXLAN architecture. These building blocks are meant to serve as a starting point to the book that can be used to address any other specific use cases or requirements.

NOTE For any questions regarding existing or Juniper roadmap support, please reach out to your account team.)

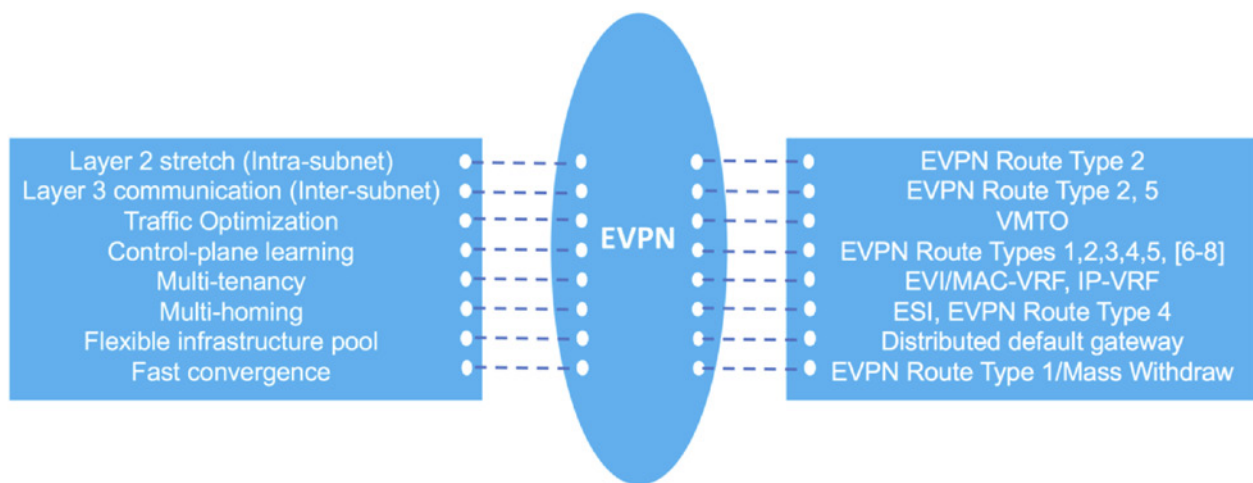


Figure 1.14 Common DC Requirements and Mapping to EVON Constructs

E-W traffic requirements can be either Layer 2 (intra-subnet) or Layer 3 (inter-subnet) in nature, while NS traffic requirements are commonly Layer 3. Distributed default gateway functionality with EVPN-VXLAN has been discussed in the previous sections. A common question that arises on assessing deployment models is: where should the Layer 3 gateway functionality be placed in a fabric – leaf or spine?

Subsequent chapters in this book, delve deeper into achieving E-W and N-S traffic requirements with both variations (routing at leaf or spine), and also show different options for firewall placement to achieve policy enforcement for tenant traffic.

Each of the subsequent chapters in this book present a case study focused on how EVPN-VXLAN can be used for data center deployments.

Chapters 2 and 3 elaborate different design components used to build a data center fabric to satisfy common traffic requirements.

Chapters 4 through 6 discuss different DCI options on how to connect different data centers.

The case studies covered in this book are:

- Building a data center fabric with multi-tenant PODs:
 - Routing at the Leaf (Chapter 2)
 - Routing at the Spine (Chapter 3)
- Data Center Interconnect (DCI):
 - OTT DCI - L3VPN-MPLS core and traffic optimization (Chapter 4)
 - OTT DCI - Public Internet (Chapter 5)
 - Layer 3 DCI - Integration with L3VPN-MPLS core (Chapter 6)

Chapter 2

Building a Data Center – Routing at the Leaf Layer

Contents

Routing at the Leaf Versus the Spine 52

Building a DC with Multi-Tenant PODs Using Routing at the Leaf Layer 54

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet)..... 60

Manual Service Chaining of East-West Traffic 82

Manual Service Chaining of North-South Traffic 91



Chapter 1 elaborated on essential EVPN and VXLAN related concepts fundamental to building next generation data center architectures. This chapter begins by using those building blocks to construct a data center, and further, to connect different data centers to realize some of the common user traffic requirements with an EVPN-VXLAN architecture. The objective for these building blocks is to serve as a starting point that can be further optimized to address any specific use cases or requirements.

IMPORTANT For any questions regarding existing Juniper roadmap support, please reach out to your Juniper account team, or consult www.juniper.net for the most current specifications for your network needs.

Let's first discuss routing at the leaf instead of the spine.

Routing at the Leaf Versus the Spine

With EVPN providing control plane functionality, a VTEP capable of switching packets in the same VNI or same broadcast domain (intra-subnet communication) is called a *Layer 2 Gateway*, whereas a *Layer 3 Gateway* (using IRB interfaces) can route packets across different VNIs or different subnets (inter-subnet communication).

In a leaf-spine fabric, when the Layer 3 gateway resides on the leaf device/s, it is referred to as 'routing at the leaf' and when it resides on spine device/s, it is referred to as 'routing at the spine'. Juniper EVPN implementation provides the flexibility to do routing at the leaf or spine or both, based on the circumstances under consideration. When routing is done at the leaf devices, they can act as both the Layer 2 and Layer 3 gateway (consolidated or collapsed Layer 2/Layer 3 gateway functionality). When routing is done at the spine devices, the leaf devices can still act as Layer 2 gateways while the spine devices can act as Layer 3 gateways. As a result, routing at the spine, makes it possible to use switches which are incapable of VXLAN routing to still act as Layer 2 gateways, allow network operators to build a more cost-effective solution.

In cases where hosts in different subnets are connected to the same leaf device/s, then routing at the leaf can prevent hair pinning and added latency encountered on being routed through the spine, thus providing traffic optimization through local routing and bridging. However, many times in production deployments, end-hosts could be dispersed across various leaf devices. In this case, routing at the spine devices could provide for better functionality distribution to support scaled-out architectures, as traffic between hosts on dispersed leaf devices will be transiting through the spine devices. In this case, there is no minimized latency advantage gained by routing at the leaf. It can offer optimal resource utilization through demarcation of responsibility, as leaf devices could be Layer 2 gateways only while spine devices could act as overlay route reflectors employing Layer 3 gateway functionality within the data center for better integration with DC edge gateways. As more feature-rich or higher-density spine devices would be used for connecting the leaf nodes, using the same as Layer 3 gateways and integration with service devices like firewalls utilizes these switches to their fullest potential. Better integration with service devices can also be achieved, for example, in an IP fabric consisting of ten leaf devices and two spine devices; service devices like firewall, load-balancer, etc., can be used for service chaining by connecting them to just two spine devices instead of ten leaf devices. Connecting these to border leaf devices still introduces an extra hop and added latency as compared to when a service device is connected at the spine layer itself.

Figures 2.1 through 2.3 illustrate this flexible Layer 3 gateway placement.

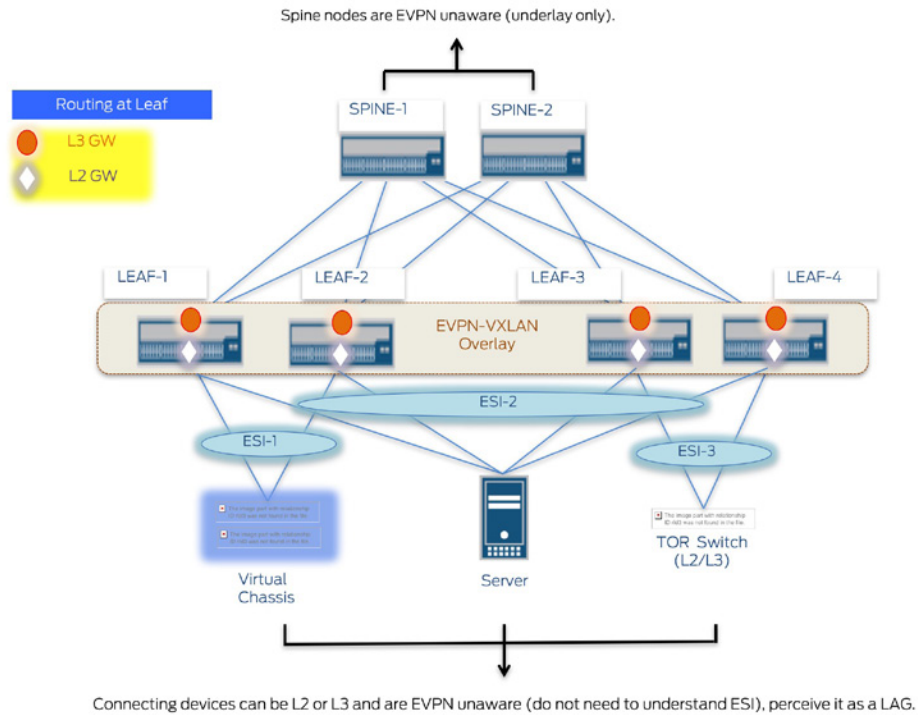


Figure 2.1 Flexible Layer 3 Gateway Placement: Routing at the Leaf Layer

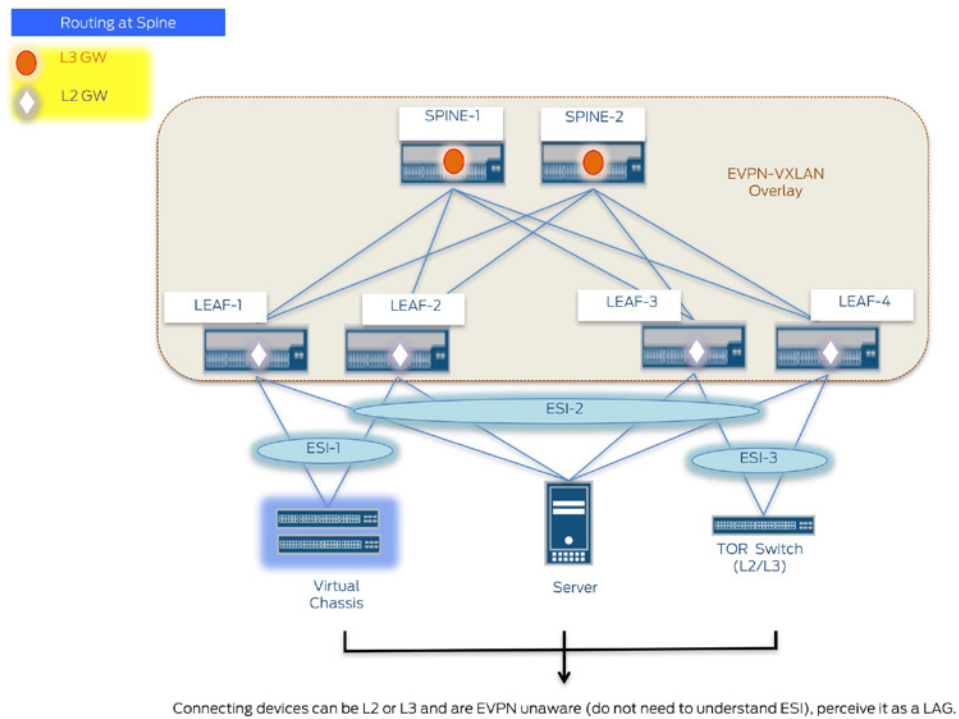


Figure 2.2 Flexible Layer 3 Gateway Placement: Routing at Spine Layer

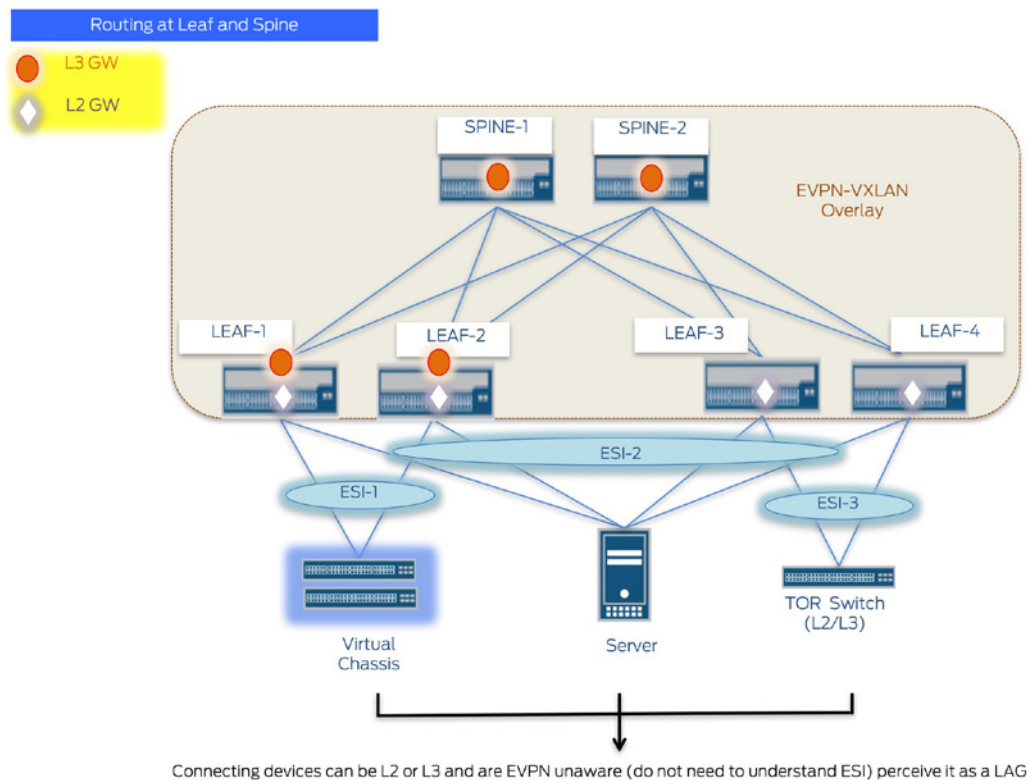


Figure 2.3 Flexible Layer 3 Gateway Placement: Routing at the Leaf and Spine

In this chapter, you will explore the building blocks involved in building a data center with multi-tenant PODs using routing at the leaf layer. The case study is categorized into three main sections:

- High-level summary: Overview on the projected design and products being positioned.
- Design summary: Technical details on the involved design components.
- Traffic scenarios: Delves further into configuration and verification details on common traffic use cases

NOTE For readers only interested in gaining design insight but not detailed configuration, you can skip the *Traffic Scenarios* section.

Building a DC with Multi-Tenant PODs Using Routing at the Leaf Layer

High-Level Summary

In this design approach, routing is done at the leaf layer thus making it possible to reduce functionality and scale requirements at the spine layer. Fabric/super-spine and spine devices act only as transit and overlay route-reflectors. More powerful border leaf devices can be employed at the leaf layer, which are responsible for servicing each POD. These border leaf device hosts the intelligence or control-plane logic and necessary scale capabilities to perform

additional functions as compared to the other devices in the IP fabric. Border leaf devices can act as Layer 3 gateways (like all other leaf devices), DC edge devices that connect to WAN and also service nodes, therefore, devices in the IP fabric that direct traffic to the service block which can consist of a firewall, load-balancer, etc. A typical deployment example could use the QFX 5110 as leaf devices, the QFX 10002/8/16 as border leaf devices, the QFX 5200 as spines devices, and the MX or QFX 10000 family as DC edge devices.

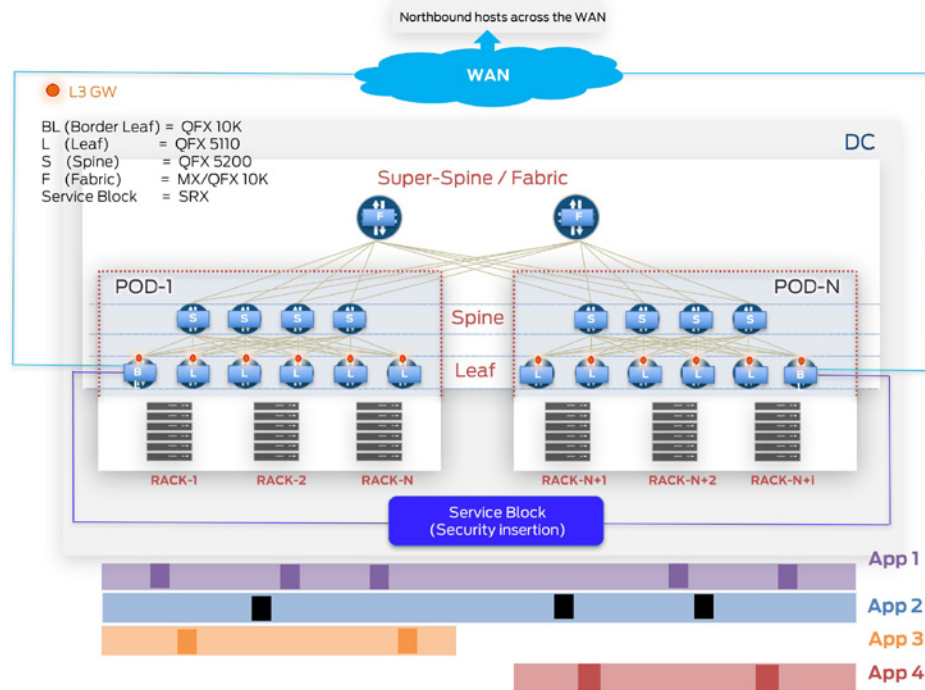


Figure 2.4 High Level Design Representation

Topology Description

- Leaf devices: Due to resource constraints, QFX10002-72Q have been used as both leaf and border leaf devices. Each POD consists of two leaf devices and one border leaf device.
- Spine devices: Four QFX5200-32C-32Q have been used as spine devices, two in each POD.
- Fabric devices: Two MX80-48T have been used as fabric/super-spine devices to connects all PODs.
- Service block: One SRX4200 has been used to demonstrate Layer 3 security insertion for both EW and NS traffic for all PODs. The term service block in this example refers to the SRX acting as the firewall for service chaining Layer 3 inter-tenant traffic.
- CE devices: Access switches (QFX5100-48S-6Q – acting as VC in POD-1 and standalone switch in POD-2) and IXIA simulate CE devices in both PODs. Servers can be directly plugged in to the TOR/leaf devices. To demonstrate the use of LACP, intermediate switches have been used here due to resource constraints. The term host (simulated on IXIA) implies any application entity (VM or container) that consumes an IP/MAC address.

- WAN: Three MX240 routers have been used to simulate WAN PEs while QFX5100-48S-6Q has been used as a P device.

Outlined below are the general data center layout assumptions mapped to the physical testbed for the case study demonstrated here:

- One Row consists of 10 Racks.
- Two leaf devices as redundant TORs are used for each rack.
- Two spine devices connect 10 rows each of 10 racks.
- These two spine devices connecting 10 rows of 10 racks each and leaf devices acting as TORs for each rack in each row, together constitute a POD.
- Border leaf (BL) devices do not require any physical connectivity with CE devices in each rack. One BL device has been used per POD that can reside in the same or different rack as the other leaf devices. BL devices connect each POD to the SRX (Service Block) and the external WAN.
- 10 PODs together constitute a data center.

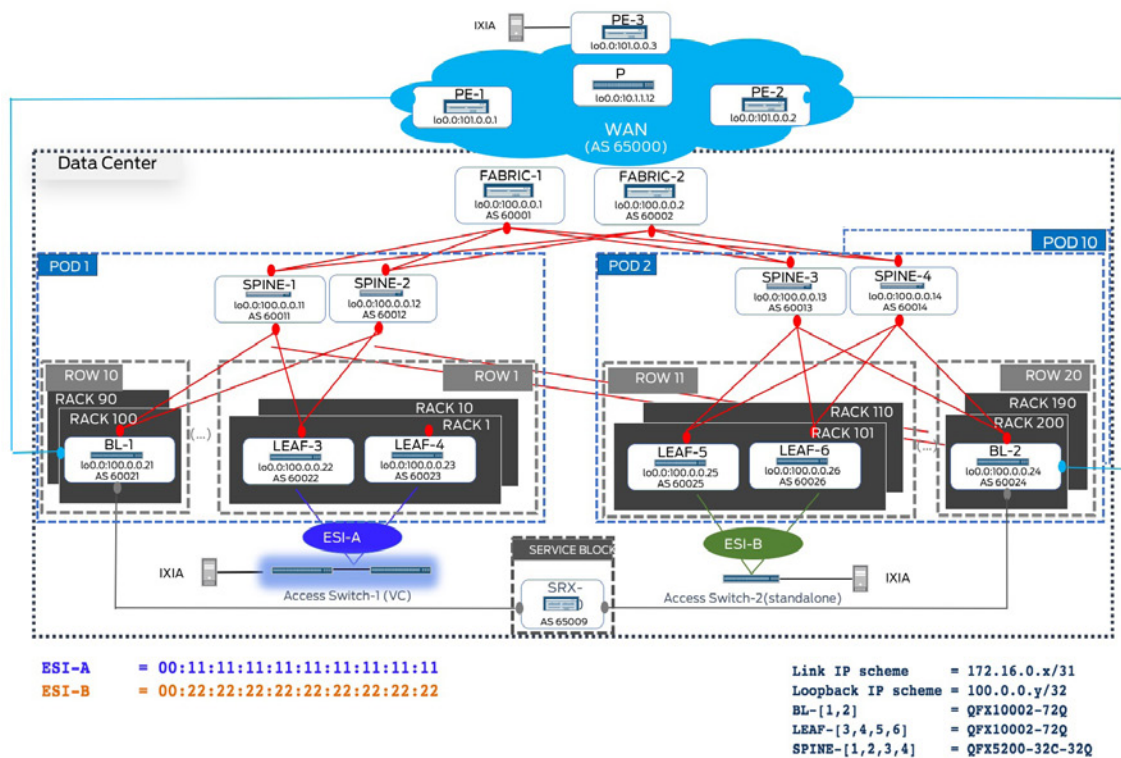


Figure 2.5 Physical Topology

Design Summary

Traffic flows demonstrated:

- East-West (traffic across PODs within the DC):
 - Layer 2 (Intra-VRF Intra-subnet): Traffic between hosts in the same subnet for a given tenant.

- Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant .
- Layer 3 security insertion (Inter-VRF Inter-subnet): Traffic between hosts in different subnets for different tenants and is manually service-chained through the SRX (service block).
- North-South (traffic between hosts across the WAN and within the DC):
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Traffic between hosts in different subnets for different tenants, one of which is external to the data center and is manually service-chained through the SRX (service block).

Access considerations:

- Two leaf devices are acting as a pair of redundant Top of Rack switches. CE-1 represents groups of hosts (e.g. VMs/containers residing on servers) mapped to ESI-A (00:11:11:11:11:11:11:11:11) on ae1 for LEAF-3 and LEAF-4 in POD-1. Similarly, hosts on CE-2 are mapped to ESI-B (00:22:22:22:22:22:22:22:22) on ae1 for LEAF-5 and LEAF-6.

Underlay considerations:

- ISIS has been used to achieve underlay IP reachability (lo0 address reachability between VTEPs). All leaf/spine/fabric devices in all PODs are part of the one ISISLayer 2 domain.
- MTU has been set on all physical interfaces for devices in a data center to account for VXLAN encapsulation.

Overlay considerations:

- Layer 3 gateway placement:
 - VXLAN routing in this use case is done on the leaf devices. Each leaf device acts as a Layer 3 gateway for VNIs residing in that POD. Though not required, border-leaf devices in this example also act as Layer 3 gateways.
 - Routing is done at the leaf layer with spine and fabric devices acting only as transit and have no data-plane configuration, therefore, spine and fabric devices are only overlay route reflectors not VTEPs.
- Distributed Layer 3 gateways:
 - Distributed Layer 3 gateway functionality has been achieved without using the `virtual-gateway` address knob, but instead by statically configuring the same anycast IP/MAC on IRB interfaces of serving gateways for a given subnet. IRB interfaces for subnets of a given tenant, that need to communicate with each other are placed in the same tenant IP-VRF. IP-VRF for tenant 1 (VRF_TENANT_1) hosts irb interfaces for VLANs 10, 20, 30, 80, 90 in POD-1 and 10, 20, 40, 85 in POD-2. IP-VRF for tenant 2 (VRF_TENANT_2) exists only in POD-2 and hosts irb interfaces for VLAN 95. IP-VRF for NS traffic (VRF_TENANT_NS) exists only on the service-node devices in each POD and populate routes received from the WAN, independent of the EVPN domain.
- EVPN NLRI exchange:
 - MP-iBGP sessions have been to exchange EVPN NLRI.

■ Route reflection:

- To avoid full-mesh, MP-iBGP EVPN sessions have been created between the leaf devices (including border-leaf) functioning as clients and spine devices being overlay route-reflectors. In POD-1, BL-1, LEAF-3 and LEAF-4 act as clients to SPINE-1 and SPINE-2 serving as RRs (cluster ID 2.2.2.2 in POD-1). In POD-2, BL-2, LEAF-5 and LEAF-6 act as clients to SPINE-3 and SPINE-4 serving as RRs (cluster ID 3.3.3.3 in POD-2).
- Hierarchical route reflection for the overlay is in use. Fabric devices acting as route-reflectors thus preventing a full-mesh of BGP sessions between spine devices (cluster-ID 1.1.1.1).
- Full mesh of VXLAN tunnels exist with VTEPs located on each leaf and border-leaf node.

■ Service interface:

- VLAN aware EVPN service is in use (please refer to the Appendix for more details on EVPN service interfaces).

■ Host communication:

- Type 2 routes (asymmetric) are used to exchange reachability information for VLANs 10 and 20 (mapped to VNI 5010 and 5020) stretched across PODs, allowing for intra-subnet communication between these hosts. Type 5 routes (symmetric) are used to exchange reachability information for hosts in VLANs 30, 80, 90 (mapped to VNI 5030, 5080, and 5090) in POD-1 and hosts in VLANs 40, 85 (mapped to VNI 5040 and 5085) in POD-2, allowing for inter-subnet communication.
- Layer 2: POD1 <> POD 2 (Intra-VRF/same tenant) --- > Type 2
- Layer 3: POD1 <> POD 2 (Intra-VRF/same tenant) --- > Type 5
- Only VLANs that are stretched across PODs are provisioned on ingress and egress VTEPs. For VLANs that do need to be extended across PODs, provisioning is limited to only those VTEPs, where respective hosts reside. For example, as shown in Figure 2.7's logical topology, VLANs 10 and 20 are stretched across PODs and hence relevant configuration exists on all servicing VTEPs. However, VLANs 80, 90 are confined to POD-1 and are provisioned only on L3 gateways in POD-1 (BL-1, LEAF-3, LEAF-4) while VLANs 85, 95 are confined to POD-2 and are provisioned on L3 gateways in POD-2 (BL-2, LEAF-5, LEAF-6). This provides the advantage of ease of provisioning.

Manual service chaining:

■ Service Nodes:

- Here, border-leaf devices act as service nodes as well as DC edge devices that connect the data center to the WAN. For demarcation of functional responsibilities and need for enhanced features or scale on the DC edge device, it is possible to separate out the DC edge functionality from the border-leaf devices, which is demonstrated in the next chapter.
- The term service node is used for devices that direct traffic from the IP-fabric towards the service block, which in this case are the border-leaf devices in each POD.

■ Service Block

- Layer 3 security insertion has been demonstrated for inter-tenant EW and NS traffic. The service block in this use case consists of a firewall that services all PODs. It is not shown here, but SRX clusters can be used one for each POD or the entire DC, depending on scale and high-availability requirements.

- SRX is physically connected to each border-leaf device. The physical path for traffic between all leaf devices goes through the connecting spine layer. As such, in order to be hair-pinned through the SRX, traffic between a source host residing on a leaf device in POD-1 and destination host residing on a leaf device in POD-2, will take an extra hop to the servicing border-leaf which connects to the SRX.

- Hair-pinning through the SRX is only treated as an IP path problem, therefore, getting intended traffic to the SRX. There is no coverage on details regarding SRX zoning or policy enforcement.

■ Manual service-chaining orchestration for EW inter-tenant inter-subnet traffic

- Layer 3 SRX: POD1 <> POD2 (Inter-VRF/different tenants) --- > An eBGP session (family inet) is created over the connecting link between each border-leaf and SRX that resides in the tenant IP-VRF. All desired host/summary routes (received via Type 5 from EVPN fabric) in that IP-VRF are exchanged through eBGP by border leaf devices in each POD with SRX and [0/0] from SRX advertised into fabric by the border leaf devices.

■ Manual service-chaining orchestration for NS inter-tenant inter-subnet traffic

- Layer 3 SRX: Host across WAN connected to PE-3 <> POD1 (Inter-VRF/different tenants) --- > One additional IP-VRF is created on each border-leaf to receive northbound prefix from the WAN, which is advertised via eBGP session to the SRX and 0/0 from SRX is advertised into fabric. IP-VRFs on all leaf devices only see 0/0 from the SRX.

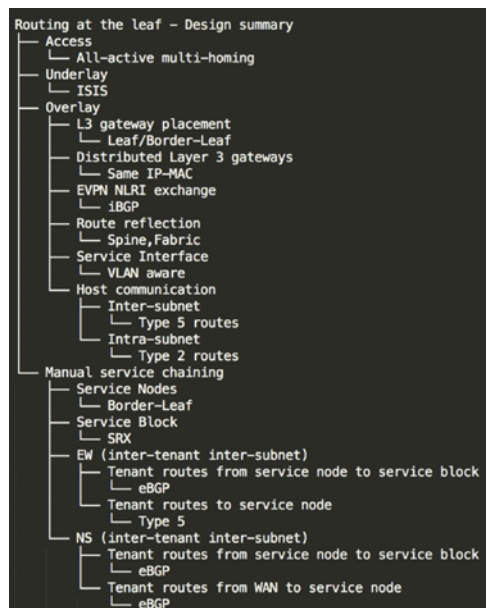


Figure 2.6 Design Summary for Case Study (Routing at Leaf)

Traffic Scenarios

The next three sections will demonstrate implementation details for three primary use cases.

1. East-West traffic for hosts in the same tenant (Intra-VRF: Intra-subnet and Inter-subnet):

- Configuration
- Verification
- Traffic flows

2. Manual service chaining of East-West traffic for hosts in different tenants (Layer 3 security insertion: Inter-tenant Inter-subnet):

- Configuration layout
- Route exchange between service nodes and service blocks
- Configuration
- Verification
- Traffic flows

3. Manual service chaining of North-South traffic for hosts in different tenants (Layer 3 security insertion: Inter-tenant Inter-subnet):

- Configuration layout
- Route exchange between service nodes and service blocks
- Configuration
- Verification
- Traffic flows

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Figure 2.7 highlights the logical topology for E-W traffic across PODs. The section above summarizes the design highlights. In the following sub-sections, details on the configuration and verification steps will be provided.

Configuration

A detailed walkthrough is done for the configuration of only BL-1, LEAF-4, BL-2, LEAF-6, SPINE-1, SPINE-4, and FABRIC-1. All other devices in the testbed are configured similarly.

Configuration is divided into three components – Underlay, Overlay, and Access, and highlighted for leaf, spine, and fabric devices.

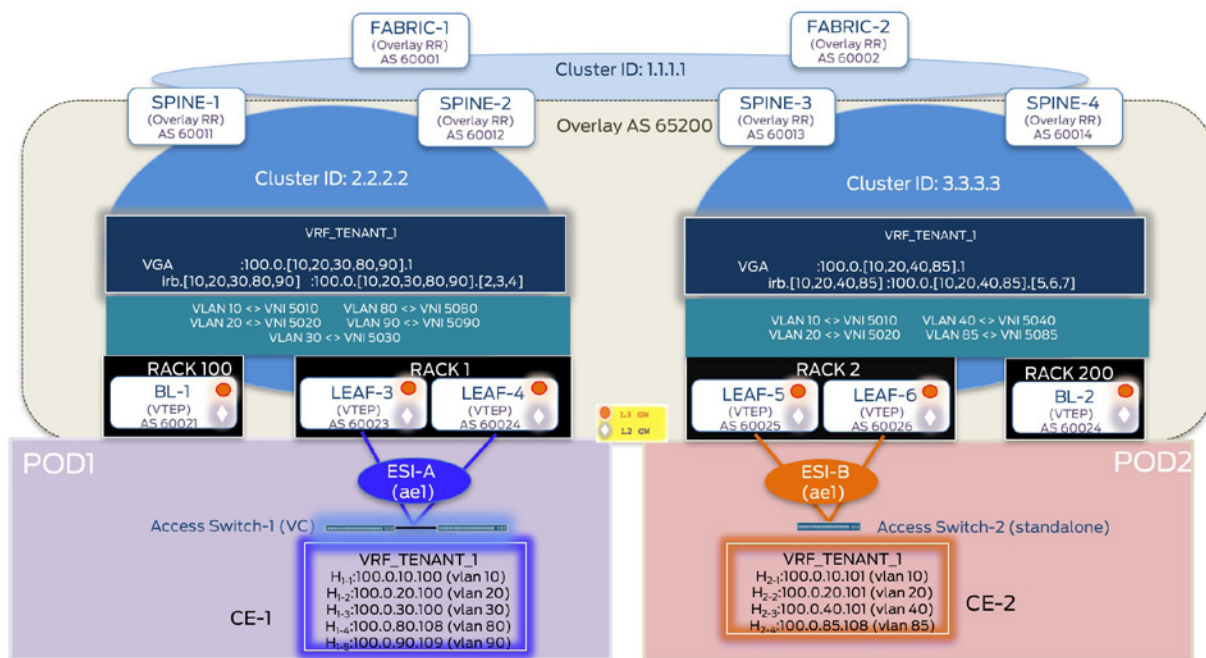


Figure 2.7 Logical Topology – EW traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Underlay

Overhead due to VXLAN encapsulation is accounted for on all devices in the IP fabric:

```
jnpr@BL-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@LEAF-4> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@BL-2> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@LEAF-6> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@SPINE-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@SPINE-4> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@FABRIC-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@BL-1> show configuration groups MTU-VXLAN <<< MTU set to account for overhead due to VXLAN
encapsulation on all leaf, spine and fabric devices

interfaces {
  <*> {
    mtu 9192;
    unit <*> {
      family inet {
        mtu 9000;
      }
    }
  }
}
```

All IP-fabric devices (border-leaf, leaf, spine, fabric) in both PODs are part of a single ISIS Layer 2 domain.

```
jnpr@BL-1> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.21/32;
    }
    family iso {
        address 49.0001.0010.0100.1021.00;
    }
}
unit 10 {
    family inet {
        address 100.0.21.10/32;
    }
}
```

```
jnpr@BL-1> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```
jnpr@LEAF-4> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.23/32;
    }
    family iso {
        address 49.0001.0010.0100.1023.00;
    }
}
unit 10 {
    family inet {
        address 100.0.23.10/32;
    }
}
```

```
jnpr@LEAF-4> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```
jnpr@BL-2> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.24/32;
    }
    family iso {
        address 49.0001.0010.0100.1024.00;
    }
}
unit 10 {
    family inet {
        address 100.0.24.10/32;
    }
}
unit 20 {
    family inet {
        address 100.0.24.20/32;
    }
}
```



```
jnpr@BL-2> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```
jnpr@LEAF-6> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.26/32;
    }
    family iso {
        address 49.0001.0010.0100.1026.00;
    }
}
unit 10 {
    family inet {
        address 100.0.26.10/32;
    }
}
unit 20 {
    family inet {
        address 100.0.26.20/32;
    }
}
```

```
jnpr@LEAF-6> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```
jnpr@SPINE-1> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.11/32;
    }
    family iso {
        address 49.0001.0010.0100.1011.00;
    }
}
```

```
jnpr@SPINE-1> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```
jnpr@SPINE-4> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.14/32;
    }
    family iso {
        address 49.0001.0010.0100.1014.00;
    }
}
```

```
jnpr@SPINE-4> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}
```

```

jnpr@FABRIC-1> show configuration interfaces lo0
apply-groups-except MTU-VXLAN;
unit 0 {
    family inet {
        address 100.0.0.1/32;
    }
    family iso {
        address 49.0001.0010.0100.1001.00;
    }
}

jnpr@FABRIC-1> show configuration protocols isis
level 1 disable;
interface all;
interface lo0.0 {
    passive;
}

```

IGP (ISIS) has been used to establish IP reachability between VTEPs (lo0 address).

```

jnpr@BL-1> show route table inet.0 | match /32 | match 100.0.0
100.0.0.1/32      *[IS-IS/18] 1w2d 00:54:07, metric 20
100.0.0.2/32      *[IS-IS/18] 1w2d 00:26:27, metric 20
100.0.0.11/32     *[IS-IS/18] 1w2d 01:43:49, metric 10
100.0.0.12/32     *[IS-IS/18] 1w2d 08:07:43, metric 10
100.0.0.13/32     *[IS-IS/18] 1w2d 01:43:37, metric 30
100.0.0.14/32     *[IS-IS/18] 1w2d 01:43:37, metric 30
100.0.0.21/32     *[Direct/0] 1w2d 08:48:21
100.0.0.22/32     *[IS-IS/18] 1d 07:34:59, metric 20
100.0.0.23/32     *[IS-IS/18] 1w2d 01:43:44, metric 20
100.0.0.24/32     *[IS-IS/18] 1w2d 01:43:37, metric 40
100.0.0.26/32     *[IS-IS/18] 1w2d 01:43:37, metric 40

```

Overlay

Below are the configuration components required for the overlay configuration.

MP-iBGP Configuration

MP-iBGP sessions exist between leaf devices (acting as clients) and spine devices (overlay route-reflectors) to support EVPN NLRI. BFD has been used for faster failure detection for BGP. Each POD uses a different cluster-ID so as to be able to exchange routes between PODs. Hierarchical route reflection is in use with fabric devices acting as overlay route reflectors (spine devices acting as clients), which prevents full-mesh iBGP requirement between spine devices.

In POD-1, BL-1, LEAF-3, and LEAF-4 (snippets for BL-1 and LEAF-4 follow) act as clients to overlay route-reflectors SPINE-1 and SPINE-2.

```

jnpr@BL-1> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.21;
    family evpn {
        signaling;
    }
}

vpn-apply-export; <<< Applied to all Layer 3 gateways (devices with IP-VRFs so that both VRF export
local-as 65200; and BGP group or neighbor (global protocol hierarchy are evaluated)
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 100.0.0.11;
neighbor 100.0.0.12;
}

jnpr@BL-1> show bfd session detail

```

Address	State	Interface	Detect Time	Transmit Interval	Multiplier
100.0.0.11	Up		1.050	0.350	3
Client BGP, TX interval 0.350, RX interval 0.350					
Session up time 1d 08:21					
Local diagnostic None, remote diagnostic None					
Remote state Up, version 1					
Session type: Multi hop BFD					
Address	State	Interface	Detect Time	Transmit Interval	Multiplier
100.0.0.12	Up		1.050	0.350	3
Client BGP, TX interval 0.350, RX interval 0.350					
Session up time 1d 08:21					
Local diagnostic None, remote diagnostic None					
Remote state Up, version 1					
Session type: Multi hop BFD					

2 sessions, 2 clients

Cumulative transmit rate 5.7 pps, cumulative receive rate 5.7 pps

```
jnpr@LEAF-4> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.23;
    family evpn {
        signaling;
    }
    vpn-apply-export;
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.11;
    neighbor 100.0.0.12;
}
```

In POD-2, BL-2, LEAF-5, and LEAF-6 (snippets for BL-2 and LEAF-6 below) act as clients to overlay route-reflectors SPINE-3 and SPINE-4.

```
jnpr@BL-2> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.24;
    family evpn {
        signaling;
    }
    vpn-apply-export;
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.13;
    neighbor 100.0.0.14;
}
```

```
jnpr@LEAF-6> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.26;
    family evpn {
        signaling;
    }
}
```

```

    vpn-apply-export;
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.13;
    neighbor 100.0.0.14;
}

```

In POD-1, SPINE-1 and SPINE-2 act as overlay route-reflectors (cluster ID 2.2.2.2 in POD-1) for clients BL-1, LEAF-3, and LEAF-4. Hierarchical route reflection is in use as SPINE-1 and SPINE-2 also act as clients to FABRIC-1 and FABRIC-2 acting as overlay route-reflectors. No data-plane (VXLAN) configuration exists on overlay route reflectors (SPINE-1 and SPINE-2 in POD-1).

```

jnpr@SPINE-1> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.11;
    family evpn {
        signaling;
    }
    cluster 2.2.2.2;
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.21;
    neighbor 100.0.0.22;
    neighbor 100.0.0.23;
}
group overlay-hrr-clients {
    type internal;
    local-address 100.0.0.11;
    family evpn {
        signaling;
    }
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.1;
    neighbor 100.0.0.2;
}

```

In POD-2, SPINE-3 and SPINE-4 act as overlay route-reflectors (cluster ID 3.3.3.3 in POD-2) for clients BL-2, LEAF-5, and LEAF-6. Hierarchical route reflection is in use as SPINE-3 and SPINE-4 also act as clients to FABRIC-1 and FABRIC-2 acting as overlay route-reflectors. No data-plane (VXLAN) configuration exists on overlay route reflectors (SPINE-3 and SPINE-4 in POD-2).

```

jnpr@SPINE-4> show configuration protocols bgp
group overlay-evpn {
    type internal;
    local-address 100.0.0.14;
    family evpn {
        signaling;
    }
}

```

```

cluster 3.3.3.3;
local-as 65200;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 100.0.0.24;
neighbor 100.0.0.25;
neighbor 100.0.0.26;
}
group overlay-hrr-clients {
    type internal;
    local-address 100.0.0.14;
    family evpn {
        signaling;
    }
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.1;
    neighbor 100.0.0.2;
}

```

FABRIC-1 and FABRIC-2 acting as overlay route-reflectors (cluster 1.1.1.1) for spine devices as clients in all PODs (SPINE-1,2,3,4). No data-plane (VXLAN) configuration exists on overlay route reflectors (FABRIC-1 and FABRIC-2 connecting all PODs).

```
jnpr@FABRIC-1> show configuration protocols bgp
```

```

group overlay-evpn {
    type internal;
    local-address 100.0.0.1;
    family evpn {
        signaling;
    }
    cluster 1.1.1.1;
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.11;
    neighbor 100.0.0.12;
    neighbor 100.0.0.13;
    neighbor 100.0.0.14;
}
group overlay-evpn-hrr {
    type internal;
    local-address 100.0.0.1;
    family evpn {
        signaling;
    }
    local-as 65200;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 100.0.0.2;
}

```

```
jnpr@BL-1> show configuration policy-options policy-statement pfe-ecmp <<< Applied on all devices
```

```

then {
    load-balance per-packet; }

jnpr@BL-1> show configuration routing-options <<< No global AS has been configured on any device and
local-as is used to establish BGP sessions for overlay.)
router-id 100.0.0.21;
forwarding-table {
    export pfe-ecmp;
}

jnpr@LEAF-4> show configuration routing-options
router-id 100.0.0.23;
forwarding-table {
    export pfe-ecmp;
}

jnpr@BL-2> show configuration routing-options
router-id 100.0.0.24;
forwarding-table {
    export pfe-ecmp;
}

jnpr@LEAF-6> show configuration routing-options
router-id 100.0.0.26;
forwarding-table {
    export pfe-ecmp;
}

jnpr@SPINE-1> show configuration routing-options
router-id 100.0.0.11;
forwarding-table {
    export pfe-ecmp;
}

jnpr@LEAF-3> show configuration routing-options
router-id 100.0.0.25;
forwarding-table {
    export pfe-ecmp;
}

jnpr@SPINE-4> show configuration routing-options
router-id 100.0.0.14;
forwarding-table {
    export pfe-ecmp;
}

jnpr@FABRIC-1> show configuration routing-options
router-id 100.0.0.1;
forwarding-table {
    export pfe-ecmp;
}

```

EVI Configuration

At the time of this writing, QFX 10002 platforms support VLAN aware service using a single global virtual-switch. The EVI related configuration on the QFX 10002 platforms includes the following sub-components:

- **VLAN-VNI mapping:** Under the vlans configuration hierarchy, each local VLAN is mapped to a unique VNI on each VTEP. Currently, leaf nodes with hosts that need to communicate with each other support configuration for the desired VLANs. For example, for hosts in VLAN 10 and VLAN 20 that reside in both POD1 (CE-1 connected to LEAF-3,4 – ESI-A) and POD2 (CE-2 connected to LEAF-5,6 - ESI-B), intra-subnet communication is intended for the same. This is achieved by means of Type 2 routes and the same set of VLANs are configured on all leaf nodes, i.e. VLAN 10 and VLAN 20 are configured on LEAF-3,4,5, and 6 (as also BL-1,2 that are also

acting as collapsed Layer 2/Layer 3 gateways). However, since inter-subnet communication has been established using Type 5 routes (symmetric forwarding), all Layer 3 gateways are not configured with the same set of VLANs. Due to the use of the Layer 3 VNI with Type 5 routes, Layer 3 gateways do not need both ingress and egress reachability information. As a result, BL-1, LEAF-3, 4 in POD1 are configured only for VLAN 30, 80, 90 while BL-2, LEAF-5, 6 in POD2 are configured only for VLAN 40, 85, while still allowing for inter-subnet communication. With 15.1X53-D60, the definition of ingress-node-replication (vlan hierarchy) for handling BUM traffic replication is optional, since that is currently supported.

NOTE Best practice would be to not explicitly specify ingress-node-replication as that interferes with the default optimal behavior.

jnpr@BL-1> show configuration vlans | except # <<< BL-1, LEAF-3, LEAF-4 share the same set of VLANs and VNI mapping

```
bd5010 {
    vlan-id 10;
    l3-interface irb.5010;
    vxlan {
        vni 5010;
    }
}
bd5020 {
    vlan-id 20;
    l3-interface irb.5020;
    vxlan {
        vni 5020;
    }
}
bd5030 {
    vlan-id 30;
    l3-interface irb.5030;
    vxlan {
        vni 5030;
    }
}
bd5080 {
    vlan-id 80;
    l3-interface irb.5080;
    vxlan {
        vni 5080;
    }
}
bd5090 {
    vlan-id 90;
    l3-interface irb.5090;
    vxlan {
        vni 5090;
    }
}
```

jnpr@LEAF-4> show configuration vlans | except # <<< BL-1, LEAF-3, LEAF-4 act as collapsed Level 2/Level 3 gateways (irb interfaces acting as routing interfaces for a given VLAN)

```
bd5010 {
    vlan-id 10;
    l3-interface irb.5010;
    vxlan {
        vni 5010;
    }
}
bd5020 {
    vlan-id 20;
    l3-interface irb.5020;
    vxlan {
        vni 5020;
    }
}
```



```

}
bd5030 {
    vlan-id 30;
    l3-interface irb.5030;
    vxlan {
        vni 5030;
    }
}
bd5080 {
    vlan-id 80;
    l3-interface irb.5080;
    vxlan {
        vni 5080;
    }
}
bd5090 {
    vlan-id 90;
    l3-interface irb.5090;
    vxlan {
        vni 5090;
    }
}

```

jnpr@BL-2> show configuration vlans | except # <<< BL-2,5,6 share the same set of VLANs and VNI mapping

```

bd5010 {
    vlan-id 10;
    l3-interface irb.5010;
    vxlan {
        vni 5010;
    }
}
bd5020 {
    vlan-id 20;
    l3-interface irb.5020;
    vxlan {
        vni 5020;
    }
}
bd5040 {
    vlan-id 40;
    l3-interface irb.5040;
    vxlan {
        vni 5040;
    }
}
bd5085 {
    vlan-id 85;
    l3-interface irb.5085;
    vxlan {
        vni 5085;
    }
}
bd5095 {
    vlan-id 95;
    l3-interface irb.5095;
    vxlan {
        vni 5095;
    }
}

```

jnpr@LEAF-6> show configuration vlans | except #

```

bd5010 {
    vlan-id 10;
    l3-interface irb.5010;
    vxlan {
        vni 5010;
    }
}
bd5020 {
    vlan-id 20;

```

```

    13-interface irb.5020;
    vxlan {
        vni 5020;
    }
}
bd5040 {
    vlan-id 40;
    13-interface irb.5040;
    vxlan {
        vni 5040;
    }
}
bd5085 {
    vlan-id 85;
    13-interface irb.5085;
    vxlan {
        vni 5085;
    }
}
bd5095 {
    vlan-id 95;
    13-interface irb.5095;
    vxlan {
        vni 5095;
    }
}
}

```

```

jnpr@SPINE-1> show configuration vlans | except # <<< Spine and fabric devices do not have any data plane
configuration and act only as control-plane RRs

```

```

jnpr@SPINE-1>

```

```

jnpr@SPINE-4> show configuration vlans | except #

```

```

jnpr@SPINE-4>

```

```

jnpr@FABRIC-1> show bridge domain

```

```

jnpr@FABRIC-1>

```

Virtual Switch Configuration

Under the 'switch-options' hierarchy, the vtep-source-interface parameter specifies the VTEP IP address used to establish the VXLAN tunnel. The loopback interface (e.g. lo0.0) is used for this purpose, reachability to which is provided by the underlay. The route-distinguisher (RD) here defines the EVI specific RD carried by Type 1 – AD per EVI, Type 2, Type 3 routes. The route-target (RT) here defines the global route target inherited by EVPN routes, unless specific RT is defined.

In the configuration snippets below, the RT (switch-options hierarchy) is inherited by Type 1 routes and specific RTs manually configured (protocol evpn hierarchy) are inherited by Type 2 and Type 3 routes. Though the output here is highlighted for LEAF-4 and LEAF-6, similar configuration exists on all other leaf and border-leaf devices.

```

jnpr@LEAF-4> show configuration switch-options
vtep-source-interface lo0.0; <<< VXLAN tunnel source address
route-distinguisher 100.0.0.23:100;
vrf-import LEAF-IN;
vrf-target target:1:100; <<< Global route target for all EVPN routes (here Type 1)
jnpr@LEAF-4> show configuration protocols evpn vni-options <<< Manual configuration - Specific route
targets used by EVPN routes for given VNI (here Type 2 and Type 3)

vni 5010 {
    vrf-target export target:1:5010;
}
vni 5020 {
    vrf-target export target:1:5020;
}

```

```

}
vni 5030 {
    vrf-target export target:1:5030;
}
vni 5080 {
    vrf-target export target:1:5080;
}
vni 5090 {
    vrf-target export target:1:5090;
}

```

jnpr@LEAF-4> show configuration policy-options policy-statement LEAF-IN

term import_leaf_es1 { <<< Import filter to accept routes with the global and specific route targets

```

    from community comm-leaf_es1;
    then accept;
}
term vni5010 {
    from community vni5010;
    then accept;
}
term vni5020 {
    from community vni5020;
    then accept;
}
term vni5030 {
    from community vni5030;
    then accept;
}
term vni5080 {
    from community vni5080;
    then accept;
}
term vni5090 {
    from community vni5090;
    then accept;
}
term default {
    then reject;
}

```

```

community comm-leaf_es1 members target:1:100;
community vni5010 members target:1:5010;
<...>
community vni5090 members target:1:5090;

```

jnpr@LEAF-6> show configuration switch-options

```

vtep-source-interface lo0.0;
route-distinguisher 100.0.0.26:100;
vrf-import LEAF-IN;
vrf-target target:1:100;

```

jnpr@LEAF-6> show configuration protocols evpn

```

vni-options
vni 5010 {
    vrf-target export target:1:5010;
}
vni 5020 {
    vrf-target export target:1:5020;
}
vni 5040 {
    vrf-target export target:1:5040;
}
vni 5085 {
    vrf-target export target:1:5085;
}
vni 5095 {
    vrf-target export target:1:5095;
}

```

```

jnpr@LEAF-6> show configuration policy-options policy-statement LEAF-IN
term import_leaf_esi {
    from community comm-leaf_esi;
    then accept;
}
term vni5010 {
    from community vni5010;
    then accept;
}
term vni5020 {
    from community vni5020;
    then accept;
}
term vni5040 {
    from community vni5040;
    then accept;
}
term vni5085 {
    from community vni5085;
    then accept;
}
term vni5095 {
    from community vni5095;
    then accept;
}
term default {
    then reject;
}

community comm-leaf_esi members target:1:100; <<< Global route-target
community vni5010 members target:1:5010;
<...>
community vni5095 members target:1:5095;

```

IRB Interface Configuration

As highlighted in the previous sections, EVPN PEs capable of inter-subnet routing (Layer 3 gateways) use IRB interfaces to provide this functionality. To realize a distributed anycast gateway, IRB interfaces for a given VNI are distributed across all PEs. The `virtual-gateway-address` knob allows the creation of any ancast IP/MAC that is shared by all servicing IRB interfaces. In a data center environment, wherein Layer 2 only PEs can coexist with Layer 3 PEs, the virtual-gateway function provides a mechanism to achieve traffic load-balancing across multiple Layer 3 gateways from a given Layer 2 only PE.

To achieve the Layer 3 gateway functionality, IRB interface configuration options may or may not include the use of the `virtual-gateway-address` knob.

Without the `virtual-gateway-address` knob, all IRB interfaces for a given subnet can be configured with either same IP/different MAC or different IP/different MAC across all EVPN PEs. As elaborated upon in the *Understanding EVPN* section of Chapter 1, Type 2 routes generated for these IRB interfaces have the default gateway community attached, that helps achieve default gateway synchronization. However, node failures lead to IRB Type 2 route withdrawal of the failing PE. Since the PE receiving end-host traffic destined to the failing node MAC can no longer recognize that the lost IRB MAC belonged to a gateway, it can no longer proxy for the same. As a result, this is not the most optimal configuration option.

Alternatively, the same IP and same MAC can be used as a configuration option across all EVPN PEs. It only requires the same unicast address to be configured on the IRB interface of all Layer 3 gateways for a given subnet, as opposed to a unique unicast and shared anycast required with the `virtual-gateway-address` knob. Default gateway synchronization in this case is manually achieved through the same static IP/MAC address configuration across all Layer 3 PEs.

However, it should be noted that in this case the source MAC on ARP requests originated by the Layer 3 gateway is that of the unicast IRB interface address that is same across all Layer 3 PEs.

As a result, symmetric forwarding will not happen and may result in cases where ARP requests might be generated by one Layer 3 gateway but the ARP response for the same might be consumed by a different, intercepting Layer 3 gateway.

- Static IRB MAC used in the ARP request originated by Layer 3 gateway:

```
jnpr@LEAF-4> monitor traffic interface irb.5010 extensive no-resolve
Address resolution is OFF.
Listening on irb.5010, capture size 1514 bytes

19:32:56.092330 Out
  Juniper PCAP Flags [Ext], PCAP Extension(s) total length 22
    Device Media Type Extension TLV #3, length 1, value: Ethernet (1)
    Logical Interface Encapsulation Extension TLV #6, length 1, value: Ethernet (14)
    Device Interface Index Extension TLV #1, length 2, value: 640
    Logical Interface Index Extension TLV #4, length 4, value: 563
    Logical Unit Number Extension TLV #5, length 4, value: 5010
  -----original packet-----
  00:00:00:01:01:10 > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 46: vlan 10, p 0, ethertype ARP, arp who-has 100.0.10.100 tell 100.0.10.1
```

With the `virtual-gateway-address` knob, all IRB interfaces for a given subnet consist of a unicast address (unique IP/MAC to each PE IRB) and a anycast address (shared across all PEs). The end-hosts use this anycast IP as the gateway address. Node failure does not impact the distributed gateway functionality in this case, since the end-host traffic is destined to the anycast MAC, which is shared across all PEs. As a result, this is the preferred configuration option.

Both configuration options have been demonstrated in this chapter, with supporting knobs required for the same. The `virtual-gateway-address` knob has been demonstrated in the next case study wherein routing is done at spine, but can also be used when routing is done at the leaf.

```
jnpr@LEAF-4> show configuration interfaces irb
unit 5010 {
  description "Tenant 1 - vlan 10 - vni5010";
  family inet {
    <<< All Layer 3 gateways (leaf and border-leaf devices here) are configured with
    the same anycast gateway IP for v4 traffic
    address 100.0.10.1/24;
  }
  mac 00:00:00:01:01:10;
}
```

```
jnpr@LEAF-6> show configuration interfaces irb
unit 5010 {
  description "Tenant 1 - vlan 10 - vni5010";
  family inet {
    <<<
    address 100.0.10.1/24;
  }
  mac 00:00:00:01:01:10;
}
```

Two additional knobs: `virtual-gateway-v4-mac` and `proxy-macip-advertisement` elaborated upon in the next case study, are not required when routing is done at the leaf.

- `virtual-gateway-v4-mac`: Not required, since the same MAC address (unicast IRB MAC address) is used both in the inner and outer packets, there is no flooding on the Layer 2 access switch connecting the end-host to the leaf device.
- `proxy-macip-advertisement`: Not required, since there are no Layer 2 only EVPN PEs and since routing is done at the leaf, MAC and IP routes for the end hosts are generated by the originating Layer 3 gateway.

EVPN Protocol Configuration

Under the global ‘protocols evpn’ hierarchy, the `extended-vni-list` statement lists all advertised VNIs that will be part of the EVPN domain, which in this case are all configured VNIs. Since manual RTs are used, `vni-options` specifies the individual RTs for each VNI. Since ingress-replication is supported by default, the `multicast-mode ingress-replication` statement to define the same is not needed.

Summarized in Table 2.1 are recommendations on the application of the `default-gateway` knobs used at this hierarchy when routing is done at leaf versus the spine. They are outlined here without, or with, the `virtual-gateway-address` knob respectively.

Table 2.1 Application of the Default-Gateway Knobs

	Routing on leaf (same IP/same MAC on all Layer 3 gateways)	Routing on spine (VGA used on all Layer 3 gateways)
<code>default-gateway do-not-advertise</code>	Required (All of the EVPN PEs are configured with the same IP/MAC and hence default-gateway synchronization across PEs is not needed. This knob suppresses the advertisement for IRB Type 2 MAC routes).	Not required (Configuring this knob allows Layer 2 PEs to learn the Layer 3 gateway and have Layer 3 PE’s IRB interface address reachability to help with symmetric forwarding).
<code>default-gateway no-gateway-community</code>	Not required (Does not really matter since same IP/MAC is used across all EVPN PEs).	Required (With this knob configured, Layer 2 PEs don’t try to proxy for Layer 3 gateway IRB and VGA MACs, will just forward traffic destined to these MACs).

As applicable to this case study, since routing is done on the leaf (using the same IP/MAC) `default-gateway do-not-advertise` has been configured under the ‘protocols evpn’ hierarchy on the Layer 3 gateway:

```
jnpr@LEAF-4> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list all;
default-gateway do-not-advertise; <<< With routing being done at the leaf, this knob suppresses
advertisement of Type 2 routes for the irb MAC, as default gateway synchronization is manually done by
means of static configuration using the same IP/MAC across all Layer 3 gateways
vni-options {
    vni 5010 {
        vrf-target export target:1:5010;
    }
    vni 5020 {
        vrf-target export target:1:5020;
    }
    vni 5030 {
        vrf-target export target:1:5030;
    }
    vni 5080 {
        vrf-target export target:1:5080;
    }
    vni 5090 {
        vrf-target export target:1:5090;
    }
}
```

IP-VRF Configuration

A tenant IP-VRF is created for IP routes on a EVPN PE and could be associated with one or more EVIs. The IRB interface, which is the Layer 3 interface for a bridge domain, connects the MAC-VRF and the IP-VRF on an EVPN PE. Only subnets within a VRF

can communicate with each other, allowing for multi-tenancy. Under the ‘routing-instances’ hierarchy, the `instance-type vrf` statement creates an IP-VRF. In this use case, Tenant 1 exists on PODs 1 and 2 and contains IRB interfaces for subnets, hosts within which need to communicate with each other.

A pure EVPN Type 5 model has been used to exchange host routes between all nodes, to allow Inter-VRF communication. All necessary routing information is provided by Type 5 routes and no Type 2 routes are exchanged (VRF-to-VRF model) for VLANs 30, 80, 90 in POD1 and VLANs 40 and 85 in POD2. Export policy within Type 5 configuration allows for the exchange of host prefix routes across PODs.

```
jnpr@LEAF-4> show configuration routing-instances
VRF_TENANT_1 {
    <<< Shared IP VRF across all leaf and border-leaf nodes in POD1 and POD2

    instance-type vrf;
    interface irb.5010;
    interface irb.5020;
    interface irb.5030;
    interface irb.5080;
    interface irb.5090;
    interface lo0.10;      <<< Recommendation would be to configure lo0 unit when VXLAN VNI irb is placed
in the tenant IP-VRF for successful ARP resolution

    route-distinguisher 100.0.23.10:10;
    vrf-import vrf-import-tenant1;
    vrf-target target:10:10;
    vrf-table-label;
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop; <<< Defined for Pure Type 5 routes

                encapsulation vxlan;
                vni 9001;      <<< Globally unique VNI used to identify Layer 3 VRF (VRF_
TENANT_1) in both POD1 and POD2

                export EVPN-host-routes;
            }
        }
    }
}
jnpr@LEAF-6> show configuration routing-instances
VRF_TENANT_1 {
    instance-type vrf;
    interface irb.5010;
    interface irb.5020;
    interface irb.5040;
    interface irb.5085;
    interface lo0.10;
    route-distinguisher 100.0.26.10:10;
    vrf-import vrf-import-tenant1;
    vrf-target target:10:10;
    vrf-table-label;
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-
nethop; <<< Pure Type 5 (Type 5 route carries all attributes for NH – No Type 2 needed)
encapsulation vxlan;
                vni 9001;
                export EVPN-host-routes;
            }
        }
    }
}
```



```
jnpr@LEAF-4> show configuration policy-options policy-statement EVPN-host-routes
<<< Export policy used with Type 5 configuration to exchange host IP prefixes
```

```
term adv-v30 {
    from {
        protocol evpn;
        route-filter 100.0.30.0/24 orlonger;
    }
    then accept;
}
term adv-v80 {
    from {
        protocol evpn;
        route-filter 100.0.80.0/24 orlonger;
    }
    then accept;
}
term adv-v90 {
    from {
        protocol evpn;
        route-filter 100.0.90.0/24 orlonger;
    }
    then accept;
}
term default {
    then reject;
}
```

```
jnpr@LEAF-6> show configuration policy-options policy-statement EVPN-host-routes
```

```
term adv-v40 {
    from {
        protocol evpn;
        route-filter 100.0.40.0/24 orlonger;
    }
    then accept;
}
term adv-85 {
    from {
        protocol evpn;
        route-filter 100.0.85.0/24 orlonger;
    }
    then accept;
}
term adv-v95 {
    from {
        protocol evpn;
        route-filter 100.0.95.0/24 orlonger;
    }
    then accept;
}
term default {
    then reject;
}
```

Access

In this case study, end-hosts (VMs or BMSs) are simulated by the IXIA test tool connected to the leaf devices using the access switches in POD-1 and POD-2. CE-1 and CE-2 simulates hosts connected to Access Switch-1 and Access Switch-2 respectively. CE-1 and CE-2 simulate hosts connected to Access Switch-1 and Access Switch-2, respectively. CE-1 is multi-homed to redundant TORs LEAF-3 and LEAF-4 in POD-1, while CE-2 is multi-homed to redundant TORs LEAF-5 and LEAF-6 in POD-2.

Each CE device views the set of redundant leaf devices as one device by virtue of the same system ID on the LAG interface configured on the multi-homed PEs. All-active multi-homing has been configured on the leaf devices. The LAG interface between PEs and CEs is a trunk interface carrying multiple VLANs. Configured VLANs to VNIs exist in a global virtual-switch. The ESI on LAG interface is configured only on the PE devices. ESI (Type 0) is in use

as indicated by the first byte (set to 0) of the 10-byte ESI value. The remaining 9 bytes are statically defined. Same ESI value is configured on PEs multi-homed to the same ES. Interface hold timers have been configured on LAG members to prevent traffic blackholing when there is a failure, providing sufficient time for control plane to converge.

NOTE At the time of the writing of this book, the LACP bring-down on the core isolation feature was not available, so interface timers have been used.

```

jnpr@LEAF-4> show configuration interfaces ae1
apply-groups-except MTU-VXLAN;
description "ae1(ESI-A) to Access-Sw-1";
mtu 9192;
esi {
    00:11:11:11:11:11:11:11:11:11; <<< LEAF-3 and LEAF-4 (all-
active) share the same ESI-A (00:11:11:11:11:11:11:11:11:11) on LAG ae1
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:00:01;<<< Same LACP System-ID on both LEAF-3 and LEAF-
4 so access switch thinks it is connected to a single device
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members all;
        }
    }
}

jnpr@LEAF-4> show configuration interfaces xe-0/0/18:0
apply-groups-except MTU-VXLAN;
description "LAG members (ae1) to VC";
hold-time up 180000 down 5;

ether-options {
    802.3ad ae1;
}

jnpr@LEAF-6> show configuration interfaces ae1
apply-groups-except MTU-VXLAN;
description "ae1 (ESI-B) to Access-Sw-2";
mtu 9192;
esi { <<< LEAF-5 and LEAF-6 (all-
active) share the same ESI-B (00:22:22:22:22:22:22:22:22:22) on LAG ae1
    00:22:22:22:22:22:22:22:22:22;
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:00:02;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members all;
        }
    }
}

```

```

jnpr@LEAF-6> show configuration interfaces xe-0/0/18:0
apply-groups-except MTU-VXLAN;
description "LAG members (ae1) to AccSw-2";
hold-time up 180000 down 5;
ether-options {
    802.3ad ae1;
}

```

Verification

For VLANs 10 and 20 stretched across PODs, BL-1, LEAF-3, and LEAF-4 learn remote MAC addresses of hosts in POD-2 as well local hosts in POD-1. MAC addresses for remote hosts are learned from the remote VTEPs BL-2, LEAF-5,6 in POD-2, that advertise Type 2 routes for the same.

```

jnpr@LEAF-6> show ethernet-switching table vlan-id 10
Vlan name  MAC address      MAC flags  Logical interface  Active source
bd5010     00:00:1e:63:c8:7c  DR         esi.1816           00:11:11:11:11:11:11:11:11
bd5010     00:00:1e:63:d1:fd  DL         ae1.0              <<< Host on ESI-B learned locally

```

```

jnpr@LEAF-6> show ethernet-switching table vlan-id 20
Vlan name  MAC address      MAC flags  Logical interface  Active source
bd5020     00:00:00:93:3c:f3  DR         esi.1816           00:11:11:11:11:11:11:11:11
bd5020     00:00:00:93:3c:f4  DL         ae1.0              <<< Host on ESI-B learned locally

```

<<< Hosts on ESI-A are learned via esi NH (esi.1816) that resolves to remote VTEPs in POD-1

```

jnpr@LEAF-6> show ethernet-switching vxlan-tunnel-end-point esi
ESI          RTT          VLNBH  INH      ESI-IFL  LOC-IFL  #RVTEPs
00:11:11:11:11:11:11:11:11 default-switch 1816 524295 esi.1816 2
RVTEP-IP     RVTEP-IFL  VENH   MASK-ID  FLAGS
100.0.0.22   vtep.32770 1825   0        2
100.0.0.23   vtep.32772 1821   2        2

```

<<< Host MAC/IP routes for VLANs 10 and 20 learned via Type 2 (MAC rewrite due to asymmetric forwarding)

```

jnpr@LEAF-6> show route forwarding-table destination 100.0.10.100
Routing table: VRF_TENANT_1.inet
Internet:
Destination      Type RtRef Next hop          Type Index  NhRef Netif
100.0.10.100/32  dest   0 45:0:0:32:0:0    ucst   1769    1
!
Interface:      irb.5010
NH Addr: 100.0.10.100
Local Addr: 100.0.10.1
rewrite: 00:00:1e:63:c8:7c

```

Type 5 routes are used to exchange host IP prefixes between PODs that allow for inter-subnet traffic between PODs to be routed on leaf devices that act as distributed Layer 3 gateways. For example, inter-subnet traffic from VLAN 30 (on TORs of POD-1) to VLAN 40 (on TORs of POD-2), is achieved through the exchange of Pure Type 5 routes in the control plane that use an Layer 3 VNI to determine the tenant IP-VRF in which the lookup needs to be done.

```

jnpr@LEAF-6> show route receive-protocol bgp 100.0.0.14 table VRF_TENANT_1.evpn.0 detail
5:100.0.22.10:10::0::100.0.30.100::32/304 (2 entries, 1 announced)
  Import Accepted
  Route Distinguisher: 100.0.22.10:10
  Route Label: 9001
  Overlay gateway address: 0.0.0.0 <<< No overlay NH (Protocol NH used for resolution by remote node)
  Nexthop: 100.0.0.22
  Localpref: 100
  AS path: I (Originator)
  Cluster list: 3.3.3.3 1.1.1.1 2.2.2.2
  Originator ID: 100.0.0.22
  Communities: target:10:10 encapsulation0:0:0:0:vxlan router-
mac:54:4b:8c:cd:b4:38 <<< Chassis MAC advertised using router-
mac extended community used for inner DMAC rewrite

```

```
jnpr@LEAF-6> show evpn ip-prefix-database l3-context VRF_
TENANT_1 direction exported extensive prefix 100.0.40.101
Level 3 context: VRF_TENANT_1
```

IPv4->EVPN Exported Prefixes

```
Prefix: 100.0.40.101/32
  EVPN route status: Created
  Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9001
  Router MAC: ec:3e:f7:84:e3:fa
```

```
jnpr@LEAF-6> show evpn ip-prefix-database l3-context VRF_
TENANT_1 direction imported extensive prefix 100.0.30.100
Level 3 context: VRF_TENANT_1
```

EVPN->IPv4 Imported Prefixes

```
Prefix: 100.0.30.100/32, Ethernet tag: 0
  Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 100.0.22.10:10
VNI: 9001
Router MAC: 54:4b:8c:cd:b4:38
BGP nexthop address: 100.0.0.22
  IP route status: Created
```

Traffic Flows

For this use case, both PODs are situated in the same data center (Intra DC) and all hosts belong to the same tenant (Intra VRF). All hosts share the same IP-VRF (VRF_TENANT_1).

```
CE-1 hosts multi-homed to LEAF-3 and LEAF-4 (ESI-A = 00:11:11:11:11:11:11:11:11:11):
Host 1-1 (H1-1) : VLAN 10 <> VNI 5010, IPv4 = 100.0.10.100, MAC = 00:00:1e:63:c8:7c
Host 1-2 (H1-2) : VLAN 20 <> VNI 5020, IPv4 = 100.0.20.100, MAC = 00:00:00:93:3c:f3
Host 1-3 (H1-3) : VLAN 30 <> VNI 5030, IPv4 = 100.0.30.100, MAC = 00:00:0e:a9:d8:9f
Host 1-4 (H1-4) : VLAN 80 <> VNI 5080, IPv4 = 100.0.80.108, MAC = 00:00:0e:a9:d8:a0
Host 1-5 (H1-5) : VLAN 90 <> VNI 5090, IPv4 = 100.0.90.109, MAC = 00:00:0e:a9:d8:a3
```

```
CE-2 hosts multi-homed to LEAF-5 and LEAF-6 (ESI-B = 00:22:22:22:22:22:22:22:22:22):
Host 2-1 (H2-1) : VLAN 10 <> VNI 5010, IPv4 = 100.0.10.101, MAC = 00:00:1e:63:d1:fd
Host 2-2 (H2-2) : VLAN 20 <> VNI 5020, IPv4 = 100.0.20.101, MAC = 00:00:00:93:3c:f4
Host 2-3 (H2-3) : VLAN 40 <> VNI 5040, IPv4 = 100.0.40.101, MAC = 00:00:4a:89:03:14
Host 2-4 (H2-4) : VLAN 85 <> VNI 5085, IPv4 = 100.0.85.108, MAC = 00:00:0e:a9:d8:a1
```

```
Default gateway : IPv4 = 100.0.10.1
                  (Static IP for irb.5010 Layer 3 interface for VLAN 10)
                  : IPv4 = 100.0.20.1
  (Static IP for irb.5020 Layer 3 interface for VLAN 20)
                  : IPv4 = 100.0.30.1
  (Static IP for irb.5030 Layer 3 interface for VLAN 30)
                  : IPv4 = 100.0.40.1
  (Static IP for irb.5040 Layer 3 interface for VLAN 40)
                  : IPv4 = 100.0.80.1
  (Static IP for irb.5080 Layer 3 interface for VLAN 80)
                  : IPv4 = 100.0.85.1
  (Static IP for irb.5085 Layer 3 interface for VLAN 85)
                  : IPv4 = 100.0.90.1
  (Static IP for irb.5090 Layer 3 interface for VLAN 90)
```

Flows demonstrated here are outlined below and illustrated in Figure 2.8.

Inter-POD, Inter-Rack, Intra-Subnet/Layer 2 (ESI A <> ESI B):

- VLAN 10 <> 10: IPv4 100.0.10.100 <> 100.0.10.101
- VLAN 20 <> 20: IPv4 100.0.20.100 <> 100.0.20.101

Inter-POD, Inter-Rack, Inter-Subnet/Layer 3 (ESI A <> ESI B):

- VLAN 30 <> 40: IPv4 100.0.30.100 <> 100.0.40.101
- VLAN 90 <> 85: IPv4 100.0.90.109 <> 100.0.85.108

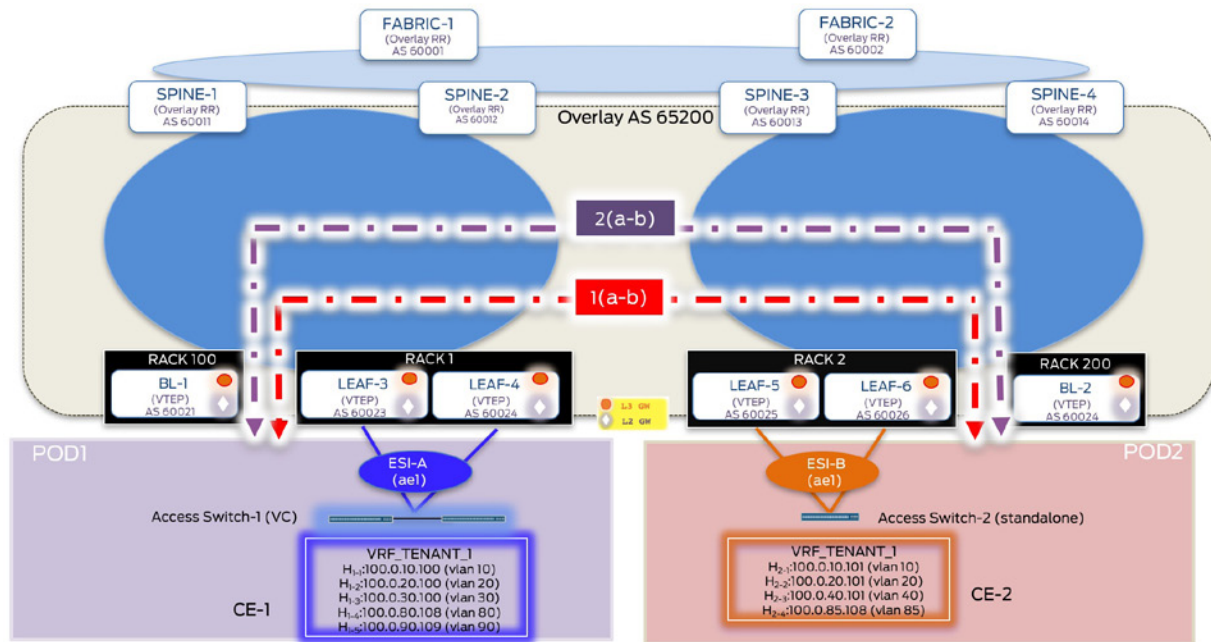


Figure 2.8 Deployment Use Case – Traffic Flows

IXIA Statistics – Intra DC Intra-VRF (Flows 1(a-b), 2(a-b))

(1000 flows @ 1000 pps)

Transmit State	Traffic Item Name	Enabled	Flow Groups	Tx Ports	Rx Ports	Endpoint/Encapsulation Sets
1	1a-EW Intra VRF Intra VNI: vlan 10 <> 10: 100.0.10.100 <> 100.0.10.101 (A-B)	✓	2	2	2	1
2	1b-EW Intra VRF Intra VNI: vlan 20 <> 20: 100.0.20.100 <> 100.0.20.101 (A-B)	✓	2	2	2	1
3	2a-EW Intra VRF Inter VNI: vlan 30 <> 40: 100.0.30.100 <> 100.0.40.101 (A-B)	✓	2	2	2	1
4	2b-EW Intra VRF Inter VNI: vlan 90 <> 85: 100.0.90.109 <> 100.0.85.108 (A-B)	✓	2	2	2	1

Select Views...		Port CPU Statistics	Port Statistics	Global Protocol Statistics	BGP Aggregated Statistics	L2-L3 Test Summary Statistics	Flow Statistics	Data Plane Port Statistics	User Defined Statistics	Traffic Item
Tx Port	Rx Port	Traffic Item	Source/Dest Endpoint Pair	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate	
1/1 <> VC	1/2 <> TOR	1a-EW Intra VRF Intra VNI: vlan 10 <> 10: 100.0.10.100 <> 100.0.10.101 (A-B)	100.0.10.100-100.0.10.101	653,776	653,776	0	0.000	1,000,000	1,000,000	
2/1 <> TOR	1/1 <> VC	1a-EW Intra VRF Intra VNI: vlan 10 <> 10: 100.0.10.100 <> 100.0.10.101 (A-B)	100.0.10.101-100.0.10.100	653,776	653,776	0	0.000	1,000,000	1,000,000	
3/1 <> VC	1/2 <> TOR	1b-EW Intra VRF Intra VNI: vlan 20 <> 20: 100.0.20.100 <> 100.0.20.101 (A-B)	100.0.20.100-100.0.20.101	651,455	651,455	0	0.000	1,000,000	1,000,000	
4/1 <> TOR	1/1 <> VC	1b-EW Intra VRF Intra VNI: vlan 20 <> 20: 100.0.20.100 <> 100.0.20.101 (A-B)	100.0.20.101-100.0.20.100	651,439	651,439	0	0.000	1,000,000	1,000,000	
5/1 <> VC	1/2 <> TOR	2a-EW Intra VRF Inter VNI: vlan 30 <> 40: 100.0.30.100 <> 100.0.40.101 (A-B)	100.0.30.100-100.0.40.101	650,402	650,402	0	0.000	1,000,000	1,000,000	
6/1 <> TOR	1/1 <> VC	2a-EW Intra VRF Inter VNI: vlan 30 <> 40: 100.0.30.100 <> 100.0.40.101 (A-B)	100.0.40.101-100.0.30.100	650,392	650,392	0	0.000	1,000,000	1,000,000	
7/1 <> VC	1/2 <> TOR	2b-EW Intra VRF Inter VNI: vlan 90 <> 85: 100.0.90.109 <> 100.0.85.108 (A-B)	100.0.90.109-100.0.85.108	648,810	648,810	0	0.000	1,000,000	1,000,000	
8/1 <> TOR	1/1 <> VC	2b-EW Intra VRF Inter VNI: vlan 90 <> 85: 100.0.90.109 <> 100.0.85.108 (A-B)	100.0.85.108-100.0.90.109	648,798	648,798	0	0.000	1,000,000	1,000,000	

Manual Service Chaining of East-West Traffic

Layer 3 Security Insertion: Inter-tenant Inter-subnet

For traffic between hosts in different tenants on separate PODs, manual service chaining is demonstrated by hair pinning traffic through the local DC firewall for policy enforcement before forwarding it to the intended destination. In this use case, the DC firewall (SRX) is connected to each border leaf device. As elaborated previously, service chaining is treated as an IP path problem by getting traffic to the firewall. Details on firewall rules and policy enforcement are outside the scope of this book.

Service node: Refers to the border-leaf devices that service each POD by directing intended traffic from the IP fabric to the service block (firewall).

Service block: Refers to the firewall that is responsible for service chaining inter-tenant traffic.

The following sub-sections elaborate upon the control plane and data plane details used to achieve the Layer 3 security insertion for EW inter-tenant traffic:

Configuration Layout

All configuration details as explained in the previous sections remain as is. Figure 2.9 highlights the logical topology on how the firewall is inserted in the DC to process traffic between different tenants.

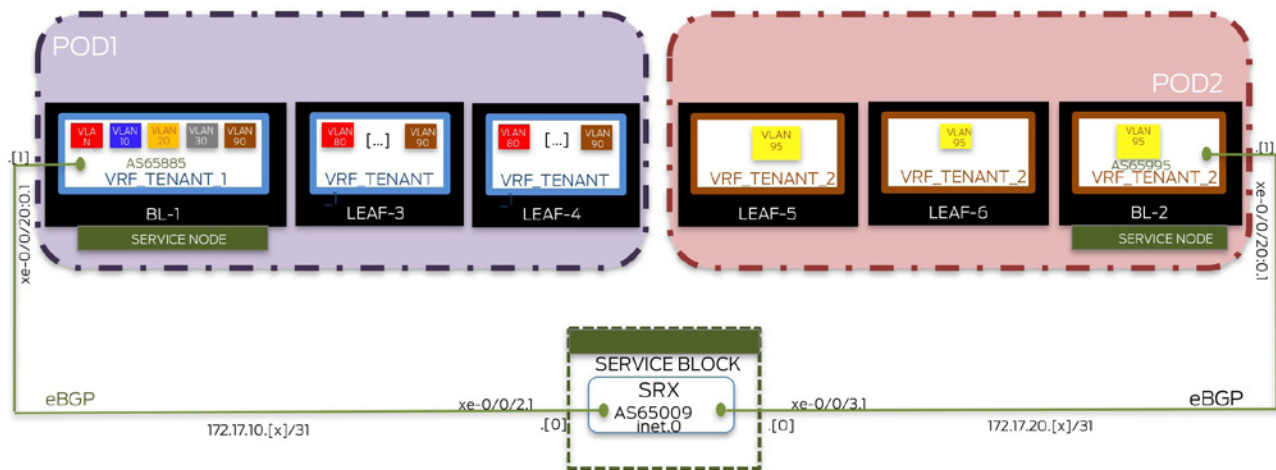


Figure 2.9 Layer 3 Security Insertion – Logical topology

One physical link each exists between SRX and service nodes, BL-1 and BL-2, respectively.

One IFL (logical interface) for each connecting physical link terminates in the IP-VRF, routes for which need to be exchanged with the firewall. For example, inter-subnet traffic between hosts in tenant-1(POD1) and tenant-2(POD2) needs to be hair-pinned through the firewall. To accomplish this, one connecting IFL terminates in tenant-1 IP-VRF (VRF_TENANT_1.inet.0) and the other in tenant-2 IP-VRF (VRF_TENANT_2.inet.0).

An eBGP session exists over this connecting IFL in the IP-VRF on the border-leaf device (VRF_TENANT_1.inet.0 on BL-1) and (VRF_TENANT_2.inet.0 on BL-2) and global table (inet.0) on the SRX. Routes for all contained VLANs can be exchanged over this single eBGP session per VRF and no additional provisioning per VLAN is required.

Route Exchange Between Service Nodes and Service Block

Figure 2.10 highlights the exchange of host routes between devices in the IP fabric and service block, focusing here on VLAN 80 (VRF_TENANT_1 in POD1) and VLAN 95 (VRF_TENANT_2 in POD2). This allows the firewall to know which service node (BL-1 or BL-2) to forward traffic to for a given host.

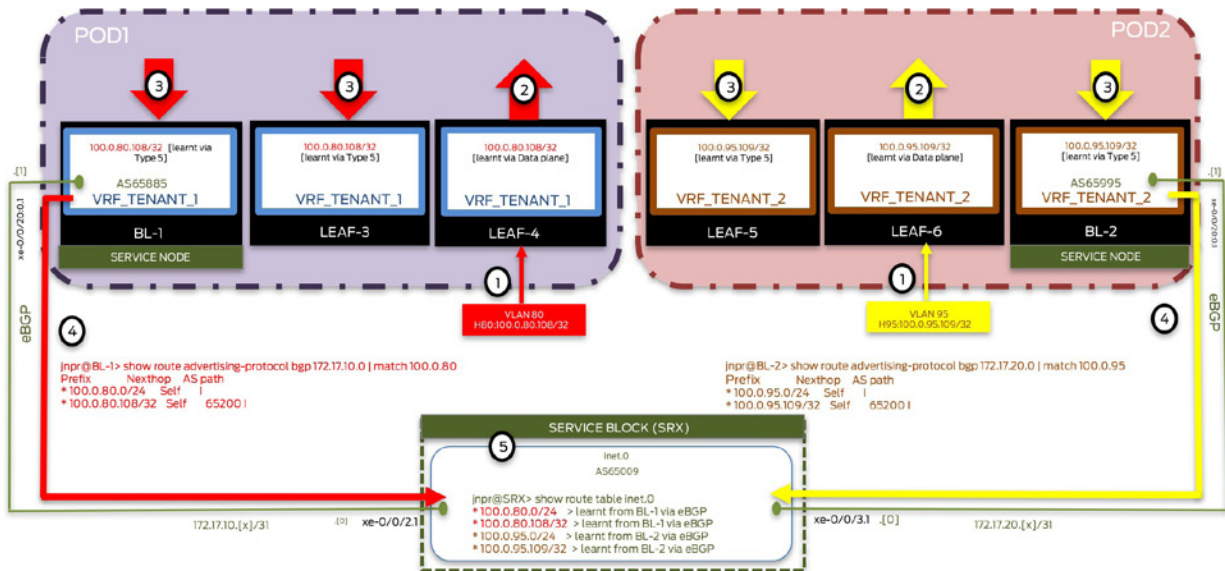


Figure 2.10 Exchange of Host Routes Between Devices in the IP Fabric and Service Block

1. Layer 3 gateways learn host routes for connected hosts through data plane learning. Shown above, ARP request for host H80 in VLAN 80, simulated by IXIA connected to Access-Switch-1, hashes on the physical link connected to LEAF-4. As a result, host route (100.0.80.108/32) is installed in the IP-VRF (VRF_TENANT_1) of the intercepting Layer 3 gateway (LEAF-4) in POD1. Similarly, host route for H90 in VLAN 90 is installed in IP-VRF (VRF_TENANT_2) of the intercepting Layer 3 gateway (LEAF-6) in POD2.
2. On each leaf and border leaf device, routes in the IP-VRF are advertised into the overlay as EVPN Type 5 routes by means of the configured export policy. Shown above, Layer 3 gateways (LEAF-4 in POD1 and LEAF-6 in POD2) advertise host IP prefixes (H80 – 100.0.80.108 in POD1 and H95 – 100.0.95.109 in POD2) as EVPN Type 5 routes to other devices in the fabric.
3. EVPN PEs supporting relevant tenants will import these Type 5 routes received from the respective POD route-reflectors. EVPN Type 5 routes advertised for a given tenant are imported by all Layer 3 gateways that have IP-VRFs configured for the same tenant, by virtue of the IP-VRF route target. Shown above, host IP prefix for H80 (100.0.80.108/32) received as EVPN Type 5 route, is installed in the IP-VRF (VRF_TENANT_1) on LEAF-3 and BL-1 in POD1. Similarly, host IP prefix for H95 (100.0.95.109/32) is installed in the IP-VRF (VRF_TENANT_2) on LEAF-5 and BL-2 in POD2.
4. Service nodes for each POD, advertise host IP prefixes over the eBGP session to the connecting service block by means of the configured export policy. Shown above, BL-1 advertises the received host IP route (H80 – 100.0.80.108 in POD1) while BL-2 advertises the received host IP route (H95 – 100.0.95.109 in POD2) to the connecting service block.

(firewall). Though not required, summary route advertisement has also been demonstrated, in case specific host location awareness is not needed.

5. Received host prefixes over the eBGP session are installed in the global table (inet.0) on the firewall. As can be seen here, the firewall will direct traffic to BL-1 (service node in POD1) if the destination is a host in POD1 (e.g. H80), or to BL-2 (service node in POD2) if the destination is a host in POD2 (e.g. H95). Configuration on the firewall is beyond the scope of this book, but it can be further optimized for specific policy enforcement.

Figure 2.11 highlights the exchange of default route service block and devices in the IP fabric. This allows for the service node (BL-1 or BL-2) to direct inter-tenant, for the servicing POD traffic, to the firewall.

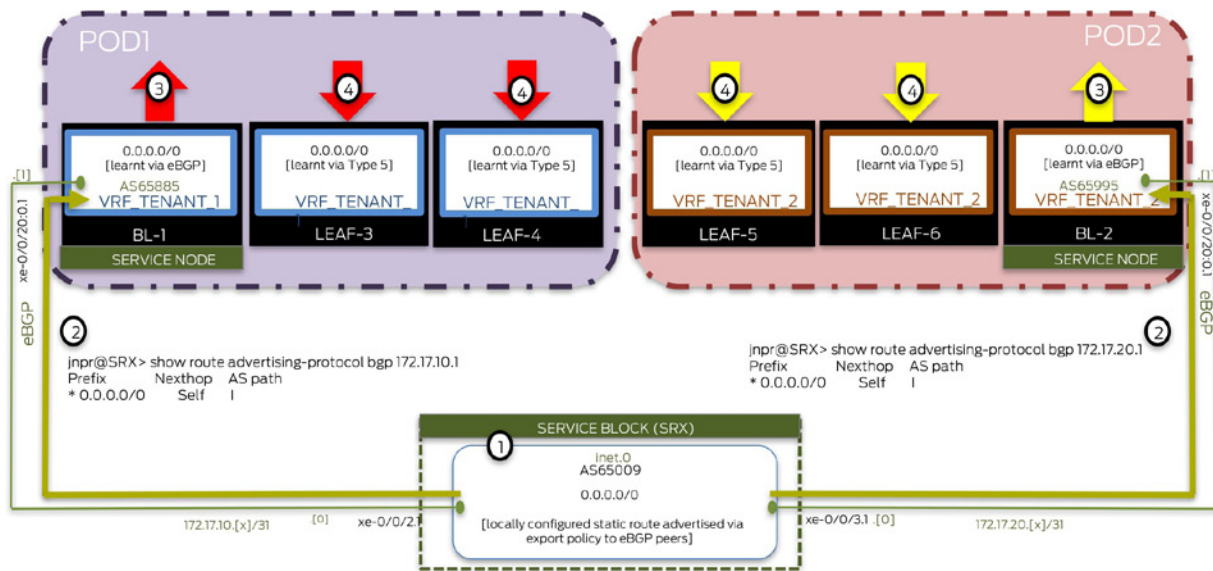


Figure 2.11 Exchange of Default Route Service Block and Devices in the IP Fabric

1. Static default route has been configured on the firewall.
2. This is advertised over the eBGP sessions to each connecting service node (BL-1 in POD1 and BL-2 in POD2).
3. This default route is further advertised by the service node in each POD to other devices of the same POD as an EVPN Type 5 route by means of the configured export policy.
4. Here, each Layer 3 gateway in a given POD will use the default route only from the service node of the respective POD. As a result, when a tenant 1 host of VLAN 80 (VRF_TENANT_1) in POD1 intends to communicate with a tenant 2 host of VLAN 95 (VRF_TENANT_2) in POD2 destination route lookup matches the default route originally advertised by the SRX. As a result, traffic is punted from the intercepting Layer 3 gateways devices (LEAF-3,4) to service nodes of the residing POD (BL-1 in POD1). BL-1 further directs this traffic to the connecting SRX that has specific route information to reach the desired destination in POD2. The same behavior is observed when hosts in POD2 (VRF_TENANT_2) need to communicate with hosts in POD1 (VRF_TENANT_1). Though only traffic for a single VLAN in each tenant has been demonstrated, only VLAN provisioning to accommodate increased scale in residing tenants is required and no additional configuration on the firewall is needed.

Configuration

As explained in the previous sections, LEAF-3, LEAF-4, and BL-1 in POD1 have been configured to accommodate hosts that belong to VRF_TENANT_1, while LEAF-5, LEAF-6, and BL-2 in POD2 have been configured to accommodate hosts that belong to VRF_TENANT_1 and VRF_TENANT_2. All configuration from previous sections remains unchanged, including that on spine and fabric devices. For brevity, only additional configuration on leaf, border-leaf, and firewall devices to support manual service chaining of inter-tenant traffic (VRF_TENANT_1 in POD1 and VRF_TENANT_2 in POD2) have been highlighted here.

Leaf Devices

Shown below is related configuration for LEAF-4 (LEAF-3 has been provisioned in a similar manner):

```
jnpr@LEAF-4> show configuration routing-instances
VRF_TENANT_1 {
    instance-type vrf;
    interface irb.5010;
    interface irb.5020;
    interface irb.5030;
    interface irb.5080;
    interface irb.5090;
    interface lo0.10;
    route-distinguisher 100.0.23.10:10;
    vrf-import vrf-import-tenant1;
    vrf-target target:10:10;
    vrf-table-label;
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop;
                encapsulation vxlan;
                vni 9001;
                export EVPN-host-routes;
            }
        }
    }
}

jnpr@LEAF-4> show configuration policy-options policy-statement vrf-import-tenant1
term rej-remotePOD-srxrts { <<< Import policy prefers POD1 service node 0/0 rt
    from {
        next-hop 100.0.0.24;
        as-path orig-in-SRX;
    }
    then reject;
}
term accept-tenant1-rts {
    from community tenant1-rt;
    then accept;
}
as-path orig-in-SRX ".* 65009";
community tenant1-rt members target:10:10;

jnpr@LEAF-4> show configuration policy-options policy-statement EVPN-host-routes
term 1 { <<< Export policy to advertise host IP prefixes as Type 5 routes
    from {
        protocol evpn;
        route-filter 0.0.0.0/0 prefix-length-range /32-/32;
    }
    then accept;
}
```

Shown below is related configuration for LEAF-6 (LEAF-5 has been provisioned in a similar manner):

```
jnpr@LEAF-6> show configuration routing-instances
VRF_TENANT_1 { ... }
VRF_TENANT_2 {
    instance-type vrf;
    interface irb.5095;
    interface lo0.20;
    route-distinguisher 100.0.26.20:20;
    vrf-import vrf-import-tenant2;
    vrf-target target:20:20;
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop;
                encapsulation vxlan;
                vni 9002;
                export EVPN-host-routes;
            }
        }
    }
}

jnpr@LEAF-6> show configuration policy-options policy-statement vrf-import-tenant2
term rej-remotePOD-srxrts { <<< Import policy prefers POD2 service node 0/0 rt
    from {
        next-hop 100.0.0.21;
        as-path orig-in-SRX;
    }
    then reject;
}
term accept-tenant2-rts {
    from community tenant2-rt;
    then accept;
}

jnpr@LEAF-6> show configuration policy-options policy-statement EVPN-host-routes
term 1 { <<< Export policy to advertise host IP prefixes as Type 5 routes
    from {
        protocol evpn;
        route-filter 0.0.0.0/0 prefix-length-range /32-/32;
    }
    then accept;
}
```

Border Leaf Devices

```
jnpr@BL-1> show configuration routing-instances
VRF_TENANT_1 {
    instance-type vrf;
    interface xe-0/0/20:0.1; <<< IFL on connecting link to SRX in IP-VRF
    interface irb.5010;
    interface irb.5020;
    interface irb.5030;
    interface irb.5080;
    interface irb.5090;
    interface lo0.10;
    route-distinguisher 100.0.21.10:10;
    vrf-import vrf-import-tenant1;
    vrf-export vrf-export-tenant1;
    vrf-target target:10:10;
    protocols {
        bgp {
            group viaSRX-ext { <<< eBGP session created between SRX and tenant VRF
                type external;
                export adv-tenant-rts;
                local-as 65885;
            }
        }
    }
}
```

```

        multipath multiple-as;
        neighbor 172.17.10.0 {
            description "eBGP: BL-1 <> SRX";
            peer-as 65009;
        }
    }
}
evpn {
    ip-prefix-routes {
        advertise direct-nexthop;
        encapsulation vxlan;
        vni 9001;
        export EVPN-host-routes;
    }
}
}
}

jnpr@BL-1> show configuration policy-options policy-statement adv-tenant-rtts
term rej-remote-POD-hostrts {<<< Export policy to advertise POD1 routes to SRX
    from {
        next-hop [ 100.0.0.24 100.0.0.25 100.0.0.26 ];
    }
    then reject;
}
term local-POD-hostrts {
    from protocol evpn;
    then accept;
}
term local-POD-summary {
    from {
        protocol direct;
        route-filter 0.0.0.0/0 prefix-length-range /24-/24;
    }
    then accept;
}

jnpr@BL-1> show configuration policy-options policy-statement EVPN-host-routes
term 1 {
    from {
        protocol evpn;
        route-filter 0.0.0.0/0 prefix-length-range /32-/32;
    }
    then accept;
}
term 2 { <<< Export policy to advertise received SRX default route to IP fabric
    from {
        protocol bgp;
        as-path orig-in-SRX;
    }
    then accept;
}

jnpr@BL-2> show configuration routing-instances
VRF_TENANT_2 {
    instance-type vrf;
    interface xe-0/0/20:0.1; <<< IFL on connecting link to SRX in IP-VRF
    interface irb.5095;
    interface lo0.20;
    route-distinguisher 100.0.24.20:20;
    vrf-target target:20:20;
    protocols {
        bgp {
            group viaSRX-ext { <<< eBGP session created between SRX and tenant VRF

                type external;
                export adv-tenant-rtts;
                local-as 65995;
                multipath multiple-as;
                neighbor 172.17.20.0 {

```

```

        description "eBGP: BL-2 <> SRX";
        peer-as 65009;
    }
}
}
evpn {
    ip-prefix-routes {
        advertise direct-nexthop;
        encapsulation vxlan;
        vni 9002;
        export EVPN-host-routes;
    }
}
}

jnpr@BL-2> show configuration policy-options policy-statement adv-tenant-rtts
term rej-remote-POD-hostrts {    <<< Export policy to advertise POD2 routes to SRX
    from next-hop [ 100.0.0.21 100.0.0.22 100.0.0.23 ];
    then reject;
}
term local-POD-hostrts {
    from protocol evpn;
    then accept;
}
term local-POD-summary {
    from {
        protocol direct;
        route-filter 0.0.0.0/0 prefix-length-range /24-/24;
    }
    then accept;
}

jnpr@BL-2> show configuration policy-options policy-statement EVPN-host-routes
term 1 {
    from {
        protocol evpn;
        route-filter 0.0.0.0/0 prefix-length-range /32-/32;
    }
    then accept;
}
term 2 { <<< Export policy to advertise received SRX default route to IP fabric
    from {
        protocol bgp;
        as-path orig-in-SRX;
    }
    then accept;
}

```

SRX

EBGP sessions are created from the SRX (inet.0) and each is connected a border-leaf device (tenant IP-VRF):

```

jnpr@SRX> show configuration protocols bgp
group viaSRX-ext {
    type external;
    export adv-default;
    local-as 65009;
    multipath multiple-as;
    neighbor 172.17.10.1 {                <<< eBGP session with BL-1
        description "to BL-1 (VRF_TENANT_1)";
        peer-as 65885;
    }
    neighbor 172.17.20.1 {                <<< eBGP session with BL-2
        description "to BL-2 (VRF_TENANT_2)";
        peer-as 65995;
    }
    neighbor 182.18.0.2 {

```

```

        description "to BL-2 (VRF_TENANT_1)";
        peer-as 65995;
    }
}

jnpr@SRX> show configuration routing-options static
    route 0.0.0.0/0 reject;

jnpr@SRX> show configuration policy-options policy-statement adv-default
term static-rt {
    from {
        protocol static;          <<< SRX advertises 0/0 to all border-leaf devices
        route-filter 0.0.0.0/0 exact;
    }
    then accept;
}
term default {
    then reject;
}

```

Verification

SRX receives VLAN 80 summary and host route from BL-1 and VLAN 95 summary and host route from BL-2.

```

jnpr@SRX> show route 100.0.80/24

inet.0: 35 destinations, 39 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100.0.80.0/24      *[BGP/170] 2d 08:27:21, localpref 100
                  AS path: 65885 I, validation-state: unverified
                  > to 172.17.10.1 via xe-0/0/2.1
100.0.80.108/32   *[BGP/170] 1d 17:14:37, localpref 100
                  AS path: 65885 65200 I, validation-state: unverified
                  > to 172.17.10.1 via xe-0/0/2.1

jnpr@SRX> show route 100.0.95/24

inet.0: 35 destinations, 39 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100.0.95.0/24     *[BGP/170] 2d 06:53:36, localpref 100
                  AS path: 65995 I, validation-state: unverified
                  > to 172.17.20.1 via xe-0/0/3.1
100.0.95.109/32   *[BGP/170] 1d 18:13:14, localpref 100
                  AS path: 65995 65200 I, validation-state: unverified
                  > to 172.17.20.1 via xe-0/0/3.1

```

BL-1 and BL-2 receive the default route from the SRX in the connecting tenant IP-VRF, which is further advertised to all devices in the IP fabric.

```

jnpr@BL-1> show route table VRF_TENANT_1.inet.0

VRF_TENANT_1.inet.0: 19 destinations, 19 routes (19 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0         *[BGP/170] 1w6d 15:51:16, localpref 100
                  AS path: 65009 I, validation-state: unverified
                  > to 172.17.10.0 via xe-0/0/20.0.1

```

```

jnpr@BL-1> show evpn ip-prefix-database extensive
Level 3 context: VRF_TENANT_1

```

IPv4->EVPN Exported Prefixes

```

Prefix: 0.0.0.0/0
EVPN route status: Created
Change flags: 0x0

```

```

Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9001
Router MAC: 54:4b:8c:62:cb:f6

```

```

jnpr@BL-2> show route advertising-protocol bgp 100.0.0.13

```

```

VRF_TENANT_2.evpn.0: 2 destinations, 3 routes (2 active, 0 holddown, 0 hidden)
Prefix                    Nexthop      MED      Lclpref  AS path
5:100.0.24.20:20::0::0.0.0.0::0/304  Self        100      65995 65009 I

```

Traffic Flows

Figure 2.12 depicts the path traversed by traffic between host H80 (VRF_TENANT_1: IP 100.0.80.108) and host H95 (VRF_TENANT_2: IP 100.0.95.109) hair-pinning through the DC firewall (SRX).

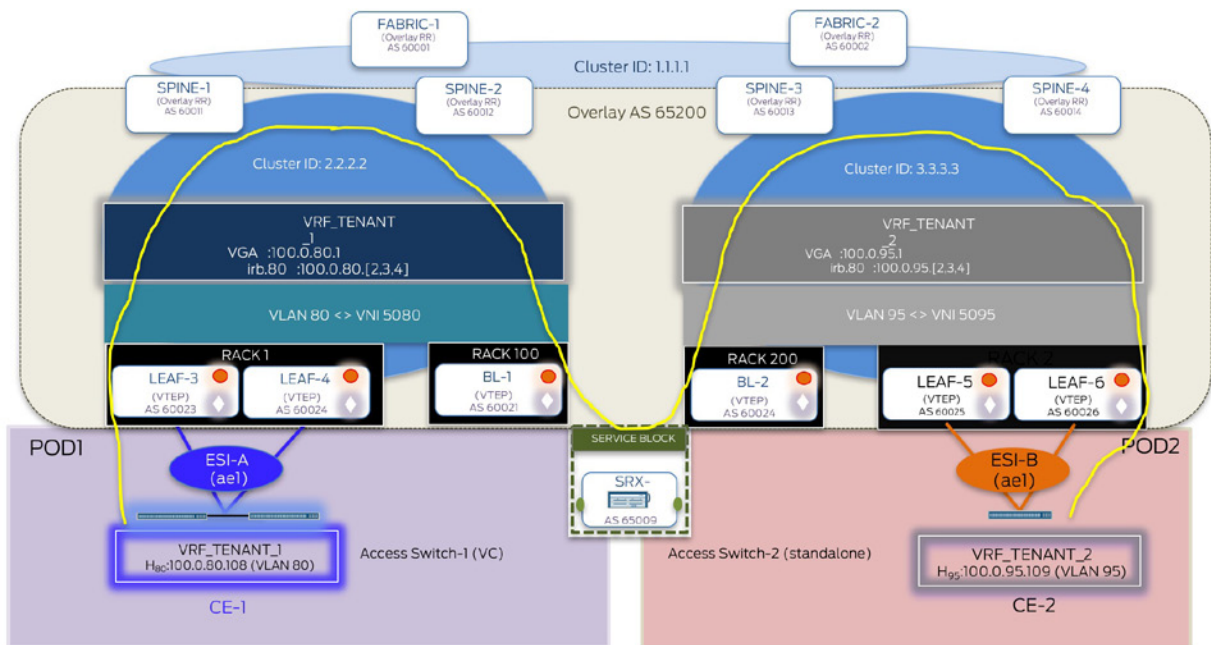


Figure 2.12 Manual Service Chaining for Inter-PODEW – Inter-VRF inter Subnet Traffic (H80 > H95)

Inter-POD traffic from host in POD2 to POD1 is hair-pinned only through the firewall.

```

H80 > H95:
SRX
Interface  Link  Input packets  Seconds: 430  Output packets  Time: 01:36:09
xe-0/0/2   Up    1182763726      (pps)         1063826109    (pps)
xe-0/0/3   Up    786587036       (1002)        901306152     (0) <<< From BL-1
                                     10000 >>> To BL-2

```

Hair-pinning through the SRX is observed for bi-directional traffic.

```

H80 <> H95:
jnpr@SRX> show security flow session
Session ID: 282557, Policy name: 1/6, Timeout: 60, Valid
In: 100.0.95.109/1 --> 100.0.80.108/1;udp, Conn Tag: 0x0, If: xe-0/0/3.1, Pkts: 7, Bytes: 10346,
Out: 100.0.80.108/1 --> 100.0.95.109/1;udp, Conn Tag: 0x0, If: xe-0/0/2.1, Pkts: 6, Bytes: 8868,

```



```

Session ID: 282559, Policy name: 1/6, Timeout: 60, Valid
  In: 100.0.95.109/2 --> 100.0.80.108/2;udp, Conn Tag: 0x0, If: xe-0/0/3.1, Pkts: 7, Bytes: 10346,
  Out: 100.0.80.108/2 --> 100.0.95.109/2;udp, Conn Tag: 0x0, If: xe-0/0/2.1, Pkts: 6, Bytes: 8868,
<...>
Session ID: 282587, Policy name: 1/6, Timeout: 60, Valid
  In: 100.0.80.108/16 --> 100.0.95.109/16;udp, Conn Tag: 0x0, If: xe-0/0/2.1, Pkts: 7, Bytes: 10346,
  Out: 100.0.95.109/16 --> 100.0.80.108/16;udp, Conn Tag: 0x0, If: xe-0/0/3.1, Pkts: 6, Bytes: 8868,
Session ID: 282589, Policy name: 1/6, Timeout: 60, Valid
  In: 100.0.80.108/17 --> 100.0.95.109/17;udp, Conn Tag: 0x0, If: xe-0/0/2.1, Pkts: 7, Bytes: 10346,
  Out: 100.0.95.109/17 --> 100.0.80.108/17;udp, Conn Tag: 0x0, If: xe-0/0/3.1, Pkts: 6, Bytes: 8868,

```

IXIA Statistics - Intra DC Inter-POD (H80 <> H95: 100.0.80.108 <> 100.0.95.109)

(1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %
3a-EW Inter VRF Inter VNI: vlan 80 <> 95: 100.0.80.108 <> 100.0.95.109 (A-B)	15,654	15,622	32	0

Manual Service Chaining of North-South Traffic

Layer 3 Security Insertion: Inter-tenant Inter-subnet

For NS traffic between hosts in different tenants, manual service chaining is demonstrated by hair pinning traffic through the local DC firewall for policy enforcement before forwarding it to the intended destination. DC firewall (SRX) connected to each border-leaf device is used to process EW and NS traffic. As elaborated previously, service chaining is treated as an IP path problem by getting traffic to the firewall. Details on firewall rules and policy enforcement are outside the scope of this book.

Service node – Refers to the border-leaf devices that service each POD by directing intended traffic from the IP fabric to the service block (firewall). These border-leaf devices are also acting as DC-edge that connect the data center to the WAN.

Service block – Refers to the firewall that is responsible for service chaining inter-tenant traffic (EW and NS).

The following sub-sections elaborate upon the control plane and data plane details used to achieve the Layer 3 security insertion for NS inter-tenant traffic:

Configuration Layout

All configuration details explained in the previous sections remain as is. Figure 2.13 highlights the logical topology on how the firewall is inserted in the DC to process NS traffic between different tenants:

An eBGP session exists over this connecting IFL in the IP-VRF on each border leaf device (VRF_TENANT_NS.inet.0 on BL-1) and (VRF_TENANT_2.inet.0 on BL-2) and global table (inet.0) on the SRX, respectively. Routes for all hosts across the WAN can be exchanged over this single eBGP session per VRF and no additional provisioning per host/VLAN is required.

Route Exchange Between Service Nodes and Service Block

This section focuses on demonstrating hair-pinning inter-subnet traffic across the WAN and between tenant-1 hosts in VLAN 10 (on POD1 and POD2) and VLAN 90 (on POD1 only).

Exchange of host routes between devices in the IP-fabric and service block for data center hosts is the same as discussed in the previous sections and will not be repeated here. Figure 2.14 summarizes the exchange of routes for data center hosts and north-bound hosts across the WAN with the local firewall.

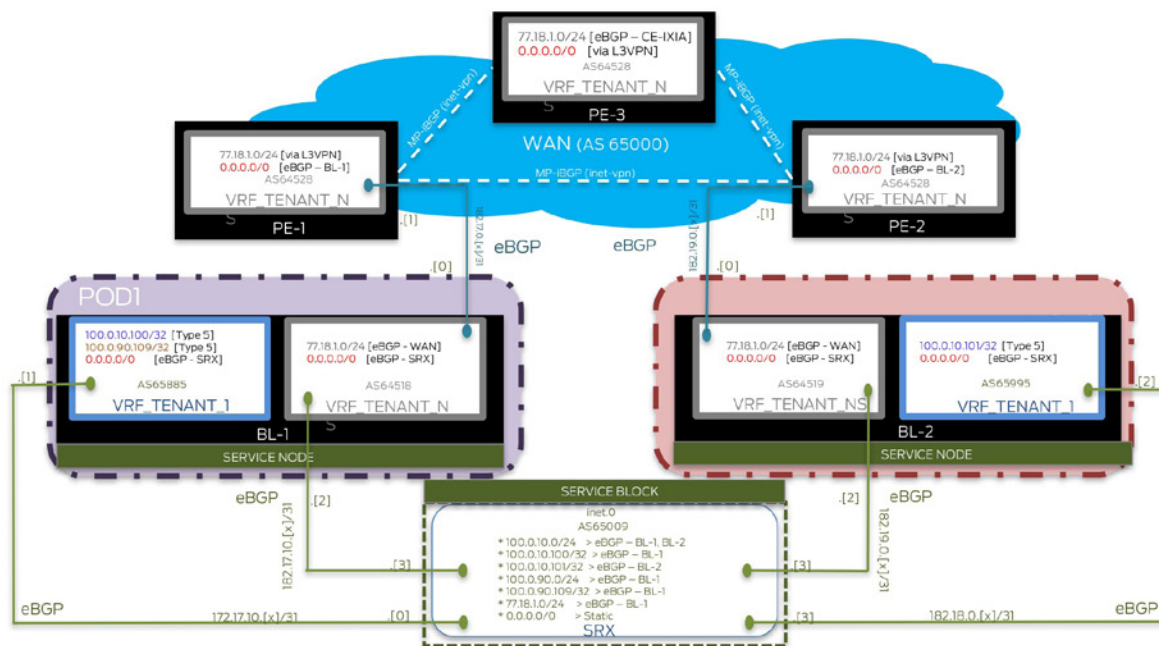


Figure 2.14 Exchange of Routes for DC Hosts

Configuration from the previous sections remains unchanged.

To hair-pin NS traffic through the SRX, each service node creates another eBGP session for NS IP-VRF (VRF_TENANT_NS) and the DC firewall, to advertise routes learned from the WAN directly to the firewall. The default route received from the SRX is further advertised towards the WAN.

As a result, when a host across the WAN (77.18.1.1) wants to send traffic to a host in tenant-1, traffic is directed from the WAN PEs to the service nodes (BL-1 and BL-2). Since the destination lookup for a tenant-1 host is done in the NS IP-VRF (VRF_TENANT_NS.inet.0), the default route is the best match.

The service nodes (BL-1 and BL-2) further direct this traffic towards the firewall.

Since the firewall has specific route information for data center hosts due to eBGP sessions terminating in the tenant IP-VRF, exact host location is known and traffic can be forwarded optimally.

Configuration

All configuration from the previous sections remains unchanged including that on leaf, spine and fabric devices. For brevity, only additional configuration on WAN, border-leaf, and fire-wall devices to support manual service chaining of NS inter-tenant traffic have been highlighted here.

WAN Devices

Shown below is configuration for PE3 and PE-1 (PE-2 has been configured similarly):

```
jnpr@PE-3> show configuration protocols
bgp {
    group Level 3VPN-PEs {
        type internal;
        local-address 101.0.0.3;
        family inet-vpn {
            any;
        }
        local-as 65000;
        bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode automatic;
        }
        multipath;
        neighbor 101.0.0.1;      <<< MP-iBGP (inet-vpn) full-mesh between WAN PEs
        neighbor 101.0.0.2;
    }
}
ospf {
    area 0.0.0.0 {
        interface xe-2/1/2.0;
        interface xe-2/1/3.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface xe-2/1/2.0;
    interface xe-2/1/3.0;
    interface lo0.0;
}

jnpr@PE-3> show configuration routing-instances
VRF_TENANT_NS {
    instance-type vrf;
    interface xe-2/2/0.0;
    route-distinguisher 101.0.3.10:10;
    vrf-target target:18:18;
    routing-options {
        multipath;
    }
    protocols {
        bgp {
            group CE_Level 3VPN {
                type external;
                family inet {
                    unicast;
                }
                peer-as 65061;
                local-as 65060;
                neighbor 60.60.60.1; <<< eBGP between PE-3 and connecting CE
            }
        }
    }
}
```

```

jnpr@PE-1> show configuration protocols
bgp {
    group Level 3VPN-PEs {
        type internal;
        local-address 101.0.0.1;
        family inet-vpn {
            any;
        }
        local-as 65000;
        bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode automatic;
        }
        multipath;
        neighbor 101.0.0.2;
        neighbor 101.0.0.3;
    }
}
ospf {
    area 0.0.0.0 {
        interface xe-2/1/2.0;
        interface xe-2/1/3.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface xe-2/1/2.0;
    interface xe-2/1/3.0;
    interface lo0.0;
}
jnpr@PE-1> show configuration routing-instances
VRF_TENANT_NS {
    instance-type vrf;
    interface xe-2/1/1.0;
    route-distinguisher 101.0.1.10:10;
    vrf-target target:18:18;
    protocols {
        bgp {
            group via-WAN { <<< eBGP between PE-1 and connecting BL-1 ifl
                type external;
                local-as 64528;
                neighbor 182.17.0.0 {
                    description "eBGP peering with BL-1";
                    peer-as 64518;
                }
            }
        }
    }
}

```

Border Leaf Devices

```

jnpr@BL-1> show configuration routing-instances VRF_TENANT_NS | except #
instance-type vrf;
interface xe-0/0/19:0.0; <<< IFL on connecting link to PE-1
interface xe-0/0/20:0.2; <<< IFL on connecting link to SRX
route-distinguisher 100.0.21.18:18;
vrf-target target:18:18;
protocols {
    bgp {
        group via-WAN {
            type external;
            local-as 64518;
            neighbor 182.17.0.1 { <<< eBGP peering with PE-1 on connecting IFL
                description "eBGP peering with WAN - PE1";
                peer-as 64528;
            }
        }
    }
}

```

```

        neighbor 182.17.0.3 { <<< eBGP peering with SRX on connecting IFL
            description "eBGP peering with SRX";
            peer-as 65009;
        }
    }
}

jnpr@BL-2> show configuration routing-instances VRF_TENANT_NS | except #
instance-type vrf;
interface xe-0/0/19:0.0; <<< IFL on connecting link to PE-2
interface xe-0/0/20:0.3; <<< IFL on connecting link to SRX
route-distinguisher 100.0.24.18:18;
vrf-target target:18:18;
protocols {
    bgp {
        group via-WAN {
            type external;
            local-as 64519;
            neighbor 182.18.0.1 { <<< eBGP peering with PE-2 on connecting IFL
                description "eBGP peering with WAN - PE2";
                peer-as 64529;
            }
            neighbor 182.19.0.3 { <<< eBGP peering with SRX on connecting IFL
                description "eBGP peering with SRX";
                peer-as 65009;
            }
        }
    }
}

```

SRX

EBGP sessions are created from the SRX (inet.0) and each connected border-leaf device (tenant IP-VRF and NS IP-VRF):

```

jnpr@SRX> show configuration protocols bgp
group viaSRX-ext {
    type external;
    export adv-default;
    local-as 65009;
    multipath multiple-as;
    neighbor 172.17.10.1 { <<< eBGP session with BL-1 (VRF_TENANT_1)
        description "to BL-1 (VRF_TENANT_1)";
        peer-as 65885;
    }
    neighbor 172.17.20.1 {
        description "to BL-2 (VRF_TENANT_2)";
        peer-as 65995;
    }
    neighbor 182.17.0.2 { <<< eBGP session with BL-1 (VRF_TENANT_NS)
        description "to BL-1 (VRF_TENANT_NS)";
        peer-as 64518;
    }
    neighbor 182.18.0.2 { <<< eBGP session with BL-2 (VRF_TENANT_1)
        description "to BL-2 (VRF_TENANT_1)";
        peer-as 65995;
    }
    neighbor 182.19.0.2 { <<< eBGP session with BL-1 (VRF_TENANT_NS)
        description "to BL-2 (VRF_TENANT_NS)";
        peer-as 64519;
    }
}

jnpr@SRX> show configuration policy-options policy-statement adv-default
term static-rt {
    from {
        protocol static; <<< SRX advertises 0/0 to all border-leaf devices
        route-filter 0.0.0.0/0 exact;
    }
}

```

```

    then accept;
}
term default {
    then reject;
}

```

Verification

SRX receives data center tenant routes, as well as those for northbound hosts across the WAN. All North to South traffic is directed to the firewall, which can then route it optimally since exact host location (by means of /32 routes) is known.

```
jnpr@SRX> show route 100.0.10.100
```

```
inet.0: 37 destinations, 41 routes (37 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
100.0.10.100/32    *[BGP/170] 00:09:06, localpref 100
                  AS path: 65885 65200 I, validation-state: unverified
                  > to 172.17.10.1 via xe-0/0/2.1 <<< To BL-1
```

```
jnpr@SRX> show route 100.0.10.101
```

```
inet.0: 37 destinations, 41 routes (37 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
100.0.10.101/32    *[BGP/170] 00:08:11, localpref 100
                  AS path: 65995 65200 I, validation-state: unverified
                  > to 182.18.0.2 via xe-0/0/3.2 <<< To BL-2
```

```
jnpr@SRX> show route 100.0.90.109
```

```
inet.0: 37 destinations, 41 routes (37 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
100.0.90.109/32    *[BGP/170] 03:00:17, localpref 100
                  AS path: 65885 65200 I, validation-state: unverified
                  > to 172.17.10.1 via xe-0/0/2.1 <<< To BL-1
```

<<< Traffic from SRX to WAN load-balanced to both BL-1 and BL-2

```
jnpr@SRX> show route forwarding-table destination 77.18.1.1
```

```
Routing table: default.inet
```

```
Internet:
```

Destination	Type	RtRef	Next hop	Type	Index	NhRef	Netif
77.18.1.0/24	user	0		ulst	262143	3	
			182.17.0.2	ucst	557	5	xe-0/0/2.2
			182.19.0.2	ucst	563	5	xe-0/0/3.3

<<< WAN PEs do not see tenant host routes only the 0/0 route from the SRX

```
jnpr@PE-3> show route 100.0.10.100
```

```
VRF_TENANT_NS.inet.0: 6 destinations, 8 routes (6 active, 0 holddown, 0 hidden)
```

```
@ = Routing Use Only, # = Forwarding Use Only
```

```
+ = Active Route, - = Last Active, * = Both
```

```
0.0.0.0/0          @[BGP/170] 1w6d 09:25:19, localpref 100, from 101.0.0.1
                  AS path: 64528 64518 65009 I, validation-state: unverified
                  > to 182.16.0.4 via xe-2/1/2.0, Push 299824
                  [BGP/170] 1w6d 08:44:09, localpref 100, from 101.0.0.2
                  AS path: 64529 64519 65009 I, validation-state: unverified
                  > to 182.16.0.6 via xe-2/1/3.0, Push 299824
                  #[Multipath/255] 1w6d 08:44:09, metric2 1
                  > to 182.16.0.4 via xe-2/1/2.0, Push 299824
                  to 182.16.0.6 via xe-2/1/3.0, Push 299824
```


Traffic Flows

Figure 2.15 depicts the path traversed by traffic between hosts across the WAN and in the DC, hair-pinning through the DC firewall (SRX):

- H0 (VRF_TENANT_NS: IP 77.18.1.1) and host H1-5 (VRF_TENANT_1: IP 100.0.90.109)
- H0 (VRF_TENANT_NS: IP 77.18.1.1) and host H2-1 (VRF_TENANT_1: IP 100.0.10.101)

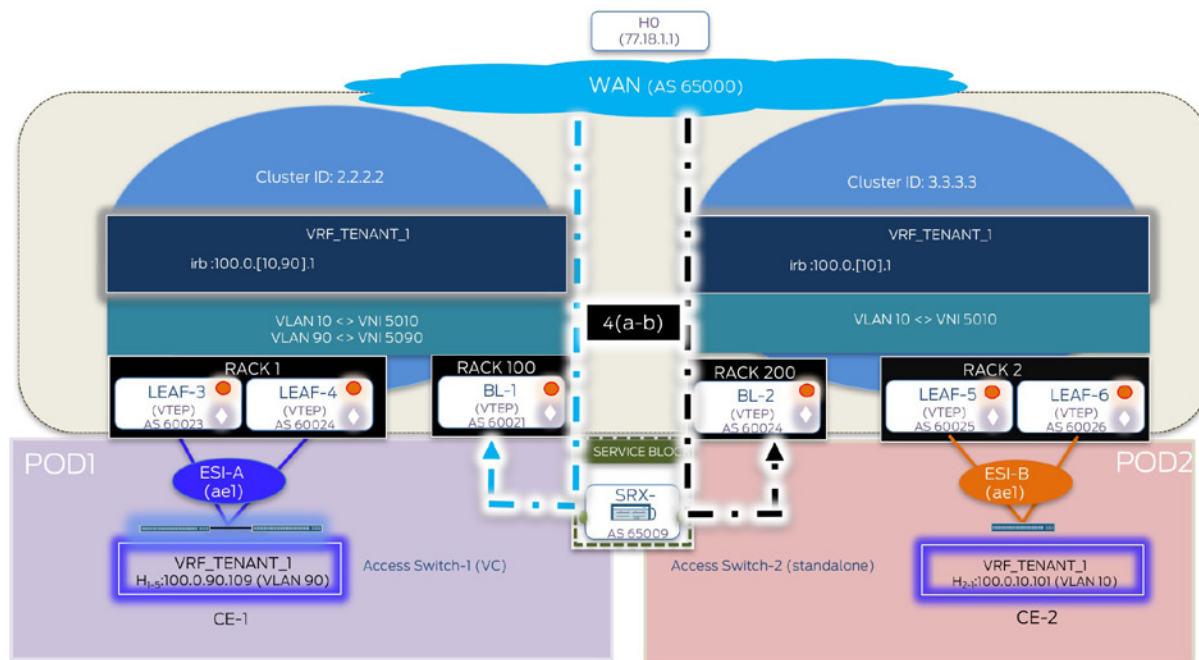


Figure 2.15 Manual Service Chaining for N-S – Inter-VRF Inter-subnet Traffic (H0 > H1-5 and H0 > H2-1)

Hair-pinning through the SRX is observed for depicted traffic.

```
SRX
Interface    Link  Input packets      Seconds: 2      Time: 07:25:11
              Up    (pps)              (pps)
xe-0/0/2     Up    1198849120         (1000)         1086030237
xe-0/0/3     Up    794510737          (1016)         903062111
```

```
jnpr@SRX> show security flow session
```

```
Session ID: 289562, Policy name: 1/6, Timeout: 60, Valid
```

```
In: 77.18.1.1/3 --> 100.0.90.109/3;udp, Conn Tag: 0x0, If: xe-0/0/3.3, Pkts: 232, Bytes: 342896,
Out: 100.0.90.109/3 --> 77.18.1.1/3;udp, Conn Tag: 0x0, If: xe-0/0/2.1, Pkts: 0, Bytes: 0,
```

```
Session ID: 289563, Policy name: 1/6, Timeout: 60, Valid
```

```
In: 77.18.1.1/3 --> 100.0.10.101/3;udp, Conn Tag: 0x0, If: xe-0/0/2.2, Pkts: 232, Bytes: 342896,
Out: 100.0.10.101/3 --> 77.18.1.1/3;udp, Conn Tag: 0x0, If: xe-0/0/3.2, Pkts: 0, Bytes: 0,
```

IXIA Statistics – Intra DC Inter-POD

(H0 > H1-5: 77.18.1.1 > 100.0.90.109, H0 > H1-5: 77.18.1.1 > 100.0.10.101)

(1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %
4a-NS Inter VRF Inter VNI: vlan 60 > 90: 77.18.1.0 > 100.0.90.109	271,873	271,329	544	0
4b-NS Inter VRF Inter VNI: vlan 60 > 10: 77.18.1.0 > 100.0.10.101	271,873	271,329	544	0

Chapter 3

Building a Data Center – Routing at the Spine Layer

Contents

High-Level Overview100

Topology Description..... 101

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet):106

Manual Service Chaining of East-West Traffic 132

Manual Service Chaining of North-South Traffic 142

Class of Service146



Chapter 2 discussed the different building blocks to construct a data center fabric with multi-tenant PODs using routing at the leaf layer. This chapter looks at an alternate design approach to achieve the same objectives. The chapter's case study has three main sections:

- High-level summary: overview on the projected design and products being positioned.
- Design summary: Technical details on the involved design components.
- Traffic scenarios: Delves further into configuration and verification details on common traffic use cases.

Readers interested in design but not detailed configuration can skip *Traffic Scenarios* section.

High-Level Overview

In this design approach, routing is done at the spine layer thus making it possible to reduce functionality and scale requirements at the leaf layer. Fabric/super-spine and spine devices also act as overlay route-reflectors. In this use case, distribution of functional responsibilities has been demonstrated by using fabric/super-spine devices as service nodes, which are responsible for servicing each POD. These fabric devices host the intelligence or control-plane logic and necessary scale capabilities to perform additional functions, as compared to the other devices in the IP fabric. Leaf devices are L2 EVPN PE's only, spines act as L3 gateways, and fabric/super-spine devices act as DC edge devices that connect to WAN also as service nodes, i.e. devices in the IP fabric that direct traffic to the service block, which can consist of a firewall, load-balancer, etc. A typical deployment example could use QFX 5100/5200 as leaf devices, QFX 10002/8/16 as spine devices, and MX/QFX 10000 as DC edge devices. A high level design representation for this chapter has been depicted in Figure 3.1.

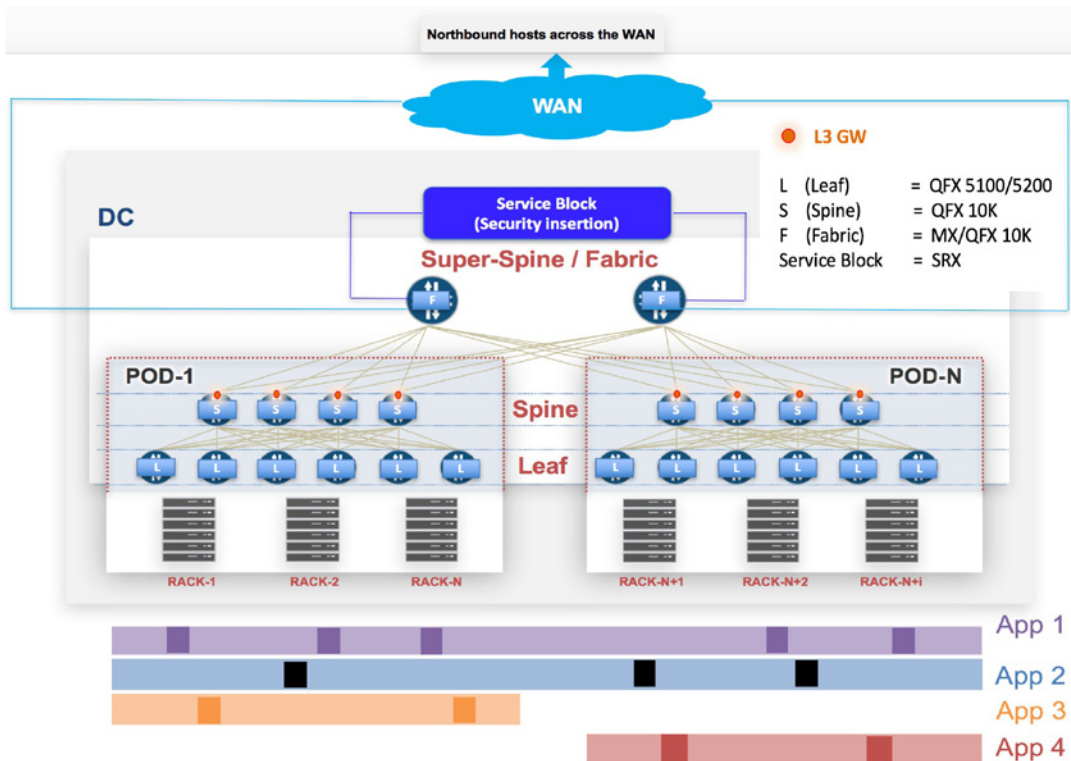


Figure 3.1 High-level Design Representation

Topology Description

- *Leaf devices:* QFX5100-48S-6Q have been used as leaf devices. For demonstration purposes, POD1 consists of two leaf devices and POD2 consists of four leaf devices.
- *Spine devices:* Four QFX10002-72Q have been used as spine devices, two in each POD.
- *Fabric devices:* Two MX960 have been used as fabric/super-spine devices to connect all PODs.
- *Service block:* One SRX5600 has been used to demonstrate Layer 3 security insertion for both EW and NS traffic for all PODs. The term service block in this example refers to the SRX acting as the firewall for service chaining Layer 3 inter-tenant traffic.
- *CE devices:* Access switches (standalone QFX5110-48S-4C in each POD) and IXIA simulate CE devices in both PODs. Servers can be directly plugged in to the TOR/leaf devices. To demonstrate the use of LACP, intermediate switches have been used here due to resource constraints. The term host (simulated on IXIA) implies any application entity (VM or container) that consumes an IP/MAC address.
- *WAN:* External tester port plugged into a Layer 2 switch (QFX5100-48S-6Q) has been used to simulate a BGP peer across the WAN.

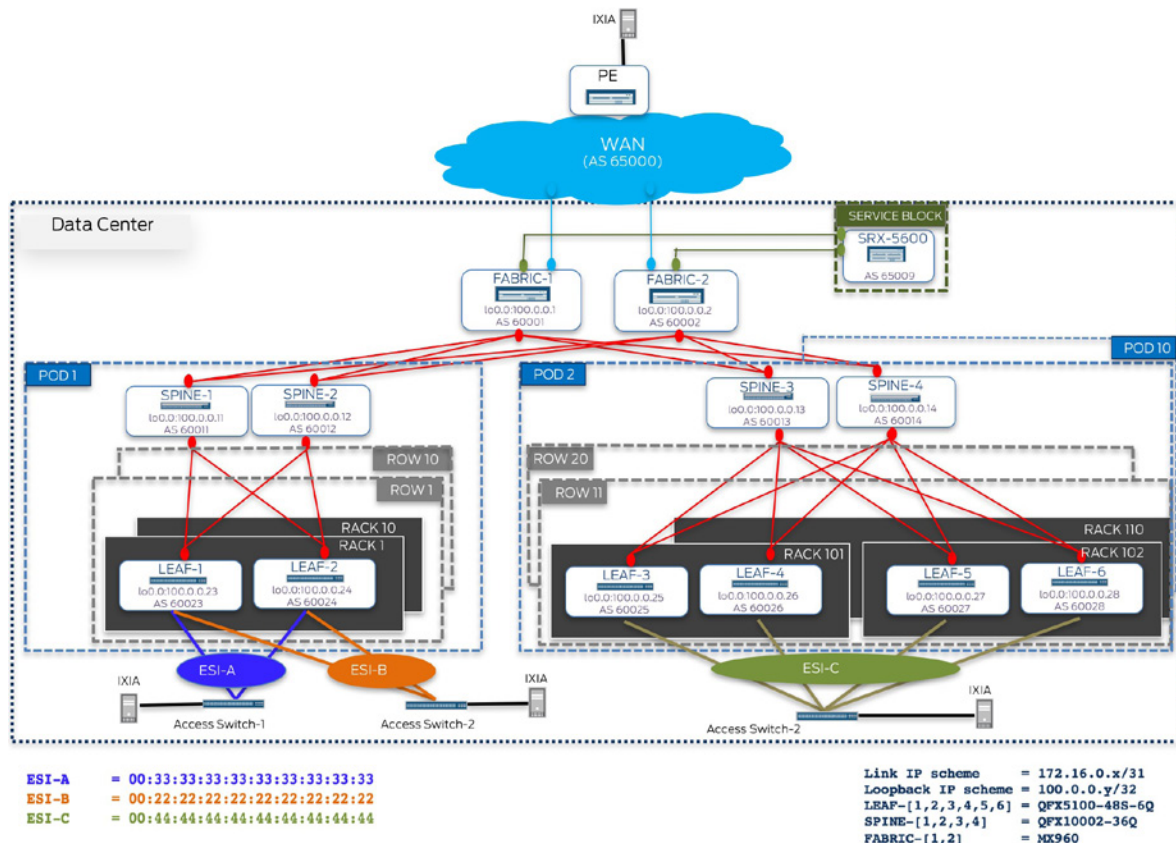


Figure 3.2 Chapter 3's Physical Topology

Here are the general data center layout assumptions mapped to the physical testbed for the demonstrated case study:

- One row consists of 10 racks.
- Two leaf devices as redundant TORs are used for each rack.
- Two spine devices connect ten rows each of ten racks. These two spine devices connecting ten rows of ten racks each and leaf devices, acting as TORs for each rack in each row, all together constitute a POD.
- Fabric/super-spine devices further connect all the PODs together. They also connect each POD to the SRX (Service Block) and the external WAN, thus acting as both service nodes and DC edge devices.
- Ten PODs together constitute a data center.

Design Summary

Traffic flows demonstrated:

- East-West (traffic across PODs within the DC)
 - Layer 2 (Intra-VRF Intra-subnet): Traffic between hosts in the same subnet for a given tenant.
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Traffic between hosts in different subnets for different tenants and is manually service-chained through the SRX (service block).
- North-South (traffic between hosts across the WAN and within the DC)
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Traffic between hosts in different subnets for different tenants, one of which is external to the data center and is manually service-chained through the SRX (service block).

Access considerations:

- Two leaf devices are acting as a pair of redundant Top of Rack switches. CE-1 represents groups of hosts (e.g. VMs/containers residing on servers) mapped to ESI-A (00:11:11:11:11:11:11:11:11:11) on ae1 for LEAF-1 and LEAF-2 in POD-1. Similarly, hosts on CE-2 are mapped to ESI-B (00:33:33:33:33:33:33:33:33:33) on ae0 for LEAF-1 and LEAF-2 and hosts on CE-3 are mapped to ESI-C (00:44:44:44:44:44:44:44:44:44) on ae2 for LEAF-3, LEAF-4, LEAF-5, LEAF-6 to demonstrate 4-way multi-homed PEs.

Underlay considerations:

- EBGp has been used to achieve underlay IP reachability (lo0 address reachability between VTEPs). EBGp with unique AS number per device is used to achieve underlay IP reachability.
- MTU has been set on all physical interfaces for devices in a data center to account for VXLAN encapsulation.

Overlay considerations:

- L3 gateway placement:

- VXLAN routing in this use case is done on the spine devices. Each spine device acts as a Layer 3 gateway for VNIs residing in that POD. Fabric devices in this example do not act as Layer 3 gateways.
- In this example, overlay support for IPv6 user traffic has been demonstrated. IPv6 traffic between tenant hosts is carried over IPv4 underlay. Dual stack address (v4 and v6) are configured only on IRB interfaces when IP lookup is done on spine nodes to support inter-subnet traffic.
- Routing is done only at the spine layer (also VTEPs), while leaf devices acting only as Layer 2 gateways. Fabric devices act only as transit and have no data plane configuration, therefore, fabric devices are only overlay route reflectors not VTEPs.
- Distributed Layer 3 gateways:
 - Distributed Layer 3 gateway functionality has been achieved by using virtual-gateway address (same anycast IP/MAC) on IRB interfaces of serving gateways for a given subnet. IRB interfaces for subnets of a given tenant, that need to communicate with each other are placed in the same tenant IP-VRF. Single tenant (VRF_TENANT_MAIN) environment has been demonstrated for intra/inter subnet communication. This IP-VRF hosts IRB interfaces for VLANs 100,101,102,103,105,106 in POD1 and 105,106,108,109 in POD2. IP-VRF for NS traffic (VRF_TENANT_NS) exists only on the service node devices in each POD and populate routes received from the WAN, independent of the EVPN domain.
- EVPN NLRI exchange:
 - MP-iBGP sessions have been to exchange EVPN NLRI.
- Route reflection:
 - To avoid full-mesh, MP-iBGP EVPN sessions have been created between the leaf devices functioning as clients and spine devices being overlay route-reflectors. In POD-1, LEAF-1 and LEAF-2 act as clients to SPINE-1 and SPINE-2 serving as RRs (cluster ID 2.2.2.2 in POD-1). In POD-2, LEAF-3,4,5,6 act as clients to SPINE-3 and SPINE-4 serving as RRs (cluster ID 3.3.3.3 in POD-2).
 - Hierarchical route reflection for the overlay is in use. Fabric devices acting as route reflectors, thus preventing a full-mesh of BGP sessions between spine devices (cluster-ID 1.1.1.1).
 - Full mesh of VXLAN tunnels exist with VTEPs located on each leaf and spine node.
- Service interface:
 - VLAN aware EVPN service is in use (please refer to the Appendix for more details on EVPN service interfaces).
- Host communication:
 - Type 2 routes (asymmetric) are used to exchange reachability information for VLANs 105 and 106 (mapped to VNI 9105 and 9106) stretched across PODs, allowing for intra-subnet communication between these hosts. Type 2 routes (asymmetric) are used to exchange reachability information for hosts in all other VLANs 100, 101, 102, 103 (mapped to VNI 9100, 9101, 9102, 9103) in POD-1 and hosts in VLANs 108, 109 (mapped to VNI 9108 and 9109) in POD-2, allowing for inter-subnet communication.

- Layer 2: POD1 <> POD 2 (Intra-VRF/same tenant) --- > Type 2
- Layer 3: POD1 <> POD 2 (Intra-VRF/same tenant) --- > Type 2
- Currently, leaf nodes with hosts that need to communicate with each other support configuration for the desired VLANs. For example, since no hosts in VLAN 100 reside on CE-3 connected to LEAF-3,4,5,6 (ESI-C) and no intra-subnet communication is intended for the same (CE-1 or CE-2 to CE-3 in VLAN 100), the same set of VLANs are not configured on all leaf nodes i.e. VLAN 100 is not configured on LEAF-3,4,5,6. However, since inter-subnet communication has been established using Type 2 routes (asymmetric forwarding), all spine devices (Layer 3 gateways) are configured with the same set of VLANs. Auto route-targets (explained later in the configuration section) can be used to ease provisioning. Type 2 routes do pose the advantage of allowing for pre-provisioning in advance in case Layer 2 stretch is needed later, as requirements in the data center evolve.
- Configuration in subsequent sections also shows application of common class-of-service functions in an EVPN-VXLAN fabric, and highlights the rewrite function taking into effect on VXLAN encapsulation.

Manual service chaining:

■ Service Nodes

- Here, fabric devices act as service nodes as well as DC edge devices that connect the data center to the WAN. This provides the advantage of demarcation of functional responsibilities and the use of more powerful devices as DC edge for enhanced features and scale.
- The term service node is used for devices that direct traffic from the IP-fabric towards the service block, which in this case are the fabric devices connecting all PODs.

■ Service Block

- Layer 3 security insertion has been demonstrated for inter-tenant EW and NS traffic. The service block in this use case consists of a firewall that services all PODs. It is not shown here, but SRX clusters can be used one for each POD, or the entire DC, depending on scale and high-availability requirements.
- SRX is physically connected to each fabric device. Physical path for traffic between all different PODs goes through the connecting fabric layer. As such, to service chain EW and NS traffic, this is an optimal point for hosting control plane logic to support security insertion.
- Hair-pinning through the SRX is only treated as an IP path problem, i.e. getting intended traffic to the SRX. There is no coverage on details regarding SRX zoning or policy enforcement.

■ Manual service-chaining orchestration for EW inter-tenant inter-subnet traffic:

- Manual service chaining through the SRX has been set up for inter-subnet traffic between different tenants (VRF_TENANT_CLEAN – VLAN 90/POD1 and VRF_TENANT_DIRTY – VLAN 91/POD2).
- Layer 3 SRX: POD1 <> POD2 (Inter-VRF/different tenants) --- > back-to-back VRFs are created on the Layer 3 gateways (spine) and DC edge/service node (fabric device). An eBGP session (family inet) that is created over the connecting

link between spine and fabric devices, resides in the tenant IP-VRF. All desired summary routes in that IP-VRF are exchanged through eBGP by fabric devices with SRX and 0/0 from SRX advertised into IP fabric by the service nodes.

- Manual service-chaining orchestration for NS inter-tenant inter-subnet traffic:
 - Layer 3 SRX: Host across WAN connected to fabric <> POD1 (Inter-VRF/different tenants) --- > One additional IP-VRF is created on each fabric device to receive the northbound prefix from the WAN, which is advertised via eBGP session to the SRX and 0/0 from SRX is advertised into the IP fabric.

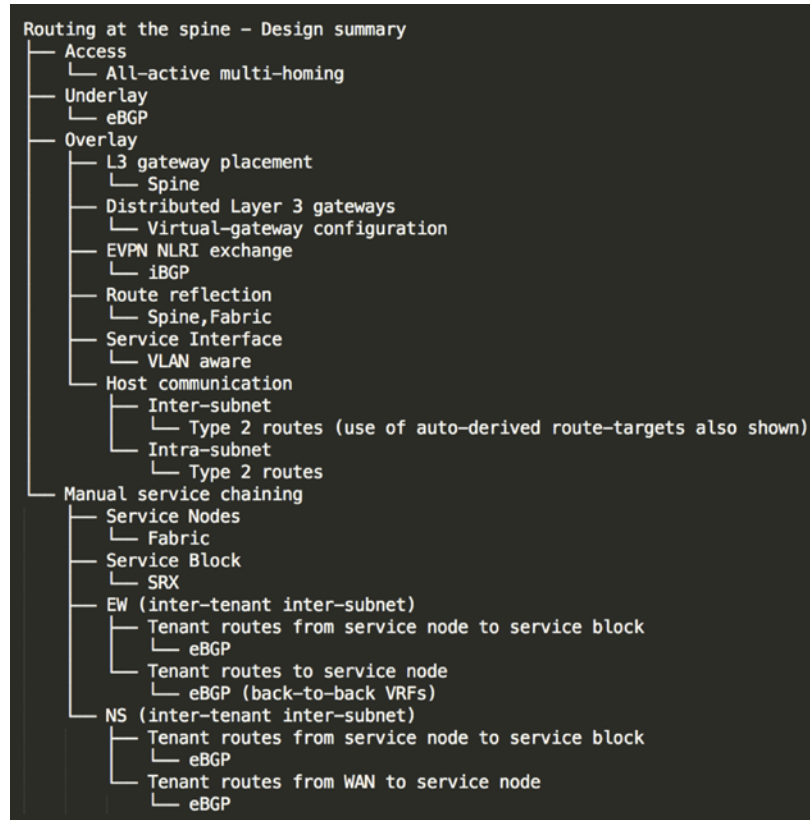


Figure 3.3 Design Summary for Case Study (Routing at Spine)

Traffic Scenarios

This chapter demonstrates implementation details in three traffic scenario use cases. East-West traffic for hosts in the same tenant (Intra-VRF: Intra-subnet and Inter-subnet):

- Configuration
- Verification
- Traffic flows

Manual service chaining of East-West traffic for hosts in different tenants (Layer 3 security insertion: Inter-tenant Inter-subnet):

- Configuration layout

- Route exchange between service nodes and service blocks
- Configuration
- Verification
- Traffic flows

Manual service chaining of North-South traffic for hosts in different tenants (Layer 3 security insertion: Inter-tenant Inter-subnet):

- Configuration layout
- Route exchange between service nodes and service blocks
- Configuration
- Verification
- Traffic flows

Let's begin with the first use case.

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Figure 3.4 highlights the logical topology for E-W traffic across PODs and all the details on the configuration and verification steps follow.

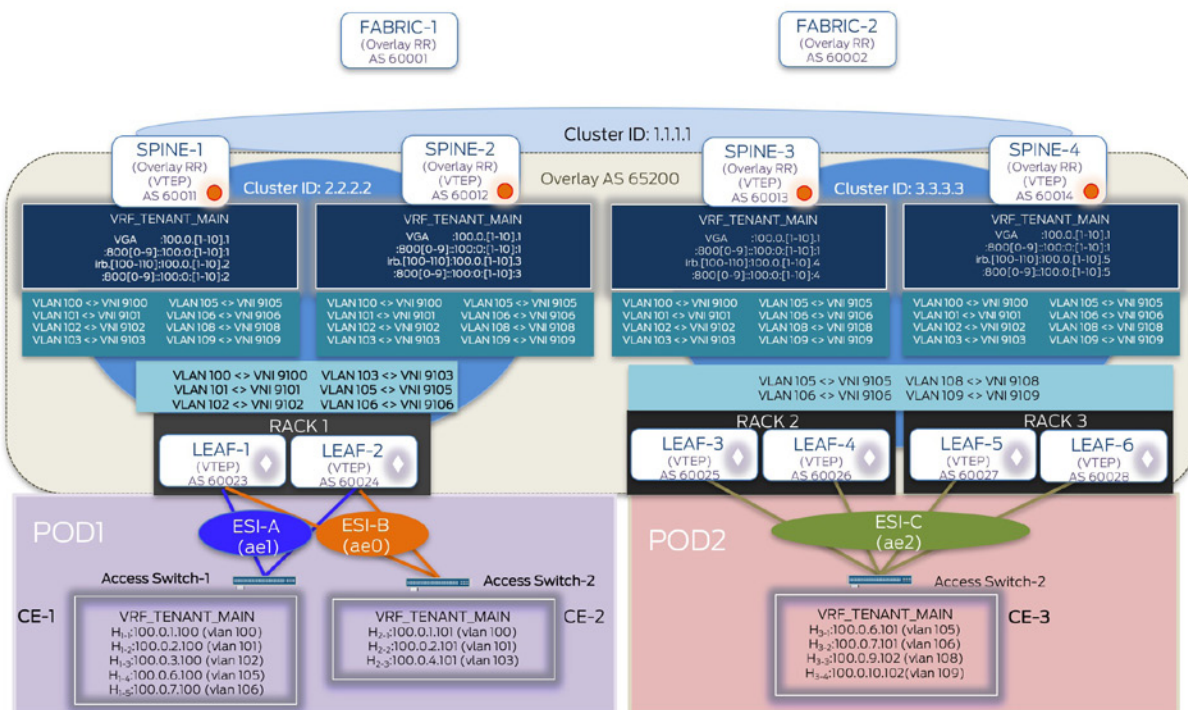


Figure 3.4 Logical Topology – EW Traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Configuration

A detailed walkthrough is done for the configuration of only LEAF-1, SPINE-1, LEAF-3, SPINE-3, and FABRIC-1. All other devices in the testbed are similarly configured. Configuration is divided into three components: Underlay, Overlay, and Access, and highlighted for leaf, spine, and fabric devices.

Underlay

Overhead due to VXLAN encapsulation is accounted for on all devices in the IP fabric:

```
jnpr@LEAF-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@LEAF-3> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@SPINE-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@SPINE-3> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@FABRIC-1> show configuration apply-groups | except #
apply-groups [ MTU-VXLAN ];

jnpr@LEAF-1> show configuration groups MTU-VXLAN <<< MTU set to account for overhead due to VXLAN
encapsulation on all leaf, spine and fabric devices
interfaces {
  <*> {
    mtu 9192;
    unit <*> {
      family inet {
        mtu 9000;
      }
    }
  }
}
```

The specifics for the EBGp (underlay) configuration on leaf devices (LEAF-1 in POD-1, and LEAF-3 in POD-2) are:

```
jnpr@LEAF-1> show configuration protocols bgp group underlay-ipfabric
type external; <<< Each leaf device exchanges loopback addresses over the EBGp session with each
connecting spine
mtu-discovery;
import bgp-ipclos-in;
export bgp-ipclos-out;
local-as 60023;
bfd-liveness-detection {
  minimum-interval 350;
  multiplier 3;
  session-mode automatic;
}
multipath multiple-as;
neighbor 172.16.0.4 {
  peer-as 60011;
}
neighbor 172.16.0.16 {
  peer-as 60012;
}
jnpr@LEAF-3> show configuration protocols bgp group underlay-ipfabric
type external;
mtu-discovery;
import bgp-ipclos-in;
export bgp-ipclos-out;
local-as 60025;
bfd-liveness-detection {
```

```

        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath multiple-as;
neighbor 172.16.0.32 {
        peer-as 60013;
    }
neighbor 172.16.0.44 {
        peer-as 60014;
    }

jnpr@LEAF-1> show configuration policy-options policy-statement bgp-ipclos-in
term loopbacks {
    from {
        route-filter 100.0.0.0/16 orlonger;
    }
    then accept;
}

jnpr@LEAF-1> show configuration policy-options policy-statement bgp-ipclos-out
term loopback {
    from { <<< Similar policy configuration exists on all leaf devices

        protocol direct;
        route-filter 100.0.0.0/16 orlonger;
    }
    then {
        next-hop self;
        accept;
    }
}
term reject {
    then reject;
}

jnpr@LEAF-1> show configuration routing-
options <<< No global AS has been configured on any leaf device and local-
as is used to establish BGP sessions for both underlay and overlay
router-id 100.0.0.23;
forwarding-table {
    export pfe-ecmp;
}

jnpr@LEAF-3> show configuration routing-options
router-id 100.0.0.25;
forwarding-table {
    export pfe-ecmp;
}

```

The specifics for underlay configuration on spine devices (SPINE-1 in POD-1, and SPINE-3 in POD-2) are:

```

jnpr@LEAF-3> show configuration policy-options policy-statement pfe-ecmp <<< Applied on all devices
then {
    load-balance per-packet;
}

```

```

jnpr@SPINE-1> show configuration protocols bgp group underlay-ipfabric
type external;
mtu-discovery;
import bgp-ipclos-in; <<< Each spine device exchanges loopback addresses over the EBGp session with each
connecting leaf and fabric device
export bgp-ipclos-out;
local-as 60011;

```

```

bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath multiple-as;
neighbor 172.16.0.5 {
    peer-as 60023;
}
neighbor 172.16.0.7 {
    peer-as 60024;
}
neighbor 172.16.0.50 {
    peer-as 60001;
}
neighbor 172.16.0.58 {
    peer-as 60002;
}

```

```

jnpr@SPINE-3> show configuration protocols bgp group underlay-ipfabric
type external;
mtu-discovery;
import bgp-ipclos-in;
export bgp-ipclos-out;
local-as 60013;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath multiple-as;
neighbor 172.16.0.33 {
    peer-as 60025;
}
neighbor 172.16.0.35 {
    peer-as 60026;
}
neighbor 172.16.0.105 {
    peer-as 60027;
}
neighbor 172.16.0.113 {
    peer-as 60028;
}
neighbor 172.16.0.54 {
    peer-as 60001;
}
neighbor 172.16.0.62 {
    peer-as 60002;
}

```

```

jnpr@SPINE-1> show configuration policy-options policy-statement bgp-ipclos-in
term loopbacks {
    from {
        route-filter 100.0.0.0/16 orlonger;
    }
    then accept;
}

```

```

jnpr@SPINE-1> show configuration policy-options policy-statement bgp-ipclos-out
term loopback {
    from {
        protocol direct;
        route-
filter 100.0.0.0/16 orlonger; <<< Leaf devices in each POD prefer local POD Layer 3 gateways
}
    then {
        community add MYCOMMUNITY;
        next-hop self;
        accept;
    }
}

```

```

    }
}
term as-path {
    from {
        as-path asPathLength2;
        community MYCOMMUNITY;
    }
    then reject;
}
!
as-path asPathLength3 "{2,}";

```

<<< Each leaf device has local POD spine reachability (VTEP IP = 100) only from the respective spine, which prevents another leaf device from being used as transit for data traffic if reachability to preferred spine is lost

```

jnpr@LEAF-1> show route 100.0.0.11
inet.0: 35 destinations, 43 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100.0.0.11/32      *[BGP/170] 4d 12:47:59, localpref 100
                  AS path: 60011 I, validation-state: unverified
                  > to 172.16.0.4 via et-0/0/48.0

:vlan.inet.0: 26 destinations, 26 routes (24 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.0.11/32      *[Static/1] 4d 12:47:47, metric2 0
                  > to 172.16.0.4 via et-0/0/48.0

```

```

jnpr@LEAF-1> show route 100.0.0.12
inet.0: 35 destinations, 43 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100.0.0.12/32      *[BGP/170] 1d 14:24:15, localpref 100
                  AS path: 60012 I, validation-state: unverified
                  > to 172.16.0.16 via et-0/0/49.0

:vlan.inet.0: 26 destinations, 26 routes (24 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.0.12/32      *[Static/1] 2d 15:50:19, metric2 0
                  > to 172.16.0.16 via et-0/0/49.0

```

```

jnpr@SPINE-1> show configuration routing-options
router-id 100.0.0.11;
forwarding-table {
    export pfe-ecmp;
}

```

<<< No global AS is configured under 'routing-options' hierarchy (local-as is used to establish BGP sessions for both underlay and overlay). This is done to ensure that the system-generated ESI for anycast MAC (Type 2 route for irb interfaces) of all redundant layer 3 gateways is the same. The system-generated ESI takes is computed based on the VNID and global AS, if configured.

```

jnpr@SPINE-3> show configuration routing-options
router-id 100.0.0.13;
forwarding-table {
    export pfe-ecmp;
}

```

Type 2 routes for the virtual-gateway MAC are multi-homed to the same system-generated ESI that factors in global AS (if one configured) and VNID (output below shows Type 2 routes for virtual-gateway MAC received by LEAF-1 from SPINE-1 and SPINE-2 for VNID [9100]10 = [23:8c]16):

```

jnpr@LEAF-1> show route table bgp.evpn.0 detail
2:100.0.0.11:1::9100::00:00:00:00:10:00/304 (2 entries, 0 announced)

```

```

*BGP      Preference: 170/-101
          Route Distinguisher: 100.0.0.11:1
          Next hop type: Indirect
          Address: 0x99120b8
          Next-hop reference count: 348
          Source: 100.0.0.11
          Protocol next hop: 100.0.0.11
          Indirect next hop: 0x2 no-forward INH Session ID: 0x0
          State: <Active Int Ext>
          Peer AS: 65200
          Age: 31:02      Metric2: 0
          Validation State: unverified
          Task: BGP_65200_65200.100.0.0.11+56070
          AS path: I
          Communities: target:1:9100 encapsulation0:0:0:0:vxlan
          Import Accepted
          Route Label: 9100
          ESI: 05:00:00:00:00:00:23:8c:00
          Localpref: 100
          Router ID: 100.0.0.11
          Secondary Tables: default-switch.evpn.0
!
2:100.0.0.12:1::9100::00:00:00:00:10:00/304 (2 entries, 0 announced)
*BGP      Preference: 170/-101
          Route Distinguisher: 100.0.0.12:1
          Next hop type: Indirect
          Address: 0x991219c
          Next-hop reference count: 300
          Source: 100.0.0.12
          Protocol next hop: 100.0.0.12
          Indirect next hop: 0x2 no-forward INH Session ID: 0x0
          State: <Active Int Ext>
          Peer AS: 65200
          Age: 1d 1:31:44      Metric2: 0
          Validation State: unverified
          Task: BGP_65200_65200.100.0.0.12+52938
          AS path: I
          Communities: target:1:9100 encapsulation0:0:0:0:vxlan
          Import Accepted
          Route Label: 9100
          ESI: 05:00:00:00:00:00:23:8c:00
          Localpref: 100
          Router ID: 100.0.0.12
          Secondary Tables: default-switch.evpn.0

```

The specifics for underlay configuration on fabric devices (FABRIC-1) are:

```

jnpr@FABRIC-1> show configuration protocols bgp group underlay-ipfabric
type external;
mtu-discovery;
import bgp-ipclos-in; <<< Each fabric device exchanges loopback addresses over the EBGp session with each
connecting spine device

export bgp-ipclos-out;
local-as 60001;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath multiple-as;
neighbor 172.16.0.51 {
    peer-as 60011;
}
neighbor 172.16.0.53 {
    peer-as 60012;
}
neighbor 172.16.0.55 {
    peer-as 60013;
}

```



```

neighbor 172.16.0.57 {
    peer-as 60014;
}

jnpr@FABRIC-1> show configuration policy-options policy-statement bgp-ipclos-in
term loopbacks {
    from {
        route-filter 100.0.0.0/16 orlonger;
    }
    then accept;
}

jnpr@FABRIC-1> show configuration policy-options policy-statement bgp-ipclos-out
term loopback {
    from {
        protocol direct;
    }
    route-filter 100.0.0.0/16 orlonger;
    then {
        next-hop self;
        accept;
    }
}

```

Overlay

The configuration components required for overlay configuration include the MP-iBGP configuration and the EVI configuration.

The MP-iBGP sessions exist between leaf devices (acting as clients) and spine devices (overlay route reflectors) to support EVPN NLRI. BFD has been used for faster failure detection for BGP. Each POD uses a different cluster ID so as to be able to exchange routes between PODs. Hierarchical route reflection is in use with fabric devices acting as overlay route reflectors (spine devices acting as clients), which prevents full-mesh iBGP requirement between spine devices.

```

jnpr@LEAF-1> show configuration protocols bgp group overlay-evpn
type internal;
local-address 100.0.0.23;
family evpn {
    signaling;
}
import OVERLAY-IN; <<< Import policy for leaf devices to prefer local POD Layer 3 gateways
local-as 65200;
bfd-liveness-detection {
    minimum-interval 350; <<< Leaf devices act as clients (NLRI = evpn)
multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 100.0.0.11;
neighbor 100.0.0.12;

jnpr@LEAF-3> show configuration protocols bgp group overlay-evpn
type internal;
local-address 100.0.0.25;
import OVERLAY-IN;
family evpn {
    signaling;
}
local-as 65200;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;

```

```
neighbor 100.0.0.13;
neighbor 100.0.0.14;
```

```
jnpr@LEAF-1> show configuration policy-options policy-statement OVERLAY-IN
term reject-remote-gw { <<< LEAF-1,2 prefer local PoD SPINE-1,2 as Layer 3 gateways
from {
    family evpn;
    next-hop [ 100.0.0.13 100.0.0.14 ];
    nlri-route-type [ 1 2 ];
}
    then reject;
}
term accept-all {
    then accept;
}
```

```
jnpr@LEAF-3> show configuration policy-options policy-statement OVERLAY-IN
term reject-remote-gw { <<< LEAF-5,6,7,8 prefer local POD SPINE-3,4 as Layer 3 gateways
from {
    family evpn;
    next-hop [ 100.0.0.11 100.0.0.12 ];
    nlri-route-type [ 1 2 ];
}
    then reject;
}
term accept-all {
    then accept;
}
```

```
jnpr@SPINE-1> show configuration protocols bgp group overlay-evpn
type internal;
local-address 100.0.0.11;
family evpn {
    signaling; <<< Spine devices act as overlay route-reflectors (NLRI = evpn)
}
cluster 2.2.2.2;
local-as 65200;
multipath;
neighbor 100.0.0.23;
neighbor 100.0.0.24;
```

```
jnpr@SPINE-3> show configuration protocols bgp group overlay-evpn
type internal;
local-address 100.0.0.13;
family evpn {
    signaling;
}
vpn-apply-export;
cluster 3.3.3.3;
local-as 65200;
multipath;
neighbor 100.0.0.25;
neighbor 100.0.0.26;
neighbor 100.0.0.27;
neighbor 100.0.0.28;
```

```
jnpr@SPINE-1> show configuration protocols bgp group overlay-hrr-clients
type internal;
local-address 100.0.0.11;
family evpn { <<< Spine devices are clients to fabric devices acting as overlay route-reflectors (NLRI = evpn)
signaling;
}
local-as 65200;
multipath;
neighbor 100.0.0.1;
neighbor 100.0.0.2;
```

```

jnpr@SPINE-3> show configuration protocols bgp group overlay-hrr-clients
type internal;
local-address 100.0.0.13;
family evpn {
    signaling;
}
local-as 65200;
multipath;
neighbor 100.0.0.1;
neighbor 100.0.0.2;

jnpr@FABRIC-1> show configuration protocols bgp group overlay-evpn
type internal;
local-address 100.0.0.1;
family evpn {
    signaling;
}
cluster 1.1.1.1; <<< Fabric devices act as overlay route-
reflectors - hierarchical route reflection (NLRI = evpn)
local-as 65200;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 100.0.0.11;
neighbor 100.0.0.12;
neighbor 100.0.0.13;
neighbor 100.0.0.14;

```

EVI Configuration

At the time of this writing, QFX 10002 platforms support VLAN-aware service using a single global virtual switch. The EVI-related configuration on the QFX 10002 platforms includes VLAN-VNI mapping.

Under the ‘vlangs’ configuration hierarchy, each local VLAN is mapped to a unique VNI on each VTEP. As explained earlier, leaf nodes with hosts that need to communicate with each other support configuration for the desired VLANs while all spine devices are configured with the same set of VLANs since Type 2 routes have been used for inter-subnet communication. With 15.1X53-D60, the definition of ingress-node-replication (in the ‘vlan’ hierarchy) for handling BUM traffic replication is optional, since that is currently supported. The recommendation would be to not explicitly specify ingress-node-replication as that interferes with the default optimal behavior.

```

jnpr@LEAF-1> show configuration vlans | except # <<< LEAF-1 and LEAF-
2 share the same set of VLANs and VNI mapping
bd9100 {
    vlan-id 100;
    vxlan {
        vni 9100;
    }
}
bd9101 {
    vlan-id 101;
    vxlan {
        vni 9101;
    }
}
bd9102 {
    vlan-id 102;
    vxlan {
        vni 9102;
    }
}

```

```

bd9103 {
    vlan-id 103;
    vxlan {
        vni 9103;
    }
}
bd9105 {
    vlan-id 105;
    vxlan {
        vni 9105;
    }
}
bd9106 {
    vlan-id 106;
    vxlan {
        vni 9106;
    }
}

```

jnpr@LEAF-3> show configuration vlans | except # <<< LEAF-3,4,5,6 share the same set of VLANs and VNI mapping

```

bd9105 {
    vlan-id 105;
    vxlan {
        vni 9105;
    }
}
bd9106 {
    vlan-id 106;
    vxlan {
        vni 9106;
    }
}
bd9108 {
    vlan-id 108;
    vxlan {
        vni 9108;
    }
}
bd9109 {
    vlan-id 109;
    vxlan {
        vni 9109;
    }
}

```

jnpr@SPINE-1> show configuration vlans | except # <<< SPINE-1,2,3,4 share the same set of VLANs and VNI mapping

```

bd9100 {
    vlan-id 100;
    l3-interface irb.100;
    vxlan {
        vni 9100;
    }
}
bd9101 {
    vlan-id 101;
    l3-interface irb.101;
    vxlan {
        vni 9101;
    }
}
bd9102 {
    vlan-id 102;
    l3-interface irb.102;
    vxlan {
        vni 9102;
    }
}
bd9103 {

```

```

    vlan-id 103;
    l3-interface irb.103;
    vxlan {
        vni 9103;
    }
}
bd9105 {
    vlan-id 105;
    l3-interface irb.105;
    vxlan {
        vni 9105;
    }
}
bd9106 {
    vlan-id 106;
    l3-interface irb.106;
    vxlan {
        vni 9106;
    }
}
bd9108 {
    vlan-id 108;
    l3-interface irb.108;
    vxlan {
        vni 9108;
    }
}
bd9109 {
    vlan-id 109;
    l3-interface irb.109;
    vxlan {
        vni 9109;
    }
}
}

```

Virtual-Switch Configuration

Under the switch-options hierarchy, the vtep-source-interface parameter specifies the VTEP IP address used to establish the VXLAN tunnel. The loopback interface (e.g. lo0.0) is used for this purpose, with reachability provided by the underlay. The route distinguisher (RD) here defines the EVI specific RD carried by Type 1 – AD per EVI, Type 2, and Type 3 routes. The route target (RT) here defines the global route target inherited by EVPN routes, unless specific RT is defined.

In the configuration snippets below, the RT (switch-options hierarchy) is inherited by Type 1 routes and specific RTs manually configured (protocol evpn hierarchy) are inherited by Type 2 and Type 3 routes. Though the output below is highlighted for LEAF-1 and SPINE-1, similar configuration exists on all other leaf and spine devices.

```

jnpr@LEAF-1> show configuration switch-options
vtep-source-interface lo0.0; <<< VXLAN tunnel source address
route-distinguisher 100.0.0.23:1;
vrf-import LEAF-IN;
    vrf-target target:9999:9999; <<< Global route target for all EVPN routes (here Type 1)
jnpr@LEAF-1> show configuration protocols evpn vni-options
vni-options {
    <<< Manual configuration - Specific route targets used by EVPN routes
    for given VNI (here Type 2 and Type 3)
    vni 9100 {
        vrf-target export target:1:9100;
    }
    vni 9101 {
        vrf-target export target:1:9101;
    }
    vni 9102 {
        vrf-target export target:1:9102;
    }
    vni 9103 {

```

```

        vrf-target export target:1:9103;
    }
    vni 9105 {
        vrf-target export target:1:9105;
    }
    vni 9106 {
        vrf-target export target:1:9106;
    }
}

jnpr@LEAF-1> show configuration policy-options policy-statement LEAF-IN
term import_leaf_easi {
    from community comm-leaf_easi; <<< Import filter to accept routes with the global and specific route
targets
    then accept;
}
term import_vni9100 {
    from community com9100;
    then accept;
}
<...>
term default {
    then reject;
}

jnpr@SPINE-1> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 100.0.0.11:1;
vrf-import LEAF-IN;
vrf-target target:9999:9999;

jnpr@SPINE-1> show configuration protocols evpn vni-options
vni 9100 {
    vrf-target export target:1:9100;
}
vni 9101 {
    vrf-target export target:1:9101;
}
vni 9102 {
    vrf-target export target:1:9102;
}
vni 9103 {
    vrf-target export target:1:9103;
}
vni 9105 {
    vrf-target export target:1:9105;
}
vni 9106 {
    vrf-target export target:1:9106;
}
vni 9108 {
    vrf-target export target:1:9108;
}
vni 9109 {
    vrf-target export target:1:9109;
}
}

```

Auto Derived Route Target

Instead of manually specifying VNI RTs, the same can be auto-derived (for more information see https://www.juniper.net/documentation/en_US/junos15.1/topics/example/vrf-target-auto.html). This can ease configuration when a large number of VNIs are in use. In the configuration snippets below, the RT specified is inherited only by Type 1 routes and an import policy is configured for accepting the same. The auto option used with the `vrf-target` statement, can automatically derive RTs for each VNI. This leads to the automatic creation of VRF-import and VRF-export policies to match on these auto-generated VNI RTs. Type 1

(AD per ES and AD per EVI) routes are used by all EVPN aware nodes for functions like fast convergence and aliasing. The separate RT ensures that these routes reach all EVPN PEs. Type 2 and Type 3 routes are generated on a per VNI basis with a different auto-generated route target, which helps confine their propagation to only interested VTEPs.

Outlined next are related configuration specifics on leaf and spine devices (LEAF-1 and SPINE-1):

```
jnpr@LEAF-1> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 100.0.0.23:1;
vrf-import LEAF-IN; <<< RT inherited by all Type 1 routes
vrf-target {
    target:9999:9999;
    auto; <<< Auto-RT generation for Type 2 and 3 routes on a per VNI basis
}
```

```
jnpr@SPINE-1> show configuration switch-options
vtep-source-interface lo0.0;
route-distinguisher 100.0.0.11:1;
vrf-import LEAF-IN;
vrf-target {
    target:9999:9999;
    auto;
}
```

<<< No VNI RTs specified manually

```
jnpr@LEAF-1> show configuration protocols evpn vni-options
```

```
jnpr@LEAF-1>
```

```
jnpr@SPINE-1> show configuration protocols evpn vni-options
```

```
jnpr@SPINE-1>
```

<<< Import filter to accept Type 1 routes

```
jnpr@LEAF-1> show configuration policy-options policy-statement LEAF-IN
term import_leaf_esi {
    from community comm-leaf_esi;
    then accept;
}
term default {
    then reject;
}
```

```
jnpr@SPINE-1> show configuration policy-options policy-statement LEAF-IN
term import_leaf_esi {
    from community comm-leaf_esi;
    then accept;
}
term default {
    then reject;
}
```

<<< vrf-export policy automatically created

(VNI 9100 = target:65200:268444556, VNI 9101 = target:65100:268444557)

```
jnpr@LEAF-1> show policy __vrf-export-autoderive-default-switch-internal__
Term unnamed:
    then community + __vrf-community-default-switch-9100-internal__ [target:65200:268444556 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9101-internal__ [target:65200:268444557 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9102-internal__ [target:65200:268444558 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9103-internal__ [target:65200:268444559 ] accept
Term unnamed:
```



```

    then community + __vrf-community-default-switch-9104-internal__ [target:65200:268444560 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9105-internal__ [target:65200:268444561 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9106-internal__ [target:65200:268444562 ] accept
jnpr@SPINE-1> show policy __vrf-export-autoderive-default-switch-internal__
Term unnamed:
    then community + __vrf-community-default-switch-9100-internal__ [target:65200:268444556 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9101-internal__ [target:65200:268444557 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9102-internal__ [target:65200:268444558 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9103-internal__ [target:65200:268444559 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9104-internal__ [target:65200:268444560 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9105-internal__ [target:65200:268444561 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9106-internal__ [target:65200:268444562 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9107-internal__ [target:65200:268444563 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9108-internal__ [target:65200:268444564 ] accept
Term unnamed:
    then community + __vrf-community-default-switch-9109-internal__ [target:65200:268444565 ] accept
<<< vrf-import policy automatically created to match on auto-generated RTs

```

```

jnpr@LEAF-1> show policy __vrf-import-autoderive-default-switch-internal__
Policy __vrf-import-autoderive-default-switch-internal__:
Term 9100:
    from community __vrf-community-default-switch-9100-internal__ [target:65200:268444556 ]
    then accept
Term 9101:
    from community __vrf-community-default-switch-9101-internal__ [target:65200:268444557 ]
    then accept
Term 9102:
    from community __vrf-community-default-switch-9102-internal__ [target:65200:268444558 ]
    then accept
Term 9103:
    from community __vrf-community-default-switch-9103-internal__ [target:65200:268444559 ]
    then accept
Term 9104:
    from community __vrf-community-default-switch-9104-internal__ [target:65200:268444560 ]
    then accept
Term 9105:
    from community __vrf-community-default-switch-9105-internal__ [target:65200:268444561 ]
    then accept
Term 9106:
    from community __vrf-community-default-switch-9106-internal__ [target:65200:268444562 ]
    then accept
Term unnamed:
    from policy LEAF-IN
    then accept
Term unnamed:
    then reject

```

```

jnpr@SPINE-1> show policy __vrf-import-autoderive-default-switch-internal__
Policy __vrf-import-autoderive-default-switch-internal__:
Term 9100:
    from community __vrf-community-default-switch-9100-internal__ [target:65200:268444556 ]
    then accept
Term 9101:
    from community __vrf-community-default-switch-9101-internal__ [target:65200:268444557 ]
    then accept
Term 9102:
    from community __vrf-community-default-switch-9102-internal__ [target:65200:268444558 ]
    then accept

```

```

Term 9103:
  from community __vrf-community-default-switch-9103-internal__ [target:65200:268444559 ]
  then accept
Term 9104:
  from community __vrf-community-default-switch-9104-internal__ [target:65200:268444560 ]
  then accept
Term 9105:
  from community __vrf-community-default-switch-9105-internal__ [target:65200:268444561 ]
  then accept
Term 9106:
  from community __vrf-community-default-switch-9106-internal__ [target:65200:268444562 ]
  then accept
Term 9107:
  from community __vrf-community-default-switch-9107-internal__ [target:65200:268444563 ]
  then accept
Term 9108:
  from community __vrf-community-default-switch-9108-internal__ [target:65200:268444564 ]
  then accept
Term 9109:
  from community __vrf-community-default-switch-9109-internal__ [target:65200:268444565 ]
  then accept
Term unnamed:
  from policy LEAF-IN
  then accept
Term unnamed:
  then reject

```

<<< At the time of writing of this book, since global AS configuration is required to use auto-derived route-targets, overlay AS (65200) which is the same on all devices has been used

```

jnpr@LEAF-1> show configuration routing-options
router-id 100.0.0.23;
autonomous-system 65200;
forwarding-table {
  export pfe-ecmp;
}

```

```

jnpr@SPINE-1> show configuration routing-options
router-id 100.0.0.11;
autonomous-system 65200;
forwarding-table {
  export pfe-ecmp;
}

```

Manual defined route targets are in use, unless specified.

IRB Interface Configuration

As highlighted in previous sections, EVPN PEs capable of inter-subnet routing (Layer 3 gateways) use IRB interfaces to provide this functionality. To realize a distributed anycast gateway, IRB interfaces for a given VNI are distributed across all PEs. The `virtual-gateway-address` feature allows the creation of any anycast IP/MAC that is shared by all servicing IRB interfaces. To achieve the Layer 3 gateway functionality, IRB interface configuration options may or may not include the use of the `virtual-gateway-address` knob (please refer to the previous case study for added details):

```

jnpr@SPINE-1> show configuration interfaces irb unit 100
proxy-macip-advertisement;
description " * TENANT 10 - vlan 100 - vni 9100 ";
family inet {
  address 100.0.1.2/24 {
    virtual-gateway-address 100.0.1.1;    <<< All Layer 3 gateways (spine devices here) are
configured with the same anycast gateway IP for v4 and v6 (dual stack)
  }
}
virtual-gateway-v4-mac 00:00:00:00:10:00;
family inet6 {

```

```

address fe80::100:0:1:1/64;
address 8000::100:0:1:2/64 {
    virtual-gateway-address 8000::100:0:1:1; <<<
}
}
virtual-gateway-v6-mac 00:00:00:06:10:00;

```

Here are the recommendations for Layer 3 gateway configuration when routing is done at the leaf versus. spine.

Proxy-macip-advertisement:

- Routing at the leaf: Do not configure (Type 2 routes for both MAC and MAC+IP are advertised by the consolidated Layer 2/Layer 3 gateways and this knob is not needed in this case).
- Routing at the spine: Configure (This knob allows for the Layer 3 gateway to learn the MAC+IP binding via ARP/NDP and advertise Type 2 MAC+IP routes with BGP next-hop set to the original Layer 2 EVPN PE advertising Type 2 MAC route. Layer 3 PE IRB interface MACs should be reachable to avoid asymmetric forwarding during the ARP resolution process).

Virtual-gateway-v4-mac/virtual-gateway-v6-mac:

- Available since D63, these knobs are applicable when there is a Layer 2 switch between hosts and Layer 2 EVPN PEs and are used to prevent flooding. Without this knob, ARP resolution for the default gateway uses anycast/VRRP MAC (assuming VGA is configured) which is contained in the inner packet, while the source MAC address used on the outer frame is that of the IRB interface. As a result, when data traffic with destination MAC set to the VRRP/anycast MAC is received by the Layer 2 switch (for example, here Access-Switch-1/2) flood traffic is seen as the Layer 2 switch only learns the IRB MAC in the outer Ethernet header and never sees the anycast/VRRP MAC during the ARP resolution process. With this knob, the configured virtual gateway MAC is used in the inner packet (ARP resolution) as well as the outer Ethernet header. The Layer 2/Access switch learns the virtual gateway MAC and no flood traffic is seen when data traffic needs to be forwarded. Similar behavior is seen with the IPv6 neighbor discovery process and the `virtual-gateway-mac` knob can be used for v4 or v6 traffic.
- Without the `virtual-gateway-v4-mac` configured:
 - ARP reply from Layer 3 gateway(s) uses IRB interface address as the source MAC in the outer Ethernet header:

```

jnpr@SPINE-1# run show interfaces extensive irb
Physical interface: irb    , Enabled, Physical link is Up
<...>
Current address: 0c:86:10:70:3f:f2, Hardware address: 0c:86:10:70:3f:f2

jnpr@SPINE-2# run show interfaces extensive irb
Physical interface: irb    , Enabled, Physical link is Up
<...>
Current address: ec:3e:f7:47:b3:64, Hardware address: ec:3e:f7:47:b3:64

jnpr@SPINE-3# run show interfaces extensive irb
Physical interface: irb    , Enabled, Physical link is Up
<...>
Current address: 54:1e:56:2c:d7:6a, Hardware address: 54:1e:56:2c:d7:6a

jnpr@SPINE-4# run show interfaces extensive irb
Physical interface: irb    , Enabled, Physical link is Up
<...>
Current address: 54:1e:56:2c:b7:6a, Hardware address: 54:1e:56:2c:b7:6a

```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	00:00:00:88:5d:6d	ff:ff:ff:ff:ff:ff	ARP	46	who has 100.0.1.1? Tell 100.0.1.100
2	0.00094100	54:1e:56:2c:b7:6a	00:00:00:88:5d:6d	ARP	60	100.0.1.1 is at 00:00:5e:00:01:01
+ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0						
- Ethernet II, Src: 54:1e:56:2c:b7:6a (54:1e:56:2c:b7:6a), Dst: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
+ Destination: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
+ Source: 54:1e:56:2c:b7:6a (54:1e:56:2c:b7:6a)						
Type: 802.1Q Virtual LAN (0x8100)						
+ 802.1Q Virtual LAN						
- Address Resolution Protocol (reply)						
Hardware type: Ethernet (1)						
Protocol type: IP (0x0800)						
Hardware size: 6						
Protocol size: 4						
Opcode: reply (2)						
Sender MAC address: 00:00:5e:00:01:01 (00:00:5e:00:01:01)						
Sender IP address: 100.0.1.1 (100.0.1.1)						
Target MAC address: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
Target IP address: 100.0.1.100 (100.0.1.100)						

- Only the host learns the anycast/VRRP MAC during ARP resolution—the access switch does not—and hence flooding occurs when data traffic needs to be forwarded (DMAC = anycast/VRRP MAC):

```

jnpr@Access-SW-1> show ethernet-switching table vlan-id 100
Vlan name      MAC address    MAC flags    Age    Logical interface    NH Index    RTR ID
v100           0c:86:10:70:3f:f2    D            -      ae0.0                0            0
v100           54:1e:56:2c:b7:6a    D            -      ae0.0                0            0
v100           54:1e:56:2c:d7:6a    D            -      ae0.0                0            0
v100           ec:3e:f7:47:b3:64    D            -      ae0.0                0            0

```

- With the virtual-gateway-v4-mac configured:

- ARP reply from Layer 3 gateway(s) uses virtual gateway MAC address as the source MAC in the outer Ethernet header.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	00:00:00:88:5d:6d	ff:ff:ff:ff:ff:ff	ARP	46	who has 100.0.1.1? Tell 100.0.1.100
2	0.00081500	00:00:00:00:10:00	00:00:00:88:5d:6d	ARP	60	100.0.1.1 is at 00:00:00:00:10:00
+ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0						
- Ethernet II, Src: 00:00:00:00:10:00 (00:00:00:00:10:00), Dst: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
+ Destination: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
+ Source: 00:00:00:00:10:00 (00:00:00:00:10:00)						
Type: 802.1Q Virtual LAN (0x8100)						
+ 802.1Q Virtual LAN						
- Address Resolution Protocol (reply)						
Hardware type: Ethernet (1)						
Protocol type: IP (0x0800)						
Hardware size: 6						
Protocol size: 4						
Opcode: reply (2)						
Sender MAC address: 00:00:00:00:10:00 (00:00:00:00:10:00)						
Sender IP address: 100.0.1.1 (100.0.1.1)						
Target MAC address: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)						
Target IP address: 100.0.1.100 (100.0.1.100)						

- Both host and access switch learn the virtual gateway MAC during ARP resolution, hence no flooding occurs when data traffic needs to be forwarded (DMAC = virtual-gateway-v4-mac).

```
jnpr@Access-SW-1> show ethernet-switching table vlan-id 100
```

Vlan name	MAC address	MAC flags	Age	Logical interface	NH Index	RTR ID
v100	00:00:00:00:10:00	D	-	ae0.0	0	0
v100	00:00:00:88:5d:6d	D	-	xe-0/0/6.0	0	0

■ Similar behavior is seen for IPv6 with virtual-gateway-v6-mac configured:

```
No.    Time    Source          Destination      Protocol Length Info
1 0.00000000 8000::100:0:1:100 ff02::1:ff01:1 ICMPv6 90 Neighbor Solicitation for 8000::100:0:1:1 from 00:00:00:88:5d:6d
2 0.00085500 8000::100:0:1:4 8000::100:0:1:100 ICMPv6 90 Neighbor Advertisement 8000::100:0:1:1 (rtr, sol, ovr) is at 00:00:00:06:10:00

Frame 2: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
Ethernet II, Src: 00:00:00:06:10:00 (00:00:00:06:10:00), Dst: 00:00:00:88:5d:6d (00:00:00:88:5d:6d)
802.1Q Virtual LAN
Internet Protocol Version 6, Src: 8000::100:0:1:4 (8000::100:0:1:4), Dst: 8000::100:0:1:100 (8000::100:0:1:100)
Internet Control Message Protocol v6
Type: Neighbor Advertisement (136)
Code: 0
Checksum: 0x0194 [correct]
Flags: 0xe0000000
Target: 8000::100:0:1:1 (8000::100:0:1:1)
ICMPv6 Option (Target link-layer address : 00:00:00:06:10:00)
Type: Target link-layer address (2)
Length: 1 (8 bytes)
Link-layer address: 00:00:00:06:10:00 (00:00:00:06:10:00)
```

```
jnpr@Access-SW-1> show ethernet-switching table vlan-id 100
```

Vlan name	MAC address	MAC flags	Age	Logical interface	NH Index	RTR ID
v100	00:00:00:06:10:00	D	-	ae0.0	0	0
v100	00:00:00:88:5d:6d	D	-	xe-0/0/6.0	0	0

IPv6 support:

- Available in D63, to support IPv6 traffic between hosts, IRB interfaces on all Layer 3 gateways (spine devices here) are configured with an IPv6 virtual gateway address (in addition to IPv4). The same link local IP address is configured on all PEs so that any packet destined to the link-local IP can be intercepted for NDP processing on any of the PEs. IPv6 addresses are only configured on the IRB interfaces (no other IPv6 configuration is required in the underlay or overlay to support the same).
- The capture here shows IPv6 data traffic between hosts transported over the EVPN-VXLAN fabric encapsulated in IPv4 outer header:

```
No.    Time    Source          Destination      Protocol Length Info
1 0.00000000 8000::100:0:1:100 8009::100:0:10:102 IPv6 1046 IPv6 no next header
2 0.00100000 8000::100:0:1:100 8009::100:0:10:102 IPv6 1046 IPv6 no next header
3 0.00200000 8000::100:0:1:100 8009::100:0:10:102 IPv6 1046 IPv6 no next header
4 0.00200000 8000::100:0:1:100 8009::100:0:10:102 IPv6 1046 IPv6 no next header

Frame 1: 1046 bytes on wire (8368 bits), 1046 bytes captured (8368 bits)
Ethernet II, Src: JuniperN 49:29:33 (64:64:9b:49:29:33), Dst: JuniperN 70:40:05 (0c:86:10:70:40:05)
Internet Protocol Version 4, Src: 100.0.0.23, Dst: 100.0.0.11
User Datagram Protocol, Src Port: 58904, Dst Port: 4789
Virtual eXtensible Local Area Network
Ethernet II, Src: 00:00:00 88:5d:6d (00:00:00:88:5d:6d), Dst: 00:00:00 06:10:00 (00:00:00:06:10:00)
Internet Protocol Version 6, Src: 8000::100:0:1:100, Dst: 8009::100:0:10:102
Data (938 bytes)
```

EVPN Protocol Configuration

Under the global 'protocols evpn' hierarchy, the extended-vni-list statement lists all advertised VNIs that will be part of the EVPN domain, which in this case are all configured VNIs. Since manual RTs are used, vni-options specifies the individual RTs for each VNI. Since ingress-replication is supported by default, the multicast-mode ingress-replication statement to define the same is not needed.

Summarized in Table 3.2 are recommendations on the application of default-gateway knobs used at this hierarchy when routing is done at leaf versus spine, and are outlined below without or with the virtual-gateway-address knob, respectively.

Table 3.2 Application of the Default-Gateway Knobs

	Routing on leaf (same IP/same MAC on all Layer 3 gateways)	Routing on spine (VGA used on all Layer 3 gateways)
default-gateway do-not-advertise	Required (All of the EVPN PEs are configured with the same IP/MAC and hence default-gateway synchronization across PEs is not needed. This knob suppresses the advertisement for IRB Type 2 MAC routes).	Not required (Configuring this knob allows Layer 2 PEs to learn the Layer 3 gateway and have Layer 3 PE's IRB interface address reachability to help with symmetric forwarding).
default-gateway no-gateway-community	Not required (Does not really matter since same IP/MAC is used across all EVPN PEs).	Required (With this knob configured, Layer 2 PEs don't try to proxy for Layer 3 gateway IRB and VGA MACs, will just forward traffic destined to these MACs).

As applicable to this case study, since routing is done on the spine (using VGA) default-gateway no-gateway-community has been configured under the 'protocols evpn' hierarchy on the Layer 3 gateway.

```

jnpr@LEAF-1> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list [ 9100 9101 9102 9103 9105 9106 ];
vni-options {
vni 9100 {
    vrf-target export target:1:9100;
}
vni 9101 {
    vrf-target export target:1:9101;
}
vni 9102 {
    vrf-target export target:1:9102;
}
vni 9103 {
    vrf-target export target:1:9103;
}
vni 9105 {
    vrf-target export target:1:9105;
}
vni 9106 {
    vrf-target export target:1:9106;
}
}

```

<<< Specific route targets used by EVPN routes for given VNI

```

jnpr@SPINE-1> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list [ 9100 9101 9102 9103 9105 9106 9108 9109 ];
default-gateway no-gateway-community;
vni-options {
vni 9100 {
    vrf-target export target:1:9100;
}
vni 9101 {
    vrf-target export target:1:9101;
}
vni 9102 {
    vrf-target export target:1:9102;
}

```

<<< With routing being done at the spine, this knob ensures that Layer 2 PEs (leaf devices) do not proxy for Layer 3 gateway

```

}
vni 9103 {
    vrf-target export target:1:9103;
}
vni 9105 {
    vrf-target export target:1:9105;
}
vni 9106 {
    vrf-target export target:1:9106;
}
vni 9108 {
    vrf-target export target:1:9108;
}
vni 9109 {
    vrf-target export target:1:9109;
}
}

```

IP-VRF Configuration

A tenant IP-VRF is created for IP routes on a EVPN PE and could be associated with one or more EVIs. The IRB interface, which is the Layer 3 interface for a bridge-domain, connects the MAC-VRF and the IP-VRF on an EVPN PE. Only subnets within a VRF can communicate with each other, allowing for multi-tenancy. Under the 'routing-instances' hierarchy, the instance-type vrf statement creates an IP-VRF. In this use case, all configured VNIs are part of a single tenant (VRF_TENANT_MAIN) that contains all IRB interfaces. Here, Layer 3 gateways advertise Type 2 routes (MAC+IP since proxy-macip-advertisement is configured) and hence host IP routes are learned through Type 2 routes. These use the VNI RTs as previously discussed and not IP-VRF RT.

```

jnpr@SPINE-1> show configuration routing-instances
VRF_TENANT_MAIN {
    <<< Shared IP VRF across all spine nodes (required on Layer 3 gateways
only)

    instance-type vrf;
    interface irb.100;
    interface irb.101;
    interface irb.102;
    interface irb.103;
    interface irb.105;
    interface irb.106;
    interface irb.108;
    interface irb.109;
    interface lo0.10;
    <<< Recommendation would be to configure lo0 unit when VXLAN VNI irb is
placed in the tenant IP-VRF for successful ARP resolution
    route-distinguisher 100.0.0.11:10;
    vrf-target target:10:10;
}

```

Access

In the use case, end hosts (VMs or BMSs) are simulated by the IXIA test tool connected to the leaf devices using the access switches in POD-1 and POD-2. CE-1 and CE-2 simulates hosts connected to Access Switch-1 and Access Switch-2 respectively, and are multi-homed to redundant TORs LEAF-1 and LEAF-2 in POD-1. CE-3 simulates hosts connected to Access Switch-2 and multi-homed to LEAF-3, LEAF-4, LEAF-5, and LEAF-6 to demonstrate no change in functional behavior with N-way multi-homing. Each CE device views the set of redundant leaf devices as one device by virtue of the same system ID on the LAG interface configured on the multi-homed PEs. All-active multi-homing has been configured on the leaf devices. The LAG interface between PEs and CEs is a trunk interface carrying multiple VLANs. Previous steps map the configured VLANs to VNIs, which exist in a global virtual

switch. The ESI on LAG interface is configured only on the PE devices. ESI (Type 0) is in use as indicated by the first byte (set to 0) of the 10-byte ESI value. The remaining nine bytes are statically defined. The same ESI value is configured on PEs multi-homed to the same ES. Interface hold timers have been configured on LAG members to prevent traffic black holing when there is a failure, providing sufficient time for control plane to converge.

NOTE At the time this book was written, the LACP bring-down on the core isolation feature was not available because of which interface timers have been used.

```
jnpr@LEAF-1> show configuration interfaces ae1
apply-groups-except MTU-VXLAN;
description "ae1(ESI-A) to Access-Sw-1";
mtu 9192;
esi {
    00:33:33:33:33:33:33:33:33:33; <<< LEAF-1 and LEAF-2 (all-
    active) share the same ESI-A (00:33:33:33:33:33:33:33:33:33) on LAG ae1
    all-active;
}
aggregated-ether-options {
    lacp { <<< Same LACP System-ID on both LEAF-1 and LEAF-
    2 so access switch thinks it is connected to a single device
    active;
        system-id 00:00:00:00:00:03;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ bd9100 bd9101 bd9102 bd9103 bd9105 bd9106 ];
        }
        filter {
            input MF-classfier;
        }
    }
}
```

```
jnpr@LEAF-1> show configuration interfaces xe-0/0/1
apply-groups-except MTU-VXLAN;
hold-time up 180000 down 5;
ether-options {
    802.3ad ae1;
}
```

```
jnpr@LEAF-1> show configuration interfaces ae0
apply-groups-except MTU-VXLAN;
description "ae0(ESI-B) to Access-Sw-2";
mtu 9192;
esi {
    00:22:22:22:22:22:22:22:22:22;
    all-active; <<< LEAF-1 and LEAF-2 (all-
    active) share the same ESI-B (00:22:22:22:22:22:22:22:22:22) on LAG ae0
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:00:02;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ bd9100 bd9101 bd9102 bd9103 ];
        }
        filter {
            input MF-classfier;
        }
    }
}
```

```

    }
}
jnpr@LEAF-1> show configuration interfaces xe-0/0/0
apply-groups-except MTU-VXLAN;
hold-time up 180000 down 5;
ether-options {
    802.3ad ae0;
}

jnpr@LEAF-3> show configuration interfaces ae2
apply-groups-except MTU-VXLAN;
mtu 9192;
esi {
    00:44:44:44:44:44:44:44;
    all-active;
}
aggregated-ether-options {
    lacp {
        active; <<< LEAF-3,4,5,6 (all-
active) share the same ESI-C (00:44:44:44:44:44:44:44) on LAG ae2
system-id 00:00:00:00:00:01;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ bd9105 bd9106 bd9107 bd9108 bd9109 bd5080 bd5091 bd1880 bd6099 ];
        }
    }
}
jnpr@LEAF-3> show configuration interfaces xe-0/0/0
apply-groups-except MTU-VXLAN;
hold-time up 180000 down 5;
ether-options {
    802.3ad ae2;
}

```

Verification

CE-3 is multihomed to 4 leaf devices in POD-2:

```

jnpr@Access-SW-2> show lldp neighbors | match ae2
xe-0/0/47      ae2      10:0e:7e:bb:1e:40   xe-0/0/47      LEAF-6
xe-0/0/46      ae2      10:0e:7e:bb:f1:80   xe-0/0/46      LEAF-5
xe-0/0/5       ae2      54:4b:8c:ff:c2:20   xe-0/0/0       LEAF-4
xe-0/0/4       ae2      ec:3e:f7:00:23:00   xe-0/0/0       LEAF-3

```

LEAF-1 and LEAF-2 learn remote MAC addresses of hosts in POD-2 as well as local hosts in POD-1. MAC addresses for remote hosts are learned from the remote VTEPs LEAF-3,4,5,6 in POD-2, that advertise Type 2 routes for the same.

```

jnpr@LEAF-1> show ethernet-switching table vlan-id 100
Vlan name  MAC address      MAC flags  Logical interface  Active source
bd9100     00:00:00:00:10:00 DR          esi.1734         05:00:00:00:00:00:23:8c:00
bd9100     00:00:00:00:10:01 D           vtep.32770       100.0.0.13
bd9100     00:00:00:06:10:00 DR          esi.1734         05:00:00:00:00:00:00:23:8c:00
bd9100     00:00:00:2e:80:c3 DL          ae0.0            <<< Host on ESI-B learned locally
bd9100     00:00:00:88:5d:6d DLR         ae1.0            <<< Host on ESI-A learned locally
bd9100     00:00:00:00:01:01 DR          esi.1734         05:00:00:00:00:00:00:23:8c:00
bd9100     00:00:05e:00:02:01 DR          esi.1734         05:00:00:00:00:00:00:23:8c:00
bd9100     0c:86:10:70:3f:f2 D           vtep.32773       100.0.0.11
bd9100     54:1e:56:2c:b7:6a D           vtep.32772       100.0.0.14
bd9100     54:1e:56:2c:d7:6a D           vtep.32770       100.0.0.13
bd9100     ec:3e:f7:47:b3:64 D           vtep.32771       100.0.0.12

```

```
jnpr@LEAF-1> show ethernet-switching table vlan-id 105
Vlan name      MAC address      MAC flags  Logical interface  Active source
bd9105         00:00:00:00:15:00 DR          esi.1743         05:00:00:00:00:00:23:91:00
bd9105         00:00:00:06:15:00 DR          esi.1743         05:00:00:00:00:00:23:91:00
bd9105         00:00:00:b6:70:f4 DLR          ae1.0          <<< Host on ESI-A learned locally
bd9105         00:00:00:b6:e9:10 DR          esi.1808         00:44:44:44:44:44:44:44
bd9105         0c:86:10:70:3f:f2 D           vtep.32773      100.0.0.11
bd9105         54:1e:56:2c:b7:6a D           vtep.32772      100.0.0.14
bd9105         54:1e:56:2c:d7:6a D           vtep.32770      100.0.0.13
bd9105         ec:3e:f7:47:b3:64 D           vtep.32771      100.0.0.12
```

<<< Host on ESI-C is learned via esi NH (esi.1808) that resolves to remote VTEPs in POD-2

```
jnpr@LEAF-1> show ethernet-switching vxlan-tunnel-end-point esi
ESI            RTT            VLNBH  INH      ESI-IFL  LOC-IFL  #RVTEPs
00:44:44:44:44:44:44:44 default-switch 1808 131092 esi.1808 ae2.0    4
RVTEP-IP      RVTEP-IFL  VENH    MASK-ID  FLAGS
100.0.0.28    vtep.32777 1811    3        2
100.0.0.27    vtep.32776 1810    2        2
100.0.0.26    vtep.32775 1809    1        2
100.0.0.25    vtep.32774 1807    0        2
```

Inter-subnet traffic between PODs is routed on spine devices that act as distributed Layer 3 gateways. For example, inter-subnet traffic from VLAN 100 (on TORs of POD-1 only) to VLAN 109 (on TORs of POD-2 only), LEAF-1 and LEAF-2 forward traffic to default Layer 3 gateway for VLAN 100 distributed on all spine devices, which in turn can route traffic to the desired destination host on VLAN 109.

```
jnpr@LEAF-1> show ethernet-switching table vlan-id 100
Vlan name      MAC address      MAC flags  Logical interface  Active source
bd9100         00:00:00:00:10:00 DR          esi.1734         05:00:00:00:00:00:23:8c:00
bd9100         00:00:00:00:10:01 D           vtep.32770      100.0.0.13
bd9100         00:00:00:06:10:00 DR          esi.1734         05:00:00:00:00:00:23:8c:00
bd9100         00:00:00:2e:80:c3 DL           ae0.0
bd9100         00:00:00:88:5d:6d DLR          ae1.0
bd9100         00:00:5e:00:01:01 DR          esi.1734         05:00:00:00:00:00:23:8c:00
bd9100         00:00:5e:00:02:01 DR          esi.1734         05:00:00:00:00:00:23:8c:00
bd9100         0c:86:10:70:3f:f2 D           vtep.32773      100.0.0.11
bd9100         54:1e:56:2c:b7:6a D           vtep.32772      100.0.0.14
bd9100         54:1e:56:2c:d7:6a D           vtep.32770      100.0.0.13
bd9100         ec:3e:f7:47:b3:64 D           vtep.32771      100.0.0.12
```

>>> Default Layer 3 gateway for VLAN 100 is distributed on all spine devices (multi-homed to the same ESI)

```
jnpr@LEAF-1> show ethernet-switching vxlan-tunnel-end-point esi
ESI            RTT            VLNBH  INH      ESI-IFL  LOC-IFL  #RVTEPs
05:00:00:00:00:00:00:23:8c:00 default-switch 1734 131075 esi.1734
RVTEP-IP      RVTEP-IFL  VENH    MASK-ID  FLAGS
100.0.0.14    vtep.32772 1815    3        2
100.0.0.13    vtep.32770 1800    2        2
100.0.0.11    vtep.32773 1804    1        2
100.0.0.12    vtep.32771 1803    0        2
```

>>> Spine devices on POD-1 learn the remote host on POD-2 through generated Type 2 routes

```
jnpr@SPINE-1> show route table bgp.evpn.0 | match 78:f5
2:100.0.0.25:1::9109::00:00:01:70:78:f5/304
2:100.0.0.25:1::9109::00:00:01:70:78:f5::100.0.10.102/304

jnpr@SPINE-1> show route table bgp.evpn.0 detail | find 78:f5
2:100.0.0.25:1::9109::00:00:01:70:78:f5/304 (2 entries, 1 announced)
  *BGP      Preference: 170/-101
            Route Distinguisher: 100.0.0.25:1
            Next hop type: Indirect, Next hop index: 0
            Address: 0xa5cc7d0
            Next-hop reference count: 66
            Source: 100.0.0.1
            Protocol next hop: 100.0.0.25
Peer AS: 65200
  Age: 1d 21:34:40      Metric2: 0
```

```

Validation State: unverified
Task: BGP_65200_65200.100.0.0.1
Announcement bits (1): 1-BGP_RT_Background
AS path: I (Originator)
Cluster list: 1.1.1.1 3.3.3.3
Originator ID: 100.0.0.25
Communities: target:1:9109 encapsulation0:0:0:0:vxlan
Import Accepted
Route Label: 9109
ESI: 00:44:44:44:44:44:44:44:44
Localpref: 100
Router ID: 100.0.0.1

```

>>> Spine devices on POD-1 have relevant information to perform MAC rewrite (asymmetric forwarding)

```

jnpr@SPINE-1> show route forwarding-table destination 100.0.10.102
Routing table: VRF_TENANT_MAIN.inet
Internet:
Destination      Type RtRef Next hop          Type Index  NhRef Netif
100.0.10.102/32  dest   0 45:0:0:32:0:0    ucst 2092    1
!
Interface:      irb.109
NH Addr: 100.0.10.102
Local Addr: 100.0.10.2
rewrite: 00:00:01:70:78:f5

```

```

jnpr@SPINE-1> show ethernet-switching table vlan-id 109
Vlan name      MAC address      MAC flags Logical interface  Active source
bd9109         00:00:00:00:19:00 DR,SD            esi.1908          05:00:00:00:00:00:23:95:00
bd9109         00:00:00:06:19:00 DR,SD            esi.1908          05:00:00:00:00:00:23:95:00
bd9109         00:00:01:70:78:f5 DR                esi.2086          00:44:44:44:44:44:44:44
bd9109         54:1e:56:2c:b7:6a D                 vtep.32773        100.0.0.14
bd9109         54:1e:56:2c:d7:6a D                 vtep.32772        100.0.0.13
bd9109         ec:3e:f7:47:b3:64 D                 vtep.32771        100.0.0.12

```

```

jnpr@SPINE-1> show ethernet-switching vxlan-tunnel-end-point esi
ESI              RTT              VLNBH INH      ESI-IFL  LOC-IFL  #RVTEPs
00:44:44:44:44:44:44:44 default-switch 2086 2097172 esi.2086 4
RVTEP-IP      RVTEP-IFL      VENH      MASK-ID      FLAGS
100.0.0.28      vtep.32777      2091       3             2
100.0.0.27      vtep.32776      2040       2             2
100.0.0.26      vtep.32774      1885       1             2
100.0.0.25      vtep.32770      1877       0             2

```

Traffic Flows

For this use case, both PODs are situated in the same data centers (Intra DC) and all hosts belong to the same tenant (Intra VRF). All hosts share the same IP-VRF (VRF_TENANT_1).

CE-1 hosts multi-homed to LEAF-1 and LEAF-2 (ESI-A = 00:33:33:33:33:33:33:33:33):

```

Host 1-1 (H1-
1) : VLAN 100 <> VNI 9100, IPv4 = 100.0.1.100, IPv6 = 8000::100:0:1:100, MAC = 00:00:00:88:5d:6d
Host 1-2 (H1-
2) : VLAN 101 <> VNI 9101, IPv4 = 100.0.2.100, IPv6 = 8001::100:0:2:100, MAC = 00:00:00:9a:d4:90
Host 1-3 (H1-
3) : VLAN 102 <> VNI 9102, IPv4 = 100.0.3.100, IPv6 = 8002::100:0:3:100, MAC = 00:00:00:b6:70:ee
Host 1-4 (H1-
4) : VLAN 105 <> VNI 9105, IPv4 = 100.0.6.100, IPv6 = 8005::100:0:6:100, MAC = 00:00:00:b6:70:f4
Host 1-5 (H1-
5) : VLAN 106 <> VNI 9106, IPv4 = 100.0.7.100, IPv6 = 8006::100:0:7:100, MAC = 00:00:00:b6:70:f6

```

CE-2 hosts multi-homed to LEAF-1 and LEAF-2 (ESI-B = 00:22:22:22:22:22:22:22:22):

```

Host 2-1 (H2-
1) : VLAN 100 <> VNI 9100, IPv4 = 100.0.1.101, IPv6 = 8000::100:0:1:101, MAC = 00:00:00:2e:80:c3
Host 2-2 (H2-
2) : VLAN 101 <> VNI 9101, IPv4 = 100.0.2.101, IPv6 = 8001::100:0:2:101, MAC = 00:00:00:9a:d4:91
Host 2-3 (H2-

```

```

3) : VLAN 103 <> VNI 9103, IPv4 = 100.0.4.101, IPv6 = 8003::100:0:4:101, MAC = 00:00:00:b6:e9:0c

CE-3 hosts multi-homed to LEAF-3, LEAF-4, LEAF-5, LEAF-6 (ESI-C = 00:44:44:44:44:44:44:44:44):
Host 3-1 (H3-
1) : VLAN 105 <> VNI 9105,IPv4 = 100.0.6.101, IPv6 = 8005::100:0:6:101, MAC = 00:00:00:b6:e9:10
Host 3-2 (H3-
2) : VLAN 106 <> VNI 9106,IPv4 = 100.0.7.101, IPv6 = 8006::100:0:7:101, MAC = 00:00:00:b6:e9:12
Host 3-3 (H3-
3) : VLAN 108 <> VNI 9108,IPv4 = 100.0.9.102, IPv6 = 8008::100:0:9:102, MAC = 00:00:01:70:78:f3
Host 3-4 (H3-
4) : VLAN 109 <> VNI 9109,IPv4 = 100.0.10.102,IPv6 = 8009::100:0:10:102,MAC = 00:00:01:70:78:f5

```

```

Default gateway : IPv4 = 100.0.1.1, IPv6 = 8000::100:0:1:1
                  (virtual-gateway-address for irb.100 Layer 3 interface for VLAN 100)
                  : IPv4 = 100.0.2.1, IPv6 = 8001::100:0:2:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 101)
                  : IPv4 = 100.0.3.1, IPv6 = 8002::100:0:3:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 102)
                  : IPv4 = 100.0.4.1, IPv6 = 8003::100:0:4:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 103)
                  : IPv4 = 100.0.6.1, IPv6 = 8005::100:0:6:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 105)
                  : IPv4 = 100.0.7.1, IPv6 = 8006::100:0:7:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 106)
                  : IPv4 = 100.0.9.1, IPv6 = 8008::100:0:9:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 108)
                  : IPv4 = 100.0.10.1, IPv6 = 8009::100:0:10:1
(virtual-gateway-address for irb.101 Layer 3 interface for VLAN 109)

```

Flows demonstrated here are listed here:

Intra-POD, Intra-Rack, Intra-Subnet/Layer 2 (ESI A <> ESI B)

```

VLAN 100 <> 100: IPv4 100.0.1.100 <> 100.0.1.101
VLAN 101 <> 101: IPv4 100.0.2.100 <> 100.0.2.101
VLAN 100 <> 100: IPv6 8000::100:0:1:100 <> 8000::100:0:1:101
VLAN 101 <> 101: IPv6 8001::100:0:2:100 <> 8001::100:0:2:101

```

Intra-POD, Intra-Rack, Inter-Subnet/Layer 3 (ESI A <> ESI B)

```

VLAN 102 <> 103: IPv4 100.0.3.100 <> 100.0.4.101
VLAN 100 <> 101: IPv4 100.0.1.100 <> 100.0.2.101
VLAN 100 <> 101: IPv6 8000::100:0:1:100 <> 8001::100:0:2:101
VLAN 102 <> 103: IPv6 8002::100:0:3:100 <> 8003::100:0:4:101

```

Inter-POD, Inter-Rack, Intra-Subnet/Layer 2 (ESI A <> ESI C)

```

VLAN 105 <> 105: IPv4 100.0.6.100 <> 100.0.6.101
VLAN 106 <> 106: IPv4 100.0.7.100 <> 100.0.7.101
VLAN 105 <> 105: IPv6 8005::100:0:6:100 <> 8005::100:0:6:101
VLAN 106 <> 106: IPv6 8006::100:0:7:100 <> 8006::100:0:7:101

```

Inter-POD, Inter-Rack, Inter-Subnet/Layer 3 (ESI A <> ESI C)

```

VLAN 100 <> 109: IPv4 100.0.1.100 <> 100.0.10.102
VLAN 102 <> 108: IPv4 100.0.3.100 <> 100.0.9.102
VLAN 100 <> 109: IPv6 8000::100:0:1:100 <> 8009::100:0:10:102
VLAN 102 <> 108: IPv6 8002::100:0:3:100 <> 8008::100:0:9:102

```

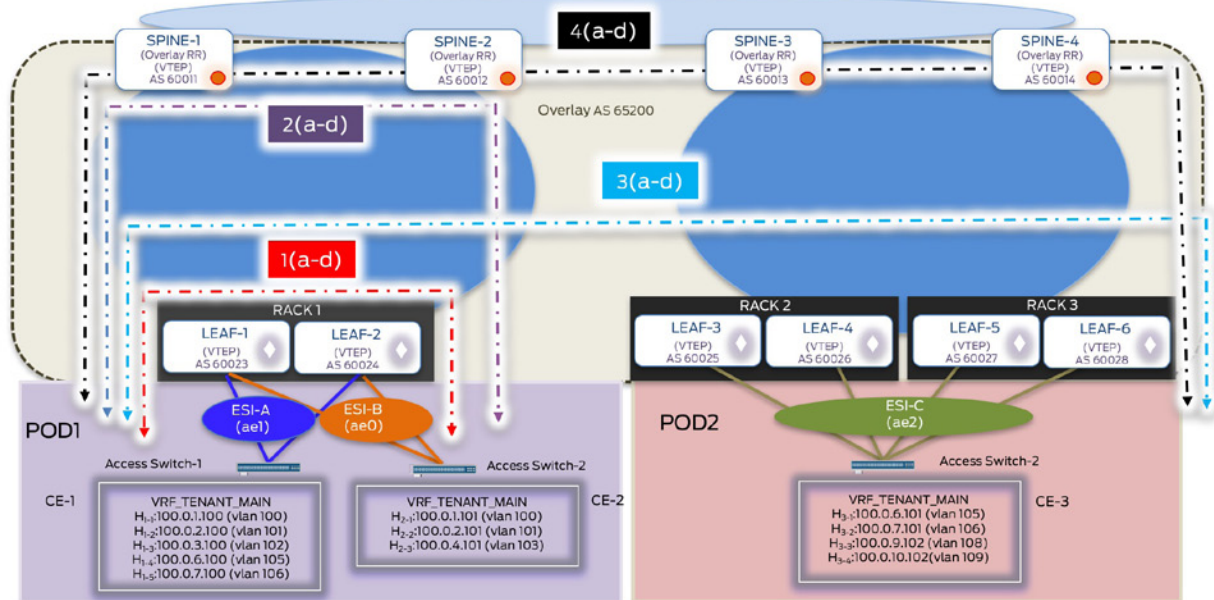


Figure 3.5 Traffic Flows

IXIA Statistics – Intra DC Intra-VRF (Flows 1(a-d), 2(a-d), 3(a-d), 4(a-d))

(1000 flows @ 1000 pps)

Transmit State	Traffic Item Name	Enabled	Flow Groups	Tx Ports	Rx Ports	Endpoint/Encapsulation Sets
1	1a_Intra-VRF: vlan 100 <-> 100: 100.0.1.100 <-> 100.0.1.101 (A-B)	✓	2	2	2	2
2	1b_Intra-VRF: vlan 101 <-> 100: 100.0.2.100 <-> 100.0.2.101 (A-B)	✓	2	2	2	2
3	1c_Intra-VRF-SPV: vlan 100 <-> 100: 8000::100:d:1:100 <-> 8000::100:d:1:101 (A-B)	✓	2	2	2	2
4	1d_Intra-VRF-SPV: vlan 101 <-> 101: 8001::100:d:2:100 <-> 8001::100:d:2:101 (A-B)	✓	2	2	2	2
5	2a_Intra-VRF: vlan 102 <-> 103: 100.0.3.100 <-> 100.0.4.101 (A-B)	✓	2	2	2	2
6	2b_Intra-VRF: vlan 100 <-> 101: 100.0.1.100 <-> 100.0.2.101 (A-B)	✓	2	2	2	2
7	2c_Intra-VRF-SPV: vlan 100 <-> 101: 8000::100:d:1:100 <-> 8000::100:d:2:101 (A-B)	✓	2	2	2	2
8	2d_Intra-VRF-SPV: vlan 102 <-> 103: 8002::100:d:3:100 <-> 8002::100:d:4:101 (A-B)	✓	2	2	2	2
9	3a_Intra-VRF: vlan 105 <-> 106: 100.0.6.100 <-> 100.0.7.101 (A-C)	✓	2	2	2	2
10	3b_Intra-VRF: vlan 106 <-> 106: 100.0.7.100 <-> 100.0.7.101 (A-C)	✓	2	2	2	2
11	3c_Intra-VRF-SPV: vlan 105 <-> 106: 8005::100:d:5:100 <-> 8005::100:d:6:101 (A-C)	✓	2	2	2	2
12	3d_Intra-VRF-SPV: vlan 106 <-> 106: 8006::100:d:7:100 <-> 8006::100:d:7:101 (A-C)	✓	2	2	2	2
13	4a_Intra-VRF: vlan 100 <-> 109: 101.0.0.1.100 <-> 100.0.9.102 (A-C)	✓	2	2	2	2
14	4b_Intra-VRF: vlan 102 <-> 108: 101.0.0.3.100 <-> 100.0.9.102 (A-C)	✓	2	2	2	2
15	4c_Intra-VRF-SPV: vlan 100 <-> 109: 8000::100:d:1:100 <-> 8009::100:d:10:102 (A-C)	✓	2	2	2	2
16	4d_Intra-VRF-SPV: vlan 102 <-> 108: 8002::100:d:3:100 <-> 8008::100:d:9:102 (A-C)	✓	2	2	2	2

Traffic Item	Tx Frames	Rx Frames	Tx L1 Rate (pps)	Rx L1 Rate (pps)	Frames Delta	Loss %	Packet Loss Duration (ms)	Store-Forward Avg Latency (ms)	Tx Frame Rate	Rx Frame Rate
1 1a_Intra-VRF: vlan 100 <-> 100: 100.0.1.100 <-> 100.0.1.101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	3.857	2,000.000	2,000.000
2 1b_Intra-VRF: vlan 101 <-> 100: 100.0.2.100 <-> 100.0.2.101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	3.857	2,000.000	2,000.000
3 1c_Intra-VRF-SPV: vlan 100 <-> 100: 8000::100:d:1:100 <-> 8000::100:d:1:101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	3.950	2,000.000	2,000.000
4 1d_Intra-VRF-SPV: vlan 101 <-> 101: 8001::100:d:2:100 <-> 8001::100:d:2:101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	3.956	2,000.000	2,000.000
5 2a_Intra-VRF: vlan 102 <-> 103: 100.0.3.100 <-> 100.0.4.101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	20.610	2,000.000	2,000.000
6 2b_Intra-VRF: vlan 100 <-> 101: 100.0.1.100 <-> 100.0.2.101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	20.828	2,000.000	2,000.000
7 2c_Intra-VRF-SPV: vlan 100 <-> 101: 8000::100:d:1:100 <-> 8001::100:d:2:101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	8.350	2,000.000	2,000.000
8 2d_Intra-VRF-SPV: vlan 102 <-> 103: 8002::100:d:3:100 <-> 8003::100:d:4:101 (A-B)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	17.434	2,000.000	2,000.000
9 3a_Intra-VRF: vlan 105 <-> 106: 100.0.6.100 <-> 100.0.7.101 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	19.960	2,000.000	2,000.000
10 3b_Intra-VRF: vlan 106 <-> 106: 100.0.7.100 <-> 100.0.7.101 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	20.211	2,000.000	2,000.000
11 3c_Intra-VRF-SPV: vlan 105 <-> 106: 8005::100:d:5:100 <-> 8005::100:d:6:101 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	19.903	2,000.000	2,000.000
12 3d_Intra-VRF-SPV: vlan 106 <-> 106: 8006::100:d:7:100 <-> 8006::100:d:7:101 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	20.019	2,000.000	2,000.000
13 4a_Intra-VRF: vlan 100 <-> 109: 101.0.0.1.100 <-> 100.0.9.102 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	21.842	2,000.000	2,000.000
14 4b_Intra-VRF: vlan 102 <-> 108: 101.0.0.3.100 <-> 100.0.9.102 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	20.206	2,000.000	2,000.000
15 4c_Intra-VRF-SPV: vlan 100 <-> 109: 8000::100:d:1:100 <-> 8009::100:d:10:102 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	23.786	2,000.000	2,000.000
16 4d_Intra-VRF-SPV: vlan 102 <-> 108: 8002::100:d:3:100 <-> 8008::100:d:9:102 (A-C)	3,782,890	3,782,890	16,320,000.000	16,320,000.000	0	0.000	0.000	19.344	2,000.000	2,000.000

Manual Service Chaining of East-West Traffic

(Level 3 Security Insertion: Inter-tenant Inter-subnet):

For traffic between hosts in different tenants on separate PODs, manual service chaining is demonstrated by hair pinning traffic through the local DC firewall for policy enforcement before forwarding it to the intended destination. In this use case, the DC firewall (SRX Series) is connected to each border leaf device. As elaborated previously, service chaining is treated as an IP path problem by getting traffic to the firewall. Details on firewall rules and policy enforcement are outside the scope of this book.

Service node: refers to the fabric/super-spine devices that service each POD by directing intended traffic from the IP fabric to the service block (firewall).

Service block: refers to the firewall that is responsible for service chaining inter-tenant traffic.

The following sections in this chapter elaborate upon the control plane and data plane details used to achieve the Layer 3 security insertion for EW inter-tenant traffic.

Configuration

All configuration details as discussed in the previous sections remain the same. Figure 3.6 highlights the logical topology on how the firewall is inserted in the DC to process traffic between different tenants.

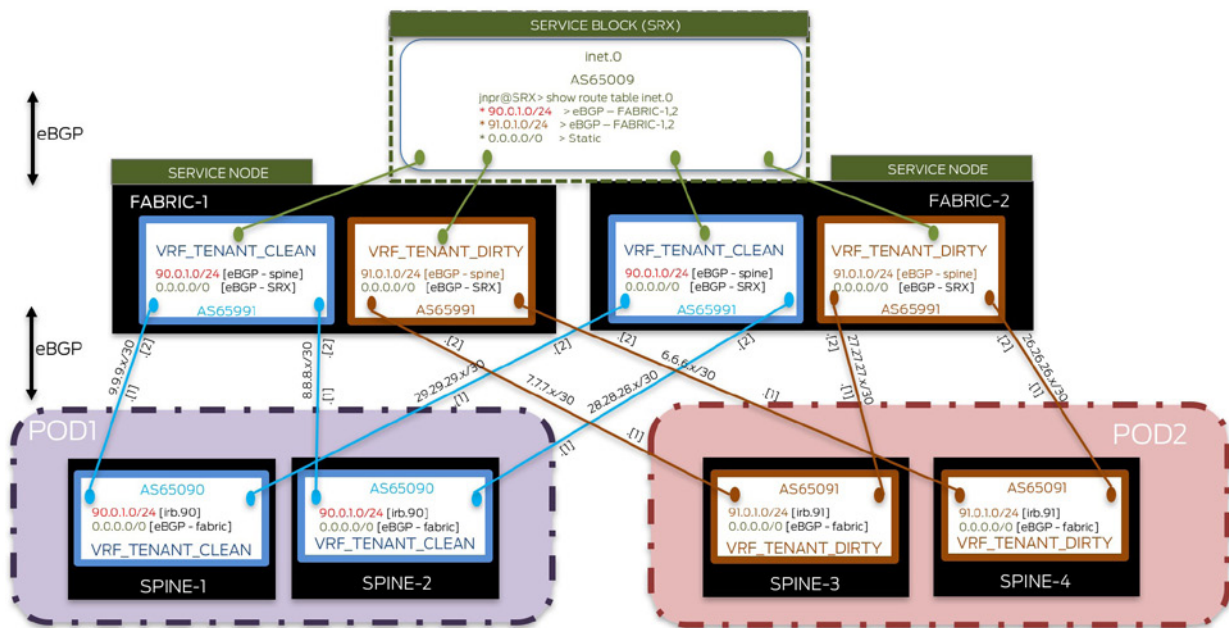


Figure 3.6 Layer 3 Security Insertion – Logical topology

One physical link each exists between SRX and service nodes - FABRIC-1 and FABRIC-2 respectively.

In the previous case study, EVPN Type 3 routes were used to get IP fabric tenant routes to the DC edge, which are then further exchanged with the service block over eBGP sessions. In this

case study, another design alternative using back-to-back VRFs between Layer 3 gateways (spine devices), service nodes (fabric devices), and the service block (firewall) is being explored.

POD1 devices (LEAF-1,2 and SPINE-1,2) are provisioned for hosts in VLAN 90 (mapped to VNI 5090), IP routes for which reside in VRF_TENANT_CLEAN. Similarly, POD2 devices (LEAF-3,4,5,6, and SPINE-3,4) are provisioned for hosts in VLAN 91 (mapped to VNI 5091), IP routes for which reside in VRF_TENANT_DIRTY. It is required for this inter-subnet traffic (VLAN 90 <> VLAN 91) between hosts residing in two different PODs to traverse the SRX.

One IFL (logical interface) for each connecting physical link (spine-fabric and fabric-firewall) terminates in the IP-VRF, routes for which need to be exchanged with the firewall. For example, inter-subnet traffic between hosts in tenant-clean (POD1) and tenant-dirty (POD2) needs to be hairpinned through the firewall. To accomplish this, one connecting IFL terminates in tenant-clean IP-VRF (VRF_TENANT_CLEAN.inet.0) and the other in tenant-dirty IP-VRF (VRF_TENANT_DIRTY.inet.0).

IP-VRF for tenant-clean exists on SPINE-1, SPINE-2, FABRIC-1, FABRIC-2, while that for tenant-dirty exists on SPINE-3, SPINE-4, FABRIC-1, FABRIC-2. An eBGP session exists over the IFL on the connecting links between spine and fabric devices and terminate in the respective IP-VRFs.

An eBGP session exists over this connecting IFL in the IP-VRF on the fabric devices (VRF_TENANT_CLEAN.inet.0 on FABRIC-1,2) and (VRF_TENANT_DIRTY.inet.0 on FABRIC-1,2) and global table (inet.0) on the SRX. Routes for all contained VLANs can be exchanged over this single eBGP session per VRF and no additional provisioning per VLAN is required.

Route Exchange Between Service Nodes and Service Block

Each spine device advertises the summary route for the host subnet to the connecting fabric device, over the eBGP session on the connecting IFL residing in the common IP-VRF. Here, SPINE-1, 2 (POD1) advertises the summary route for VLAN 90 (90.0.1/24) in VRF_TENANT_CLEAN, and SPINE-3,4 (POD2) advertises the summary route for VLAN 91 (91.0.1/24) in VRF_TENANT_DIRTY to both fabric devices.

Each fabric device, in turn, advertises these summary routes in each VRF to the firewall via eBGP. This allows the firewall to receive summary routes for both PODs via the connecting service nodes (FABRIC-1 and FABRIC-2), which can then optimally forward traffic to for a given host.

The SRX advertises a default route to each connecting fabric node, which is further advertised to all spine nodes.

Thus, when a tenant host of VRF_TENANT_DIRTY in POD2 residing on VLAN 91 (VNI 5091) intends to communicate with a tenant host of VRF_TENANT_CLEAN in POD1 residing on VLAN 90 (VNI 5090), when traffic reaches the spine devices in POD2 the destination route lookup matches the default route advertised by the SRX. As a result, traffic is punted from the spine devices to the SRX via fabric devices, which has specific route information to reach desired destination in POD1. The same behavior is observed when hosts in POD1 (VRF_TENANT_CLEAN) need to communicate with hosts in POD2 (VRF_TENANT_DIRTY). Though only traffic for a single VLAN in each tenant (VLAN 90 – VNI 5090 in VRF_TENANT_CLEAN and VLAN 91 – VNI 5091 in VRF_TENANT_DIRTY) has been demonstrated, only VLAN provisioning to accommodate increased scale in residing tenants is required and no additional configuration on the firewall is needed.

NOTE All configuration from the previous sections remains unchanged including that on spine and fabric devices. For brevity, only additional configuration on spine, fabric and firewall devices, to support manual service chaining of inter-tenant traffic (VRF_TENANT_CLEAN in POD1 and VRF_TENANT_DIRTY in POD2) have been highlighted here.

Spine Devices

SPINE-1 and SPINE-2 in POD1 have been configured to accommodate only hosts that belong to VRF_TENANT_CLEAN (VLAN 90 mapped to VNI 5090). Overlay configuration on spine devices is similar to that described in the previous use case and will not be repeated here.

Shown here is related configuration for SPINE-1 (SPINE-2 has been provisioned in a similar manner):

```
jnpr@SPINE-1> show configuration routing-instances VRF_TENANT_CLEAN | except #
instance-type vrf;
interface et-0/0/8.190;    <<< IFL on connecting link to FABRIC-1 in IP-VRF
interface et-0/0/9.190;    <<< IFL on connecting link to FABRIC-2 in IP-VRF
interface irb.90;
interface lo0.90;
route-distinguisher 90.0.10.2:10;
vrf-target target:90:90;
protocols {
  bgp {
    group viaSRX-ext { <<< eBGP sessions created between FABRIC-1,2 and tenant VRF
      type external;
      export adv-clean-summ;
      local-as 65090;
      multipath;
      neighbor 9.9.9.2 {
        description "eBGP: Spine-1 <> Fabric-1";
        peer-as 65991;
      }
      neighbor 29.29.29.2 {
        description "eBGP: Spine-1 <> Fabric-2";
        peer-as 65991;
      }
    }
  }
}

jnpr@SPINE-1> show configuration policy-options policy-statement adv-clean-summ
term 1 {
  from {
    route-
filter 90.0.1.0/24 orlonger; <<< Export policy to advertise POD1 routes to SRX via service
    nodes
  }
  then accept;
}

jnpr@SPINE-1> show configuration interfaces et-0/0/8.190
vlan-id 190;
family inet {
  address 9.9.9.1/30;
}

jnpr@SPINE-1> show configuration interfaces et-0/0/9.190
vlan-id 190;
family inet {
  address 29.29.29.1/30;
}
```

SPINE-3 and SPINE-4 in POD2 have been configured to accommodate only hosts that belong to VRF_TENANT_DIRTY (VLAN 91 mapped to VNI 5091). Overlay configuration on spine devices is similar to that described in the previous use case and will not be repeated here.

Shown herewith is the related configuration for SPINE-3 (SPINE-4 has been provisioned in a similar manner):

```
jnpr@SPINE-3> show configuration routing-instances VRF_TENANT_DIRTY | except #
instance-type vrf;
interface et-0/0/8.190; <<< IFL on connecting link to FABRIC-1 in IP-VRF
interface et-0/0/9.190; <<< IFL on connecting link to FABRIC-2 in IP-VRF
interface irb.91;
interface lo0.91;
route-distinguisher 91.0.10.4:10;
vrf-target target:91:91;
protocols {
  bgp {
    group viaSRX-ext { <<< eBGP sessions created between FABRIC-1,2 and tenant VRF
      type external;
      export adv-dirty-summ;
      local-as 65091;
    multipath;
    neighbor 7.7.7.2 {
      description "eBGP: Spine-3 <> Fabric-1";
      peer-as 65991;
    }
    neighbor 27.27.27.2 {
      description "eBGP: Spine-3 <> Fabric-2";
      peer-as 65991;
    }
  }
}

jnpr@SPINE-3> show configuration policy-options policy-statement adv-dirty-summ
term 1 {
  from {
    route-
filter 91.0.1.0/24 orlonger; <<< Export policy to advertise POD1 routes to SRX via service
                                nodes
  }
  then accept;
}

jnpr@SPINE-3> show configuration interfaces et-0/0/8.190
vlan-id 190;
family inet {
  address 7.7.7.1/30;
}

jnpr@SPINE-3> show configuration interfaces et-0/0/9.190
vlan-id 190;
family inet {
  address 27.27.27.1/30;
}
```

Fabric Devices

FABRIC-1 and FABRIC-2 have been configured with IP-VRFs (VRF_TENANT_CLEAN and VRF_TENANT_DIRTY) to exchange POD routes from the spine device with the fire-wall and vice versa. Overlay configuration on fabric devices remains unchanged and will not be repeated here.

Shown next is the related configuration for FABRIC-1 (FABRIC-2 has been provisioned in a similar manner):

```

jnpr@FABRIC-1> show configuration routing-instances VRF_TENANT_CLEAN
instance-type vrf;
interface et-3/2/0.190; <<< IFL on connecting link to SPINE-1 in IP-VRF
interface et-3/2/1.190; <<< IFL on connecting link to SPINE-2 in IP-VRF
interface xe-4/0/0.191; <<< IFL on connecting link to SRX in IP-VRF
route-distinguisher 100.0.0.1:9091;
vrf-target target:90:90;
protocols {
    bgp {
        group viaSRX-ext {
            type external;
            local-as 65991;
            multipath;
            neighbor 9.9.9.1 {
                description "eBGP: Fabric-1 <> Spine-1";
                peer-as 65090;
            }
            neighbor 19.19.19.2 {
                description "eBGP: Fabric-1 <> SRX";
                peer-as 65009;
            }
            neighbor 8.8.8.1 {
                description "eBGP: Fabric-1 <> Spine-2";
                peer-as 65090;
            }
        }
    }
}

```

```

jnpr@FABRIC-1> show configuration routing-instances VRF_TENANT_DIRTY
instance-type vrf;
interface et-3/2/2.190; <<< IFL on connecting link to SPINE-3 in IP-VRF
interface et-3/3/0.190; <<< IFL on connecting link to SPINE-4 in IP-VRF
interface xe-4/0/0.181; <<< IFL on connecting link to SRX in IP-VRF
route-distinguisher 100.0.0.3:9091;
vrf-target target:91:91;
protocols {
    bgp {
        group viaSRX-ext {
            type external;
            local-as 65991;
            multipath;
            neighbor 7.7.7.1 {
                description "eBGP: Fabric-1 <> Spine-3";
                peer-as 65091;
            }
            neighbor 18.18.18.2 {
                description "eBGP: Fabric-1 <> SRX";
                peer-as 65009;
            }
            neighbor 6.6.6.1 {
                description "eBGP: Fabric-1 <> Spine-4";
                peer-as 65091;
            }
        }
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces et-3/2/0
description " * to qfx10002-SPINE1";
vlan-tagging;
mtu 9192;
unit 0 {
    vlan-id 10;
    family inet {
        mtu 9000;
        address 172.16.0.50/31;
    }
}
unit 190 {

```

```

    description "MX ifl <> Spine-1 ifl (VRF-TENANT_CLEAN) ";
    vlan-id 190;
    family inet {
        address 9.9.9.2/30;
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces et-3/2/1
description " * to qfx10002-SPINE2";
vlan-tagging;
mtu 9192;
unit 0 {
    vlan-id 11;
    family inet {
        mtu 9000;
        address 172.16.0.52/31;
    }
}
unit 190 {
    description "MX ifl <> Spine-2 ifl (VRF-TENANT_CLEAN) ";
    vlan-id 190;
    family inet {
        address 8.8.8.2/30;
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces et-3/2/2
description " * to qfx10002-SPINE3";
vlan-tagging;
mtu 9192;
unit 0 {
    vlan-id 12;
    family inet {
        mtu 9000;
        address 172.16.0.54/31;
    }
}
unit 190 {
    description "MX ifl <> Spine-3 ifl (VRF-TENANT_DIRTY) ";
    vlan-id 190;
    family inet {
        address 7.7.7.2/30;
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces et-3/3/0
description " * to qfx10002-SPINE4";
vlan-tagging;
mtu 9192;
unit 0 {
    vlan-id 13;
    family inet {
        mtu 9000;
        address 172.16.0.56/31;
    }
}
unit 190 {
    description "MX ifl <> Spine-4 ifl (VRF-TENANT_DIRTY) ";
    vlan-id 190;
    family inet {
        address 6.6.6.2/30;
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces xe-4/0/0
description "4/0/0 FABRIC-1 to 0/0/0 SRX";
vlan-tagging;
unit 18 {
    description "MX ifl <> SRX ifl (VRF_TENANT_NS)";
    vlan-id 18;
    family inet {

```

```

        address 10.18.0.0/31;
    }
}
unit 181 {
    description "MX if1 <> SRX if1 (VRF-TENANT_DIRTY)";
    vlan-id 181;
    family inet {
        address 18.18.18.1/30;
    }
}
unit 191 {
    description "MX if1 <> SRX if1 (VRF-TENANT_CLEAN)";
    vlan-id 191;
    family inet {
        address 19.19.19.1/30;
    }
}
}

```

SRX

EBGP sessions are created from the SRX (inet.0) and each connected fabric device (tenant IP-VRF).

```

jnpr@SRX> show configuration protocols bgp
group viaSRX-ext {
    type external;
    export adv-default;
    local-as 65009;
    multipath;
    neighbor 19.19.19.1 {
        description "eBGP: SRX <> Fabric-1 (VRF_TENANT_CLEAN)";
        peer-as 65991;
    }
    neighbor 39.39.39.1 {
        description "eBGP: SRX <> Fabric-2 (VRF_TENANT_CLEAN)";
        peer-as 65991;
    }
    neighbor 18.18.18.1 {
        description "eBGP: SRX <> Fabric-1 (VRF_TENANT_DIRTY)";
        peer-as 65991;
    }
    neighbor 38.38.38.1 {
        description "eBGP: SRX <> Fabric-2 (VRF_TENANT_DIRTY)";
        peer-as 65991;
    }
    neighbor 10.18.0.0 {
        peer-as 65180;
    }
    neighbor 10.19.0.0 {
        peer-as 65180;
    }
}
jnpr@SRX> show configuration routing-options static
route 0.0.0.0/0
    reject;

jnpr@SRX> show configuration policy-options policy-statement adv-default
term static-rt { <<< Export default route to DC PODs
from {
    protocol static;
    route-filter 0.0.0.0/0 exact;
}
    then accept;
}
term default {
    then reject;
}
}

```

Verification

Spine devices advertise summary routes to fabric devices and receive the default route originally advertised by the SRX.

<<< Summary route for VLAN 90 advertised to FABRIC-1 and FABRIC-2 (POD1 spine devices)

```
jnpr@SPINE-1> show route advertising-protocol bgp 9.9.9.2
VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 90.0.1.0/24  Self                MED      Lclpref      I
```

```
jnpr@SPINE-1> show route advertising-protocol bgp 29.29.29.2
VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 90.0.1.0/24  Self                MED      Lclpref      I
```

<<< 0/0 advertised by SRX received from FABRIC-1 and FABRIC-2 by POD1 spine devices

```
jnpr@SPINE-1> show route receive-protocol bgp 9.9.9.2
VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 0.0.0.0/0    9.9.9.2                MED      Lclpref      65991 65009 I
```

```
jnpr@SPINE-1> show route receive-protocol bgp 29.29.29.2
VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
  0.0.0.0/0    29.29.29.2                MED      Lclpref      65991 65009 I
```

<<< Summary route for VLAN 91 advertised to FABRIC-1 and FABRIC-2 (POD2 spine devices)

```
jnpr@SPINE-3> show route advertising-protocol bgp 7.7.7.2
VRF_TENANT_DIRTY.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 91.0.1.0/24  Self                MED      Lclpref      I
```

```
jnpr@SPINE-3> show route advertising-protocol bgp 27.27.27.2
VRF_TENANT_DIRTY.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 91.0.1.0/24  Self                MED      Lclpref      I
```

<<< 0/0 advertised by SRX received from FABRIC-1 and FABRIC-2 by POD2 spine devices

```
jnpr@SPINE-3> show route receive-protocol bgp 7.7.7.2
VRF_TENANT_DIRTY.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
* 0.0.0.0/0    7.7.7.2                MED      Lclpref      65991 65009 I
```

```
jnpr@SPINE-3> show route receive-protocol bgp 27.27.27.2
VRF_TENANT_DIRTY.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED      Lclpref      AS path
  0.0.0.0/0    27.27.27.2                MED      Lclpref      65991 65009 I
```

The SRX receives POD summary routes from the fabric devices.

```
jnpr@SRX> show route 90.0.1.100
```

```
inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
90.0.1.0/24      *[BGP/170] 04:51:31, localpref 100, from 19.19.19.1
                  AS path: 65991 65090 I, validation-state: unverified
                  > to 39.39.39.1 via xe-0/0/10.191
                  > to 19.19.19.1 via xe-0/0/0.191
                  [BGP/170] 00:15:56, localpref 100
                  AS path: 65991 65090 I, validation-state: unverified
                  > to 39.39.39.1 via xe-0/0/10.191
```

```
jnpr@SRX> show route forwarding-table destination 90.0.1.100
```

```
Routing table: default.inet
```

```
Internet:
```

Destination	Type	RtRef	Next hop	Type	Index	NhRef	Netif
90.0.1.0/24	user	0		ulst	1048575	2	


```

39.39.39.1      ucst      589      3 xe-0/0/10.191
19.19.19.1      ucst      545      3 xe-0/0/0.191

```

```

jnpr@SRX> show route 91.0.1.100
inet.0: 29 destinations, 31 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

91.0.1.0/24      *[BGP/170] 03:26:05, localpref 100
                  AS path: 65991 65091 I, validation-state: unverified
> to 18.18.18.1 via xe-0/0/0.181
> to 38.38.38.1 via xe-0/0/10.181
[BGP/170] 00:15:56, localpref 100
                  AS path: 65991 65091 I, validation-state: unverified
> to 38.38.38.1 via xe-0/0/10.181

```

```

jnpr@SRX> show route forwarding-table destination 91.0.1.101
Routing table: default.inet

```

```

Internet:
Destination      Type RtRef Next hop      Type Index  NhRef Netif
91.0.1.0/24      user  0
                  18.18.18.1    ucst    557      3 xe-0/0/0.181
                  38.38.38.1    ucst    594      3 xe-0/0/10.181

```

Fabric devices can optimally forward traffic to the relevant POD.

```

jnpr@FABRIC-1> show route 90.0.1.100

```

```

VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

90.0.1.0/24      *[BGP/170] 05:22:39, localpref 100 <<< VLAN 90 traffic directed to POD1
                  AS path: 65090 I, validation-state: unverified
> to 9.9.9.1 via et-3/2/0.190
> to 8.8.8.1 via et-3/2/1.190
[BGP/170] 04:29:48, localpref 100
                  AS path: 65090 I, validation-state: unverified
> to 8.8.8.1 via et-3/2/1.190

```

```

jnpr@FABRIC-1> show route 91.0.1.101
VRF_TENANT_DIRTY.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

91.0.1.0/24      *[BGP/170] 03:36:52, localpref 100 <<< VLAN 91 traffic directed to POD2
                  AS path: 65091 I, validation-state: unverified
> to 7.7.7.1 via et-3/2/2.190
> to 6.6.6.1 via et-3/3/0.190
[BGP/170] 03:03:43, localpref 100
                  AS path: 65091 I, validation-state: unverified
> to 6.6.6.1 via et-3/3/0.190

```

Traffic Flows

Figure 3.7 depicts the path traversed by traffic between host H1-90 (VRF_TENANT_CLEAN: IP 90.0.1.100) and host H3-91 (VRF_TENANT_DIRTY: IP 91.0.1.101) hair-pinning through the DC firewall (SRX).

Hair-pinning through the SRX is observed for bidirectional traffic:

SRX			Seconds: 174		Time: 11:27:27
Interface	Link	Input packets	(pps)	Output packets	(pps)
xe-0/0/0	Up	135202843	(1050)	149176680	(999)
xe-0/0/10	Up	19153901	(947)	45755	(998)

```

jnpr@SRX> show security flow session
Flow Sessions on FPC2 PIC1:

```

Session ID: 93922154, Policy name: 1/6, Timeout: 58, Valid
 In: 91.0.1.101/75 --> 90.0.1.100/84;udp, Conn Tag: 0x0, If: xe-0/0/0.181, Pkts: 298, Bytes: 291444, CP Session ID: 94915992
 Out: 90.0.1.100/84 --> 91.0.1.101/75;udp, Conn Tag: 0x0, If: xe-0/0/0.191, Pkts: 0, Bytes: 0, CP Session ID: 94915992

Session ID: 93945983, Policy name: 1/6, Timeout: 58, Valid
 In: 91.0.1.101/797 --> 90.0.1.100/806;udp, Conn Tag: 0x0, If: xe-0/0/0.181, Pkts: 296, Bytes: 289488, CP Session ID: 96030510
 Out: 90.0.1.100/806 --> 91.0.1.101/797;udp, Conn Tag: 0x0, If: xe-0/0/0.191, Pkts: 0, Bytes: 0, CP Session ID: 96030510

Session ID: 94450140, Policy name: 1/6, Timeout: 58, Valid
 In: 90.0.1.100/187 --> 91.0.1.101/195;udp, Conn Tag: 0x0, If: xe-0/0/0.191, Pkts: 274, Bytes: 267972, CP Session ID: 96030524
 Out: 91.0.1.101/195 --> 90.0.1.100/187;udp, Conn Tag: 0x0, If: xe-0/0/0.181, Pkts: 0, Bytes: 0, CP Session ID: 96030524

Session ID: 94450151, Policy name: 1/6, Timeout: 58, Valid
 In: 90.0.1.100/188 --> 91.0.1.101/196;udp, Conn Tag: 0x0, If: xe-0/0/0.191, Pkts: 272, Bytes: 266016, CP Session ID: 96030526
 Out: 91.0.1.101/196 --> 90.0.1.100/188;udp, Conn Tag: 0x0, If: xe-0/0/0.181, Pkts: 0, Bytes: 0, CP Session ID: 96030526

IXIA Statistics - Intra DC Inter-POD (H1-90 <> H3-91: 90.0.1.100 <> 91.0.1.101)

(1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Tx L1 Rate (bps)	Rx L1 Rate (bps)	Frames Delta	Loss %	Packet Loss (ms)
5_Inter-VNI-SRX-bi-90 < 91	333,873	333,873	8,160,000.000	8,160,000.000	0	0.000	
5_Inter-VNI-SRX-bi-90 > 91	17,881	17,880	8,160,000.000	8,160,000.000	1	0.006	

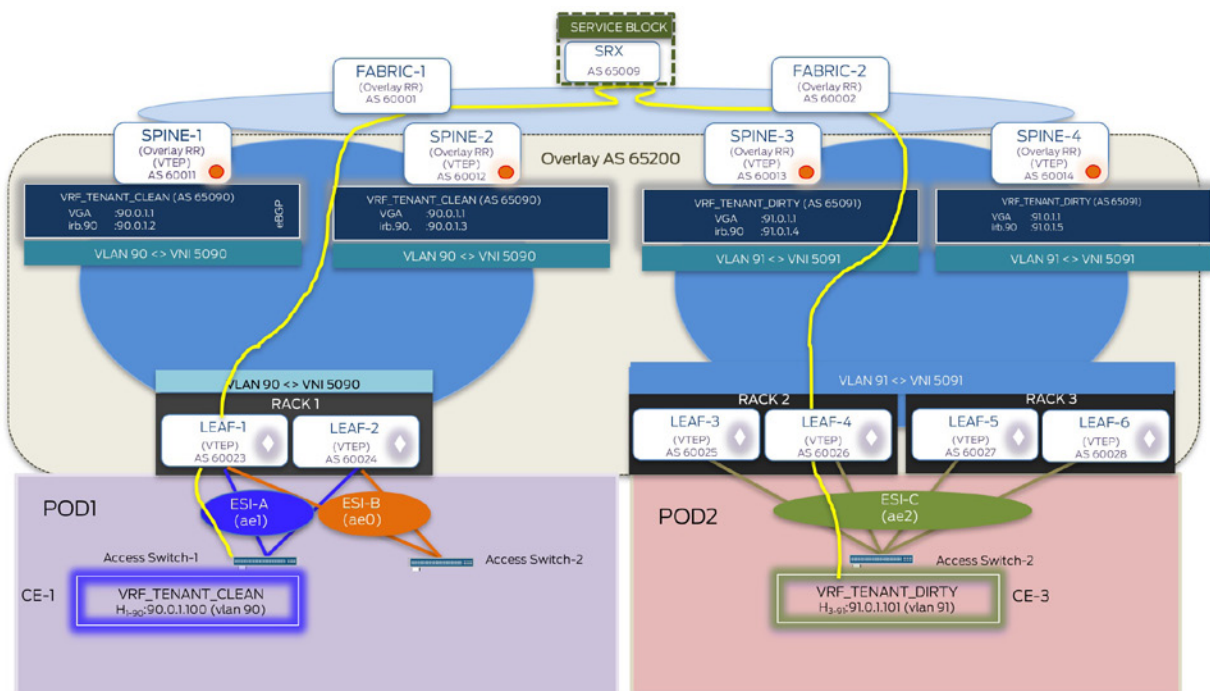


Figure 3.7 Manual Service Chaining for Inter-POD EW – Inter-VRF Inter Subnet Traffic (H1-90 > H3-91)

Manual Service Chaining of North-South Traffic

(Layer 3 security insertion: Inter-tenant Inter-subnet)

For N-S traffic between hosts in different tenants, manual service chaining is demonstrated by hair-pinning traffic through the local DC firewall for policy enforcement before forwarding it to the intended destination. The DC firewall (SRX) connected to each fabric device is used to process E-W and N-S traffic. As elaborated previously, service chaining is treated as an IP path problem by getting traffic to the firewall. Details on firewall rules and policy enforcement are outside the scope of this book.

Service node: Refers to the fabric devices that service each POD by directing intended traffic from the IP fabric to the service block (firewall). These fabric devices are also acting as DC-edge that connect the data center to the WAN.

Service block: Refers to the firewall that is responsible for service chaining inter-tenant traffic (E-W and N-S).

The following sections elaborate upon the control plane and data plane details used to achieve the Layer 3 security insertion for N-S inter-tenant traffic.

Configuration Layout

All configuration details as explained in previous sections remain as is. Figure 3.8 illustrates the logical topology on how the firewall is inserted in the DC to process N-S traffic between different tenants.

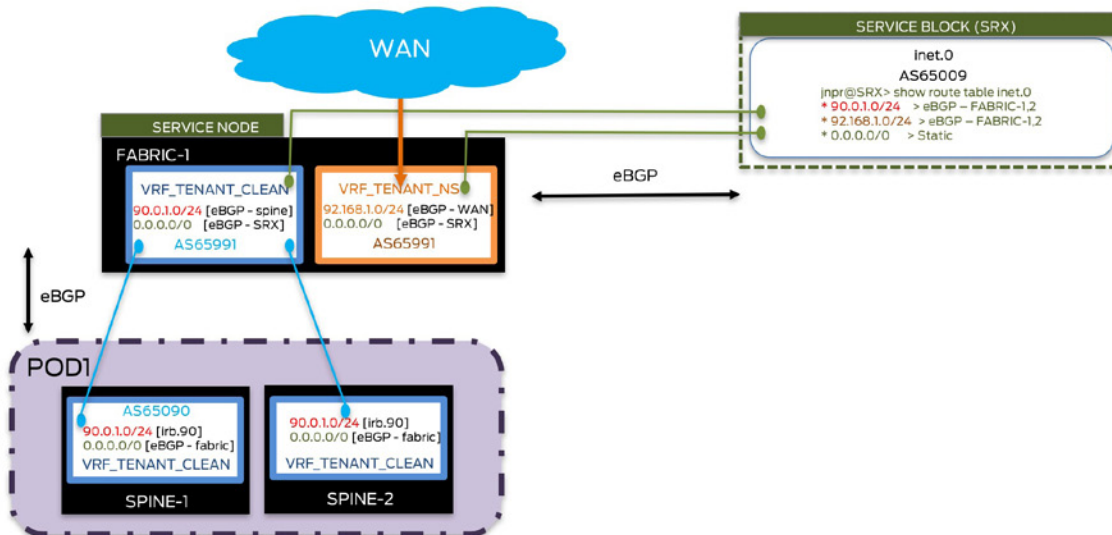


Figure 3.8 Layer 3 Security Insertion for N-S Traffic – Logical Topology

For brevity, Figure 3.8 focuses on FABRIC-1 and SPINE-1,2 – other devices have been configured similarly.

The route for a north-bound host across the WAN is received by FABRIC-1 from a CE simulated on IXIA. This route is received from eBGP over the PE-CE link (IFL on the fabric device in VRF_TENANT_NS).

Each DC edge consists an IP-VRF (VRF_TENANT_NS) for populating routes learned from the WAN. No DC routes exist in this IP-VRF.

Each fabric device acts as the DC edge connecting the data center to the WAN.

On the connecting physical link between fabric devices and the firewall one IFL (logical interface) exists in this IP-VR (VRF_TENANT_NS).

An eBGP session is created over this connecting IFL in the IP-VRF on each fabric device and the firewall. Shown above, eBGP sessions exist on the IFLs in VRF_TENANT_NS.inet.0 on FABRIC-1 and the global table (inet.0) on the firewall. Routes for all hosts across the WAN can be exchanged over this single eBGP session per VRF and no additional provisioning per host/VLAN is required.

Demonstrated in this section is inter-subnet traffic between hosts in tenant-clean(POD1) and across the WAN, hair-pinning through the firewall.

Route Exchange Between Service Nodes and Service Block

Configuration from the previous sections remains unchanged.

To hair-pin N-S traffic through the SRX, each service node advertise routes learned from the WAN directly to the firewall, over the eBGP session between NS IP-VRF (VRF_TENANT_NS) and the DC firewall.

The default route received from the SRX is further advertised towards the WAN.

As a result, when a host across the WAN (92.168.1.1) wants to send traffic to a host in DC host, tenant clean, the destination lookup for the tenant clean host is done in the N-S IP-VRF (VRF_TENANT_NS.inet.0). Since this IP-VRF is not populated with any DC tenant routes, the default route received from the firewall is the best match. Thus traffic is directed from the service nodes (FABRIC-1 and FABRIC-2) to the firewall.

Since the firewall has summary route information for data center hosts due to eBGP sessions terminating in the tenant IP-VRFs on the fabric devices, traffic is hair-pinned through the firewall and forwarded optimally by the service nodes towards the DC PODs.

Configuration

All configuration from the previous sections remains unchanged including that on leaf, spine, and fabric devices. For brevity, only additional configuration on fabric and firewall devices, to support manual service chaining of N-S inter-tenant traffic, have been highlighted here.

Fabric Devices

Shown below is configuration for FABRIC-1 (FABRIC-2 has been configured similarly):

```
jnpr@FABRIC-1> show configuration routing-instances VRF_TENANT_NS
instance-type vrf;
  interface et-3/3/1.0; <<< IFL on connecting link to WAN-PE
  interface xe-4/0/0.18; <<< IFL on connecting link to SRX
route-distinguisher 100.0.1.18:18;
vrf-target target:18:18;
protocols {
  bgp {
    group from-WAN {
      type external;
      local-address 162.18.0.2;
      local-as 65180;
      multipath;
      neighbor 162.18.0.1 {
```

```

        description "eBGP peering with WAN-PE (VRF_TENANT_NS)";
        peer-as 65181;
    }
}
group NS-viaSRX {
    type external;
    local-address 10.18.0.0;
    local-as 65180;
    multipath;
    neighbor 10.18.0.1 {
description "eBGP peering with SRX (VRF_TENANT_NS)";
        peer-as 65009;
    }
}
}
}

```

```

jnpr@FABRIC-1> show configuration interfaces et-3/3/1
mtu 9192;
unit 0 {
    description "0/0/49 Acc-SW to 3/3/1 Fabric-1";
    family inet {
        address 162.18.0.2/30;
    }
}

```

```

jnpr@FABRIC-1> show configuration interfaces xe-4/0/0
description "4/0/0 FABRIC-1 to 0/0/0 SRX";
vlan-tagging;
unit 18 {
    description "MX ifl <> SRX ifl (VRF_TENANT_NS)";
    vlan-id 18;
    family inet {
        address 10.18.0.0/31;
    }
}

```

SRX

EBGP sessions are created from the SRX (inet.0) and each connected fabric device (tenant IP-VRF and NS IP-VRF):

```

jnpr@SRX> show configuration protocols bgp
group viaSRX-ext {
    type external;
    export adv-default;
    local-as 65009;
    multipath;
    neighbor 19.19.19.1 {
        description "eBGP: SRX <> Fabric-1 (VRF_TENANT_CLEAN)";
        peer-as 65991;
    }
    neighbor 39.39.39.1 {
        description "eBGP: SRX <> Fabric-2 (VRF_TENANT_CLEAN)";
        peer-as 65991;
    }
    neighbor 18.18.18.1 {
        description "eBGP: SRX <> Fabric-1 (VRF_TENANT_DIRTY)";
        peer-as 65991;
    }
    neighbor 38.38.38.1 {
        description "eBGP: SRX <> Fabric-2 (VRF_TENANT_DIRTY)";
        peer-as 65991;
    }
}

```

```

}
neighbor 10.18.0.0 {
    description "eBGP: SRX <> Fabric-1 (VRF_TENANT_NS)";
    peer-as 65180;
}
neighbor 10.19.0.0 {
    description "eBGP: SRX <> Fabric-2 (VRF_TENANT_NS)";
    peer-as 65180;
}
}

```

Verification

The SRX receives data center tenant routes as well as those for northbound hosts across the WAN. All North to South traffic is directed to the firewall, which it can then route to the DC tenant hosts since summary routes are known.

```

jnpr@SRX> show route 92.168.1.100 <<< WAN route received by SRX from fabric devices

```

```

inet.0: 29 destinations, 32 routes (29 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

92.168.1.0/24      *[BGP/170] 00:02:24, localpref 100, from 10.18.0.0
                   AS path: 65180 65181 65171 65172 I, validation-state: unverified
                   > to 10.18.0.0 via xe-0/0/0.18
                   > to 10.19.0.0 via xe-0/0/10.18
                   [BGP/170] 00:02:23, localpref 100
                   AS path: 65180 65182 65171 65172 I, validation-state: unverified
                   > to 10.19.0.0 via xe-0/0/10.18

```

```

jnpr@SRX> show route 90.0.1.100 <<< DC tenant route received by SRX from fabric devices

```

```

inet.0: 26 destinations, 26 routes (26 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

90.0.1.0/24      *[BGP/170] 05:07:37, localpref 100
                   AS path: 65991 65090 I, validation-state: unverified
                   > to 19.19.19.1 via xe-0/0/0.191
                   > to 39.39.39.1 via xe-0/0/10.191

```

```

jnpr@SPINE-1> show route 92.168.1.100 <<< To reach WAN, DC host traffic directed to SRX
VRF_TENANT_CLEAN.inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

0.0.0.0/0        *[BGP/170] 00:01:18, localpref 100, from 9.9.9.2
                   AS path: 65991 65009 I, validation-state: unverified
                   > to 9.9.9.2 via et-0/0/8.190
                   > to 29.29.29.2 via et-0/0/9.190
                   [BGP/170] 00:01:09, localpref 100
                   AS path: 65991 65009 I, validation-state: unverified
                   > to 29.29.29.2 via et-0/0/9.190

```

Traffic Flows

Figure 3.9 depicts the path traversed by traffic between hosts across the WAN and in the DC, hair-pinning through the DC firewall (SRX).

H0 (VRF_TENANT_NS: IP 92.168.1.1) and host H1-90 (VRF_TENANT_CLEAN: IP 90.0.1.100)

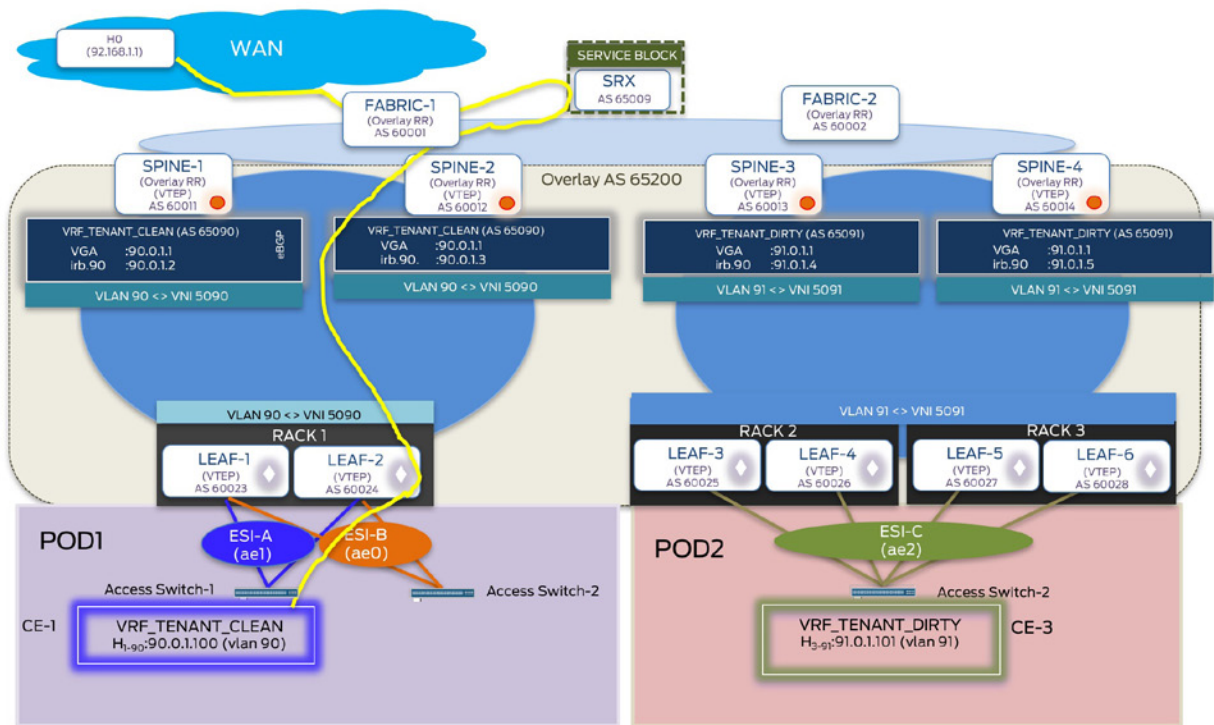


Figure 3.9 Manual Service Chaining for N-S – Inter-VRF Inter Subnet Traffic (H0 > H1-90 traffic path)

Hair-pinning through the SRX is observed for bidirectional traffic:

SRX		Seconds: 7		Time: 08:19:1	
Interface	Link	Input packets	(pps)	Output packets	(pps)
xe-0/0/0	Up	894969	(1998)	813449	(999)
xe-0/0/10	Up	235	(0)	77749	(997)

```
jnpr@SRX> show security flow session
Flow Sessions on FPC2 PIC1:
```

```
Session ID: 90001923, Policy name: 1/6, Timeout: 58, Valid
In: 92.168.1.1/1 --> 90.0.1.100/9;udp, Conn Tag: 0x0, If: xe-
0/0/0.18, Pkts: 106, Bytes: 104092, CP Session ID: 90000654
Out: 90.0.1.100/9 --> 92.168.1.1/1;udp, Conn Tag: 0x0, If: xe-
0/0/10.191, Pkts: 0, Bytes: 0, CP Session ID: 90000654
!
Session ID: 100000127, Policy name: 1/6, Timeout: 1800, Valid
In: 90.0.1.100/1 --> 92.168.1.1/1;61, Conn Tag: 0x0, If: xe-
0/0/0.191, Pkts: 562371, Bytes: 549998838, CP Session ID: 100000067
Out: 92.168.1.1/1 --> 90.0.1.100/1;61, Conn Tag: 0x0, If: xe-
0/0/0.18, Pkts: 0, Bytes: 0, CP Session ID: 100000067
```

Class of Service

This additional section illustrates application of basic CoS functions in a leaf-spine fabric employing an EVPN-VXLAN overlay. CoS configuration below has been added to the existing configuration and applied to a subset of the topology as depicted in Figure 3.10.

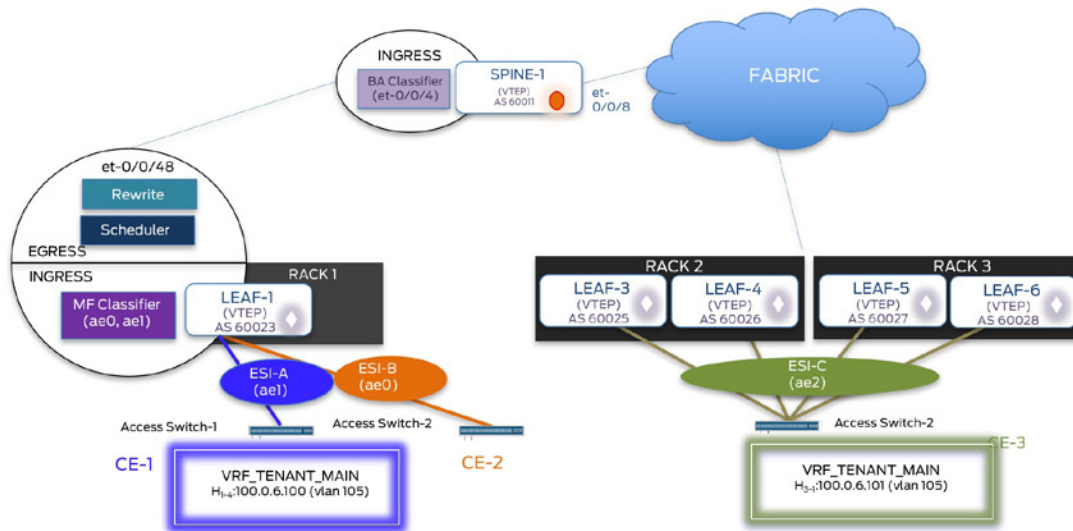


Figure 3.10 Class of Service Functions on LEAF-1 and SPINE-1

Configuration

For the single tenant environment (VRF_TENANT_MAIN) demonstrated in this use case, CoS functions are applied to different subnets for the same tenant. However, the same can be applied to multiple tenants as well. CoS configuration has been applied only to LEAF-1 and SPINE-1, with all other interfaces between the two disabled except for those highlighted in the diagram above.

LEAF-1

Multi-field classifier has been applied to the ingress interfaces on LEAF-1 that receive traffic from CE-1 (ESI-A) and CE-2 (ESI-B). This further classifies traffic for VLAN 105 into three forwarding classes: FC_v4-1p (dot1p traffic), FC_v4-UDP (IPv4 UDP traffic), and FC_v4-TCP (IPv4 TCP traffic).

```
jnpr@LEAF-1> show configuration firewall
family ethernet-switching {
  filter MF-classifier {
    term dot1p {
      from {
        user-vlan-1p-priority 7;
      }
      then {
        accept;
        forwarding-class FC_v4-1p;
        loss-priority low;
      }
    }
    term IP-UDP {
      from {
        ip-source-address {
          100.0.6.0/24;
        }
        ip-protocol udp;
      }
      then {
        accept;
        forwarding-class FC_v4-UDP;
      }
    }
  }
}
```

```

        loss-priority low;
    }
}
term IP-TCP {
    from {
        ip-source-address {
            100.0.6.0/24;
        }
        ip-protocol tcp;
    }
    then {
        accept;
        forwarding-class FC_v4-TCP;
        loss-priority low;
    }
}
term default {
    then accept;
}
}
}

```

```

jnpr@LEAF-1> show configuration interfaces ae1
apply-groups-except MTU-VXLAN;
description "ae1(ESI-A) to Access-Sw-1";
mtu 9192;
esi {
    00:33:33:33:33:33:33:33:33;
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:00:03;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ bd9100 bd9101 bd9102 bd9103 bd9104 bd9105 bd9106 bd5090 ];
        }
        filter {
            input MF-classfier; <<< MF classification for tenant traffic on ESI-A (CE-1)
        }
    }
}

```

```

jnpr@LEAF-1> show configuration interfaces ae0
apply-groups-except MTU-VXLAN;
description "ae0(ESI-B) to Access-Sw-2";
mtu 9192;
esi {
    00:22:22:22:22:22:22:22:22;
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        system-id 00:00:00:00:00:02;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ bd9100 bd9101 bd9102 bd9103 bd9104 ];
        }
    }
}

```

```

    }
    filter {
        input MF-classfier; <<< MF classification for tenant traffic on ESI-B (CE-2)
    }
}

jnpr@LEAF-1> show configuration class-of-service

forwarding-classes { <<< classified traffic is placed into the specified forwarding classes
    class FC_v4-UDP queue-num 2;
    class FC_v4-1p queue-num 3;
    class FC_v4-TCP queue-num 5;
}

<<< Scheduling and shaping is configured using 3 constructs - schedulers, scheduler-maps and traffic-
control-profiles

schedulers {
    SCH_v4-UDP {
        transmit-rate percent 60;
    }
    priority low;
    SCH_v4-1p {
        transmit-rate percent 35;
        priority low;
    }
    SCH_v4-TCP {
        shaping-rate percent 5; <<< Rate limits strict-high traffic to prevent
        starvation of other queues
        priority strict-high;
    }
}

scheduler-maps { <<< maps scheduler to forwarding classes present on an interface
    SCH-MAP_DC-CoS {
        forwarding-class FC_v4-UDP scheduler SCH_v4-UDP;
        forwarding-class FC_v4-1p scheduler SCH_v4-1p;
    }
    SCH-MAP_DC-CoS-SH {
        forwarding-class FC_v4-TCP scheduler SCH_v4-TCP;
    }
}

traffic-control-profiles {
    TC_DC_COS {
        scheduler-map SCH-MAP_DC-CoS;
        guaranteed-rate percent 60;
    }
    TC_DC_COS-SH {
        scheduler-map SCH-MAP_DC-CoS-SH;
    }
}

forwarding-class-sets {
    TC_DC_COS-SH {
        class FC_v4-TCP;
    }
    TC_DC_COS {
        class FC_v4-UDP;
        class FC_v4-1p;
    }
}

interfaces {
    et-0/0/48 {
        forwarding-class-set {
            TC_DC_COS-SH {
                output-traffic-control-profile TC_DC_COS-SH;
            }
        }
    }
}

```

```

    }
    TC_DC_COS {
        output-traffic-control-profile TC_DC_COS;
    }
}
rewrite-rules {
    dscp dscp_rewrite;
}
}
}

```

<<< VXLAN honors the rewrite function by marking DSCP (BE) in the inner IP header to DSCP (AF21) in the outer IP header on performing encapsulation

```

rewrite-rules {
    dscp dscp_rewrite {
        forwarding-class FC_v4-UDP {
            loss-priority low code-point af21;
        }
    }
}

```

SPINE-1

The BA classifier is applied on the ingress interface on SPINE-1 that classifies encapsulated-VXLAN traffic based on the rewrite marking on the outer IP header.

```

jnpr@SPINE-1> show configuration class-of-service
classifiers {
    dscp dscp_classifier {
        forwarding-class FC_v4-UDP {
            loss-priority low code-points af21;
        }
    }
}
forwarding-classes {
    class FC_v4-UDP queue-num 2;
    class FC_v4-1p queue-num 3;
    class FC_v4-TCP queue-num 5;
}
interfaces {
    et-0/0/4 {
        unit 0 {
            classifiers {
                dscp dscp_classifier;
            }
        }
    }
}
}

```

Verification

Multi-field classification can be verified from the output below – ingress UDP IPv4 traffic (VLAN 105: ESI A – ESI C: 100.0.6.100 <> 100.0.6.101) is classified into the intended forwarding class.

```

jnpr@LEAF-1> monitor interface traffic
LEAF-1
Interface  Link  Input packets  Seconds: 7  Output packets  Time: 19:54:16
et-0/0/48  Up    37153593483    (49950)    35293315982    (50100) <<< To SPINE-1
et-0/0/49  Down  36839035479    (0)        37897822986    (0)
et-0/0/50  Down  3412795529     (0)        2659789934     (0)
et-0/0/51  Down  4430853025     (0)        6202651636     (0)
et-0/0/52  Up    48             (0)        27541331        (0)
ae0        Up    51537323790    (1)        36408690666    (1)
ae1        Up    80375813346    (50100)    95080856704    (49943)

```

```

jnpr@LEAF-1> show interfaces queue et-0/0/48
Physical interface: et-0/0/48, Enabled, Physical link is Up
Interface index: 684, SNMP ifIndex: 512
Description: * to SPINE-1
Forwarding classes: 16 supported, 7 in use
Egress queues: 12 supported, 7 in use
Queue: 0, Forwarding classes: best-effort
  Queued:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
  Transmitted:
    Packets      : 61 0 pps
    Bytes        : 4809 0 bps
    Tail-dropped packets : Not Available
    RL-dropped packets : 0 0 pps
    RL-dropped bytes  : 0 0 bps
    Total-dropped packets: 0 0 pps
    Total-dropped bytes : 0 0 bps
Queue: 2, Forwarding classes: FC_v4-UDP
  Queued:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
  Transmitted:
    Packets      : 12434245 49924 pps
    Bytes        : 13006221316 417775456 bps
    Tail-dropped packets : Not Available
    RL-dropped packets : 0 0 pps
    RL-dropped bytes  : 0 0 bps
    Total-dropped packets: 0 0 pps
    Total-dropped bytes : 0 0 bps
Queue: 3, Forwarding classes: FC_v4-1p
  Queued:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
  Transmitted:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
    Tail-dropped packets : Not Available
    RL-dropped packets : 0 0 pps
    RL-dropped bytes  : 0 0 bps
    Total-dropped packets: 0 0 pps
    Total-dropped bytes : 0 0 bps

Queue: 4, Forwarding classes: no-loss
  Queued:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
  Transmitted:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
    Tail-dropped packets : Not Available
    RL-dropped packets : 0 0 pps
    RL-dropped bytes  : 0 0 bps
    Total-dropped packets: 0 0 pps
    Total-dropped bytes : 0 0 bps
Queue: 5, Forwarding classes: FC_v4-TCP
  Queued:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
  Transmitted:
    Packets      : 0 0 pps
    Bytes        : 0 0 bps
    Tail-dropped packets : Not Available
    RL-dropped packets : 0 0 pps
    RL-dropped bytes  : 0 0 bps
    Total-dropped packets: 0 0 pps
    Total-dropped bytes : 0 0 bps

```

```

Queue: 7, Forwarding classes: network-control
Queued:
  Packets      :           0           0 pps
  Bytes       :           0           0 bps
Transmitted:
  Packets      :       2398           9 pps
  Bytes       :    170101        6424 bps
Tail-dropped packets : Not Available
RL-dropped packets  :           0           0 pps
RL-dropped bytes   :           0           0 bps
Total-dropped packets:           0           0 pps
Total-dropped bytes :           0           0 bps
Queue: 8, Forwarding classes: mcast
Queued:
  Packets      :           0           0 pps
  Bytes       :           0           0 bps
Transmitted:
  Packets      :           0           0 pps
  Bytes       :           0           0 bps
Tail-dropped packets : Not Available
RL-dropped packets  :           0           0 pps
RL-dropped bytes   :           0           0 bps
Total-dropped packets:           0           0 pps
Total-dropped bytes :           0           0 bps

```

Oversubscription is simulated on the egress port with strict-high traffic being out-of-contract (@3Gbps) while the other lower priority traffic is in-contract. Because the shaping rate has been defined, no queue starvation is caused and strict-high is rate-limited at 5% of egress interface bandwidth (5% of 40G = 2G):

```

jnpr@LEAF-1> show interfaces queue et-0/0/48
Physical interface: et-0/0/48, Enabled, Physical link is Up
  Interface index: 684, SNMP ifIndex: 512
  Description: * to SPINE-1
Forwarding classes: 16 supported, 7 in use
Egress queues: 12 supported, 7 in use
Queue: 0, Forwarding classes: best-effort
Queued:
  Packets      :           0           0 pps
  Bytes       :           0           0 bps
Transmitted:
  Packets      :       325           0 pps
  Bytes       :    28239        432 bps
Tail-dropped packets : Not Available
RL-dropped packets  :           0           0 pps
RL-dropped bytes   :           0           0 bps
Total-dropped packets:           0           0 pps
Total-dropped bytes :           0           0 bps
Queue: 2, Forwarding classes: FC_v4-UDP
Queued:
  Packets      :           0           0 pps
  Bytes       :           0           0 bps
Transmitted:
  Packets      :    550940995        2794914 pps
  Bytes       :  576284299598        23387842736 bps
Tail-dropped packets : Not Available
RL-dropped packets  :           0           0 pps <<< No drops
RL-dropped bytes   :           0           0 bps
Total-dropped packets:           0           0 pps
Total-dropped bytes :           0           0 bps
Queue: 3, Forwarding classes: FC_v4-1p

```

```

Queued:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
Transmitted:
  Packets      :      339704900      1618108 pps
  Bytes       :      355331336906      13540334560 bps
  Tail-dropped packets : Not Available <<< No drops
  RL-dropped packets :      0      0 pps
  RL-dropped bytes  :      0      0 bps
  Total-dropped packets:      0      0 pps
  Total-dropped bytes :      0      0 bps
Queue: 4, Forwarding classes: no-loss
Queued:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
Transmitted:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
  Tail-dropped packets : Not Available
  RL-dropped packets :      0      0 pps
  RL-dropped bytes  :      0      0 bps
  Total-dropped packets:      0      0 pps
  Total-dropped bytes :      0      0 bps

Queue: 5, Forwarding classes: FC_v4-TCP
Queued:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
Transmitted:
  Packets      :      61325704      239073 pps
  Bytes       :      64146686384      2000552256 bps <<<Rate-limited
  Tail-dropped packets : Not Available
  RL-dropped packets :      0      0 pps
  RL-dropped bytes  :      0      0 bps
  Total-dropped packets:      10829598      128536 pps
  Total-dropped bytes :      10829598000      1028284496 bps
Queue: 7, Forwarding classes: network-control
Queued:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
Transmitted:
  Packets      :      10425      9 pps
  Bytes       :      739469      5440 bps
  Tail-dropped packets : Not Available
  RL-dropped packets :      0      0 pps
  RL-dropped bytes  :      0      0 bps
  Total-dropped packets:      0      0 pps
  Total-dropped bytes :      0      0 bps
Queue: 8, Forwarding classes: mcast
Queued:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
Transmitted:
  Packets      :      0      0 pps
  Bytes       :      0      0 bps
  Tail-dropped packets : Not Available
  RL-dropped packets :      0      0 pps
  RL-dropped bytes  :      0      0 bps
  Total-dropped packets:      0      0 pps
  Total-dropped bytes :      0      0 bps

```

Rewrite rules are appropriately translated on VXLAN encapsulation performed at LEAF-1. As can be seen here, DSCP bits (BE) on the inner IP header are re-written to DSCP (AF21) on the outer IP header and classified into the desired queue on SPINE-1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	100.0.6.100	100.0.6.101	UDP	1046	22320 → 35940 Len=950
2	0.000016770	100.0.6.100	100.0.6.101	UDP	1046	24234 → 57639 Len=950
3	0.000096250	100.0.6.100	100.0.6.101	UDP	1046	12308 → 18979 Len=950
4	0.000100200	100.0.6.100	100.0.6.101	UDP	1046	14030 → 20111 Len=950
▶ Frame 1: 1046 bytes on wire (8368 bits), 1046 bytes captured (8368 bits)						
▶ Ethernet II, Src: JuniperN_49:29:33 (64:64:9b:49:29:33), Dst: JuniperN_70:40:05 (0c:86:10:70:40:05)						
▼ Internet Protocol Version 4, Src: 100.0.0.23, Dst: 100.0.0.26						
0100 = Version: 4						
.... 0101 = Header Length: 20 bytes (5)						
▼ Differentiated Services Field: 0x48 (DSCP: AF21, ECN: Not-ECT)						
0100 10.. = Differentiated Services Codepoint: Assured Forwarding 21 (18)						
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)						
Total Length: 1028						
Identification: 0x54b6 (21686)						
▶ Flags: 0x00						
Fragment offset: 0						
Time to live: 64						
Protocol: UDP (17)						
Header checksum: 0x59ba [validation disabled]						
[Header checksum status: Unverified]						
Source: 100.0.0.23						
Destination: 100.0.0.26						
[Source GeoIP: Unknown]						
[Destination GeoIP: Unknown]						
▶ User Datagram Protocol, Src Port: 40471, Dst Port: 4789						
▶ Virtual eXtensible Local Area Network						
▶ Ethernet II, Src: 00:00:00_b6:70:f4 (00:00:00:b6:70:f4), Dst: 00:00:00_b6:e9:10 (00:00:00:b6:e9:10)						
▼ Internet Protocol Version 4, Src: 100.0.6.100, Dst: 100.0.6.101						
0100 = Version: 4						
.... 0101 = Header Length: 20 bytes (5)						
▼ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)						
0000 00.. = Differentiated Services Codepoint: Default (0)						
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)						
Total Length: 978						
Identification: 0x0000 (0)						
▶ Flags: 0x00						
Fragment offset: 0						
Time to live: 64						
Protocol: UDP (17)						
Header checksum: 0xa252 [validation disabled]						
[Header checksum status: Unverified]						
Source: 100.0.6.100						
Destination: 100.0.6.101						
[Source GeoIP: Unknown]						
[Destination GeoIP: Unknown]						
▶ User Datagram Protocol, Src Port: 22320, Dst Port: 35940						
▶ Data (950 bytes)						

```
jnpr@SPINE-1> show class-of-service classifier name dscp_classifier
Classifier: dscp_classifier, Code point type: dscp, Index: 65170
```

Code point	Forwarding class	Loss priority
010010	FC_v4-UDP	low

```
jnpr@SPINE-1> show interfaces queue et-0/0/8
Physical interface: et-0/0/8, Enabled, Physical link is Up
Interface index: 664, SNMP ifIndex: 536
Description: * to FABRIC-1
Forwarding classes: 16 supported, 6 in use
Egress queues: 8 supported, 6 in use
```

Queue: 0, Forwarding classes: best-effort

Queued:		
Packets	:	17171500760 29701 pps
Bytes	:	17959010657354 248523104 bps
Transmitted:		
Packets	:	17171500760 29701 pps
Bytes	:	17959010657354 248523104 bps
Tail-dropped packets : Not Available		
RL-dropped packets	:	0 0 pps
RL-dropped bytes	:	0 0 bps
Total-dropped packets:		0 0 pps
Total-dropped bytes :		0 0 bps

Queue: 2, Forwarding classes: FC_v4-UDP

Queued:		
Packets	:	2845686435 59413 pps
Bytes	:	2976588011010 497172936 bps
Transmitted:		
Packets	:	2845686435 59413 pps
Bytes	:	2976588011010 497172936 bps
Tail-dropped packets : Not Available		
RL-dropped packets	:	0 0 pps
RL-dropped bytes	:	0 0 bps
Total-dropped packets:		0 0 pps
Total-dropped bytes :		0 0 bps

Queue: 3, Forwarding classes: FC_v4-1p

Queued:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Transmitted:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Tail-dropped packets : Not Available		
RL-dropped packets	:	0 0 pps
RL-dropped bytes	:	0 0 bps
Total-dropped packets:		0 0 pps
Total-dropped bytes :		0 0 bps

Queue: 4, Forwarding classes: no-loss

Queued:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Transmitted:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Tail-dropped packets : Not Available		
RL-dropped packets	:	0 0 pps
RL-dropped bytes	:	0 0 bps
Total-dropped packets:		0 0 pps
Total-dropped bytes :		0 0 bps

Queue: 5, Forwarding classes: FC_v4-TCP

Queued:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Transmitted:		
Packets	:	0 0 pps
Bytes	:	0 0 bps
Tail-dropped packets : Not Available		
RL-dropped packets	:	0 0 pps
RL-dropped bytes	:	0 0 bps
Total-dropped packets:		0 0 pps
Total-dropped bytes :		0 0 bps

Queue: 7, Forwarding classes: network-control

Queued:

Packets	:	7343901	5 pps
Bytes	:	538919005	3256 bps

Transmitted:

Packets	:	7343901	5 pps
Bytes	:	538919005	3256 bps

Tail-dropped packets : Not Available

RL-dropped packets	:	0	0 pps
--------------------	---	---	-------

RL-dropped bytes	:	0	0 bps
------------------	---	---	-------

Total-dropped packets:	:	0	0 pps
------------------------	---	---	-------

Total-dropped bytes	:	0	0 bps
---------------------	---	---	-------

Chapter 4

Data Center Interconnect – OTT DCI (L3VPN-MPLS Core)

Contents

<i>OTT DCI L3VPN MPLS Core and Traffic Optimization High-Level Summary</i>	<i>161</i>
<i>East-West traffic (Intra-VRF: Intra-subnet and Inter-subnet)</i>	<i>166</i>
<i>North-South Traffic Optimization for Hosts in Different Tenants (Inter-VRF: Inter-subnet)</i>	<i>178</i>



DCI allows for geographically dispersed data centers to be connected, allowing for the extension of Layer 2 and Layer 3 connectivity across DCs. In previous chapters, this book explored different conceptual models and related building blocks on how to build a data center. In this chapter and following chapters, DCI case studies examine connecting different data centers. For brevity, this chapter will not repeat all the traffic scenarios previously explored, but the same concepts can be re-applied to achieve similar results for DCI use cases.

As mentioned before, for any questions regarding existing or roadmap support, please reach out to your account teams.

Each case study presented here has three main sections:

- A high-level summary with an overview on the projected design and products being positioned.
- A design summary with technical details on the involved design components.
- Traffic scenarios that delve into configuration and verification details on common traffic use cases.

Let's get started by examining Figure 4.1 and 4.2, which provide a consolidated view of the data center design and DCI integration discussion at this point in the book.

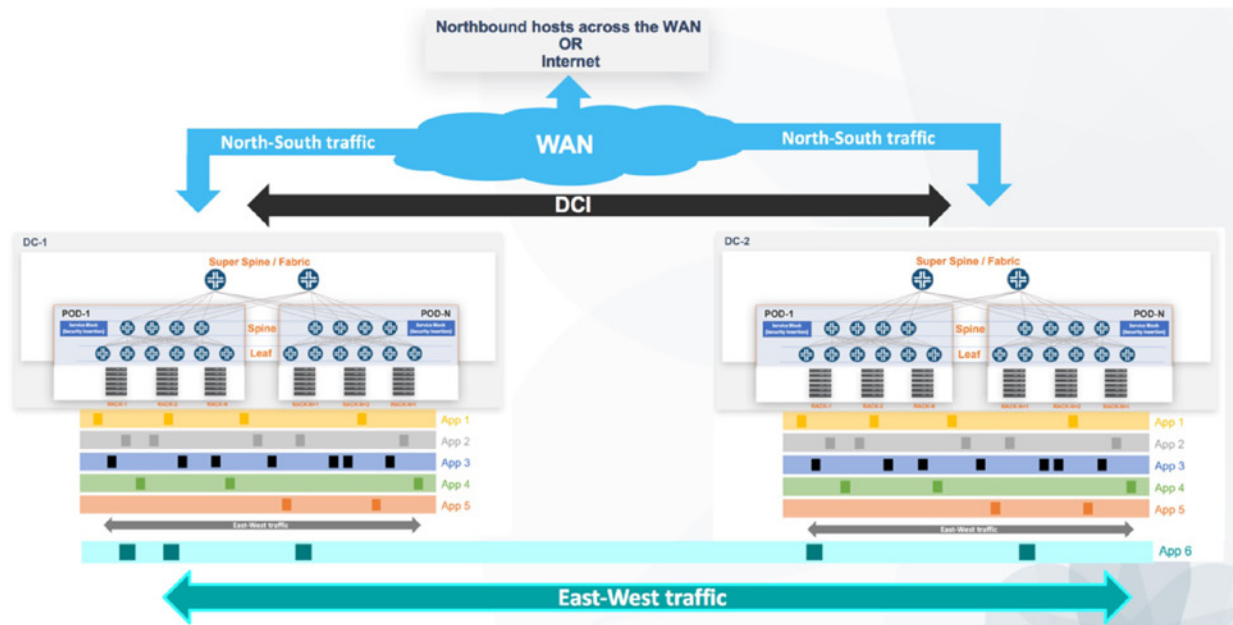


Figure 4.1 DC Design and DCI Integration

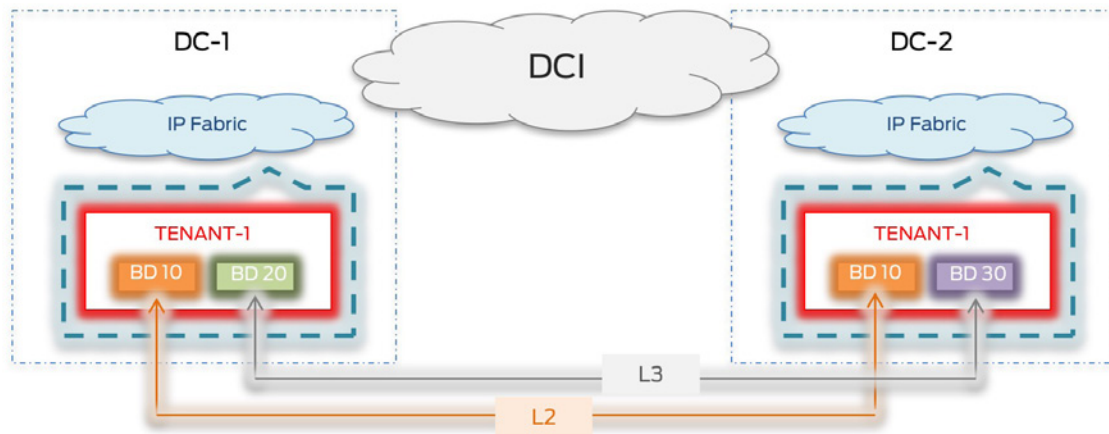


Figure 4.2 DCI – E-W Traffic Support for Layer 2 and Layer 3 Workloads Across DCs

Over the Top (OTT)

With OTT DCI, the control plane is extended across sites with the connecting infrastructure used as transport only (EVPN unaware). These interconnected sites thus act as a single logical data center and the unified EVPN control plane can facilitate both Layer 2 and Layer 3 communication.

Advantage: Ease of implementation makes this a feasible choice for smaller deployments.

Disadvantage: Need for full mesh of VXLAN tunnels across DCs can become a scaling constraint.

Examples: OTT DCI – EVPN/VXLAN over provider L3VPN, public Internet, dark fiber, etc.

OTT DCI

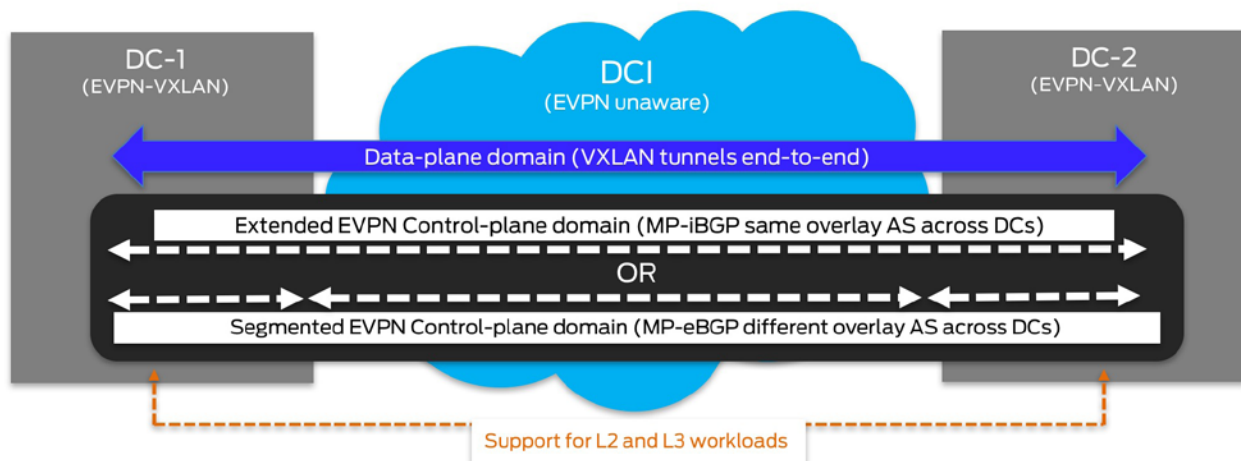


Figure 4.3 OTT DCI

Segmentation of DC and WAN Domains

With this deployment model, a clear demarcation of DC and WAN boundaries is maintained. Infrastructure connecting data center sites now participates in the EVPN control plane. These interconnected sites thus act as independent domains and the unified EVPN control plane can facilitate both Layer 2 and Layer 3 communication.

Advantage: No need for full mesh of VXLAN tunnels across DCs (makes this a feasible choice for larger deployments); MPLS core can be leveraged for traffic engineering of tenant traffic.

Disadvantage: Data plane encapsulation translation at DC boundaries can lead to some provisioning complexity.

Examples: EVPN-VXLAN (inside DC) to EVPN-MPLS (inside WAN – MPLS core) stitching, EVPN-VXLAN (inside DC) to EVPN-VXLAN (inside WAN – IP core).

DCI with Data Plane Stitching

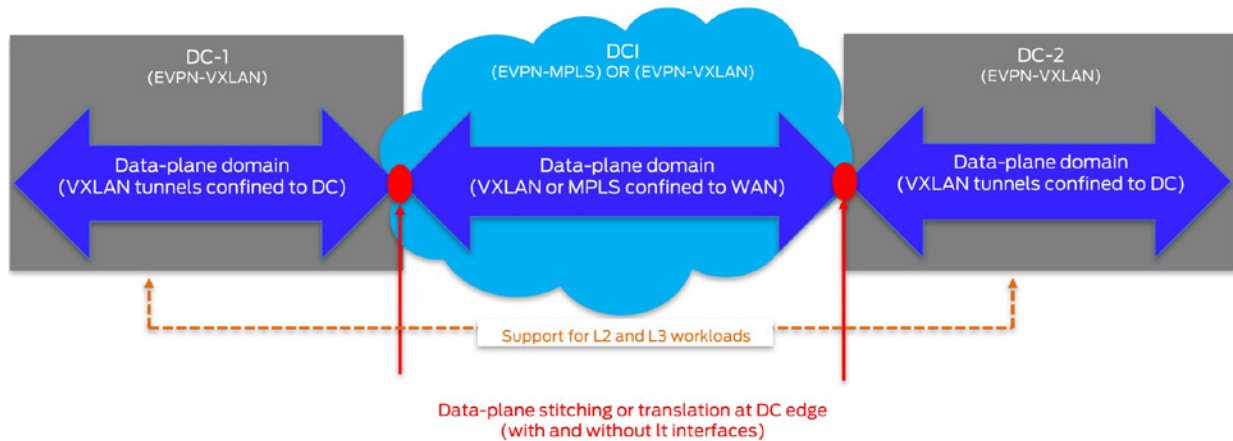


Figure 4.4 DCI with Data Plane Stitching

Layer 3 DCI

With this deployment model, only Layer 3 connectivity is extended across DCs (no Layer 2). EVPN (host IP/summary) routes inside a data center are re-originated as IP routes into L3VPN towards the WAN. EVPN control plane is confined within DC boundaries and not extended across DCs, nor is EVPN used in the core.

Advantage: Simplified implementation for Layer 3 connectivity can be obtained while still leveraging the benefits associated with an EVPN control plane inside the DC.

Disadvantage: No Layer 2 connectivity.

Examples: Border gateways that serve as exit points out of the data center and connect to an L3VPN WAN.

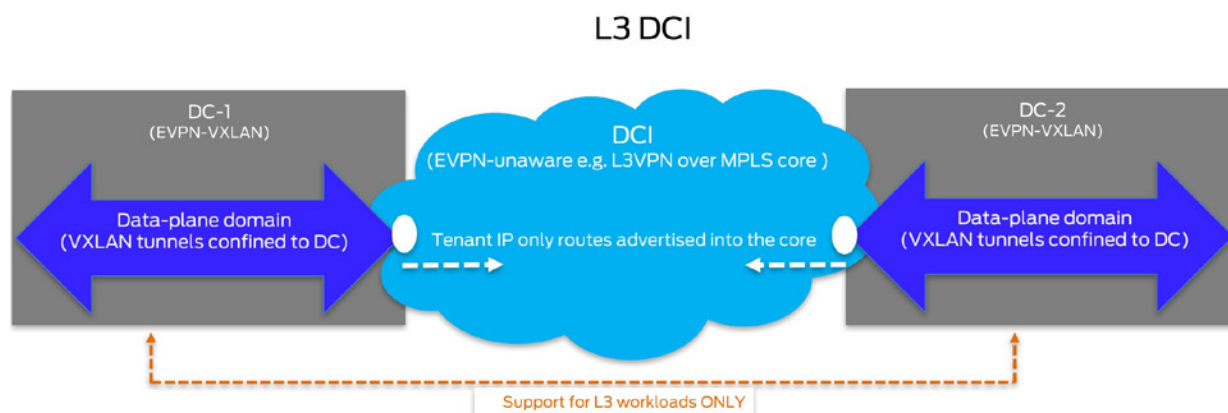


Figure 4.5 Level 3 DCI

DCI Case Studies

Chapters 4, 5, and 6 each include a different DCI deployment scenario that demonstrates the application of each concept, including key design, configuration, and verification highlights provided for each case study:

- OTT DCI – L3VPN-MPLS core and traffic optimization
- OTT DCI – Public Internet
- Layer 3 DCI – Integration with L3VPN-MPLS core

OTT DCI L3VPN MPLS Core and Traffic Optimization High-Level Summary

In this design approach, similar to the case study in Chapter 2, routing is done at the leaf layer within each data center thus making it possible to reduce functionality and scale requirements at the spine layer. Spine devices act only as transit and overlay route reflectors.

The case study in Chapter 2 demonstrated the use of border leaf devices that act as a data center edge, connecting the data center to the WAN. However, for better scale and performance, demarcation of responsibility can be achieved by using separate fabric/super-spine devices to act as the DC edge. Fewer and more powerful fabric/super-spine devices can be used instead, which reduces functional and scale requirements on border leaf devices that are needed per POD. The QFX 10000 or the MX Series can act as DC edge devices that host the intelligence or control plane logic with necessary scaling capabilities to perform additional DCI functions.

Not demonstrated here, but as described in Chapter 2, border leaf devices can still act as service nodes for each POD, to direct both E-W and N-S traffic for manual service chaining by the DC firewall. The QFX5110 can be used both as border leaf and leaf devices acting as Layer 3 gateways. A typical deployment example could use the QFX 5110 as leaf/border leaf devices, QFX 5200 as spine devices, and the MX Series or QFX 10000 Series as DC edge devices.

This particular case study (shown in Figure 4.6) focuses only on the DCI aspect on how the different data centers connect to the WAN enabling East-West (EW) traffic flows across tenant hosts in different data centers (Inter-DC), separated by a L3VPN-MPLS backbone.

EVPN-VXLAN is set up between data centers over the top (OTT) of an existing WAN network to demonstrate both intra-subnet (Layer 2 domain stretched across data centers) and inter-subnet communication between hosts. North-South (N-S) traffic between hosts across the WAN entering and exiting the data center boundary have also been set up. Additionally, this case study also addresses traffic optimization to counter traffic tromboning otherwise seen on Layer 2 domain extension across data centers.

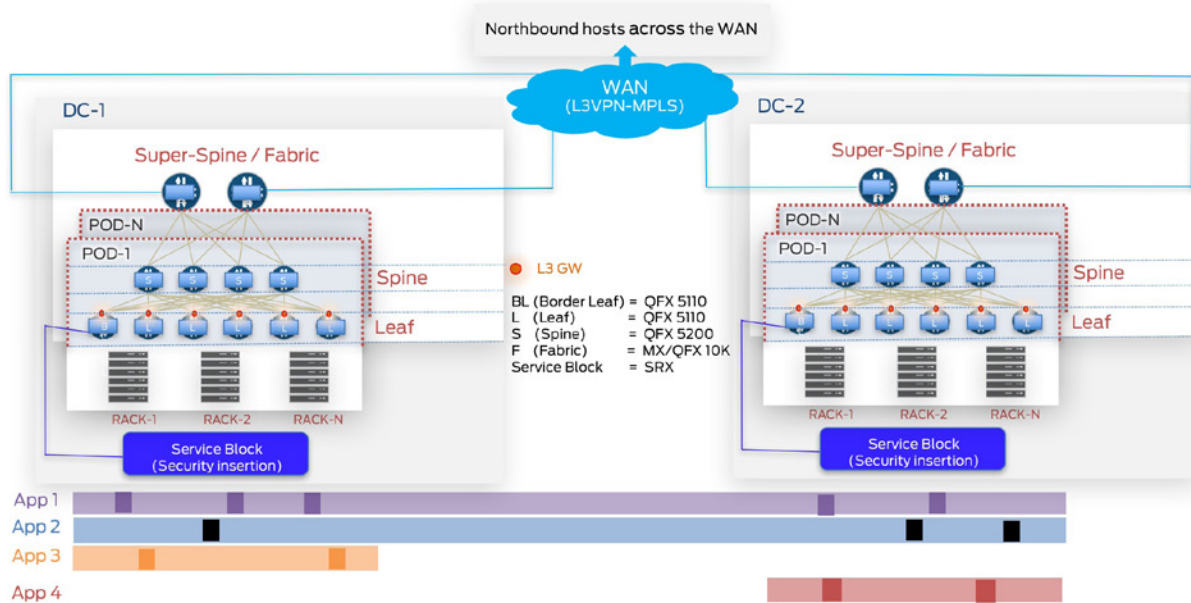


Figure 4.6 High-Level Design Representation

Topology Description

Leaf devices: Due to resource constraints, the QFX10002-36Qs have both been used as leaf devices. Border leaf devices have not been simulated in this testbed. However, design considerations from Chapter 2 can be reused here. Each POD consists of two leaf devices.

Spine devices: One QFX10002-36Q has been used as the spine device, in each POD in every DC.

Fabric devices: One QFX10002-36Q has been used as fabric/super-spine device to connect all PODs in each DC.

Service block: Due to resource constraints, service block (firewall) has not been simulated in this example. However, design considerations from the case study in Chapter 2 can be reused here. As described earlier, border leaf devices in every POD can be used as service nodes to connect to the DC firewall.

CE devices: Access switches (the QFX5100-48S-6Q – acting as VC in POD-1 and standalone switch in POD-2) and the IXIA simulate CE devices in both PODs in both DCs. Servers can be directly plugged in to the TOR leaf devices. To demonstrate the use of LACP, intermediate switches have been used here due to resource constraints. The term *host* (simulated on IXIA) implies any application entity (VM or container) that consumes an IP/MAC address.

WAN: Three MX104 routers have been used to simulate WAN PEs, while the QFX5100-48S-6Q has been used as a P device.

Chapter 2 outlined general data center layout assumptions and for brevity the same will *not* be repeated here. In summary, ten PODs each consisting of ten rows of ten racks each, constitute a data center. Figure 4.7 portrays these assumptions that have been logically mapped to the physical testbed simulated in the lab for this case study.

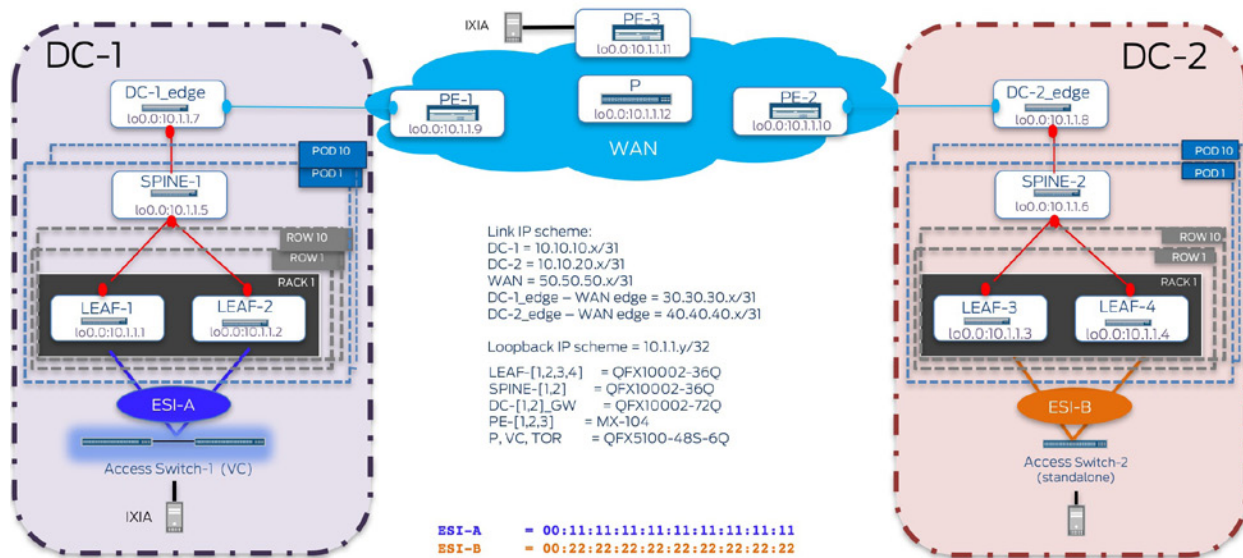


Figure 4.7 Physical Topology

Design Summary

Traffic flows demonstrated:

- East-West (traffic across PODs within the DC):
 - Layer 2 (Intra-VRF Intra-subnet): Traffic between hosts in the same subnet for a given tenant.
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Not demonstrated in this case study, but the same building blocks as those described in the case study from Chapter 2 can be used.
 - North-South (traffic between hosts across the WAN and within the DC).
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant, within data centers and across the WAN.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Not demonstrated in this case study, but the same building blocks as those described in the case study from Chapter 2 can be used.

Access considerations:

- Two leaf devices are acting as a pair of redundant Top of Rack switches in each DC. CE-1 represents groups of hosts (e.g. VMs/containers residing on servers) mapped

to ESI-A (00:11:11:11:11:11:11:11:11:11) on ae0 for LEAF-1 and LEAF-2 in DC-1. Similarly, hosts on CE-2 are mapped to ESI-B (00:22:22:22:22:22:22:22:22:22) on ae0 for LEAF-3 and LEAF-4 in DC-2.

Underlay considerations:

- ISIS has been used to achieve underlay IP reachability (lo0 address reachability between VTEPs).
- DC-1 uses ISIS Level 1 (Area 1) and DC-2 uses ISIS Level 1 (Area 2). WAN devices use ISIS Level 2 (Area 3), which does not extend up to the DC edge devices, therefore, ISIS domains in the two data centers are discontinuous and are not connected by a common ISIS Level 2 backbone.
- MTU has been set on all physical interfaces for devices in a data center to account for VXLAN encapsulation.

Overlay considerations:

- L3 gateway placement:
 - VXLAN routing in this use case is done on the leaf devices. Each leaf device acts as a Layer 3 gateway for VNIs residing in that POD.
 - Routing is done at the leaf layer (consolidated L2 and L3 gateway functionality in a single device).
- EVPN NLRI exchange:
 - MP-iBGP sessions have been used to exchange EVPN NLRI within the data center.
 - To support EW Inter-DC communication, the EVPN/VXLAN domain within each data center is extended by means of a EVPN MP-eBGP (multihop) session between DC gateways to exchange routes learned in each data center. This multihop EVPN MP-eBGP session is established between the loopback addresses (e.g. lo0.0) of the DC edge devices. More details on this will be provided in the subsequent sections, but MP-eBGP multihop sessions between DC edge devices have been created such that there is no change in the protocol next-hop of advertised routes. As such, data plane domain or VXLAN tunnels still exist end-to-end across DCs, as in a OTT DCI scenario. Using eBGP (EVPN) however, provides visibility into the AS from which a route is advertised providing operational ease.
- Route reflection:
 - To avoid full-mesh of control-plane connections (EVPN MP-iBGP sessions within each data center), MP-iBGP EVPN sessions have been created between the leaf devices functioning as clients and spine devices being overlay route reflectors, responsible for the exchange of EVPN NLRI. In DC-1 (POD-1), LEAF-1 and LEAF-2 act as clients to SPINE-1 serving as RRs (cluster ID 11.11.11.11 in DC-1). In DC-2 (POD-1), LEAF-3 and LEAF-4 act as clients to SPINE-3 serving as RRs (cluster ID 22.22.22.22 in DC-2). Not shown here, but spine devices with different cluster IDs can provide for added redundancy.
 - Hierarchical route reflection for the overlay is in use. Fabric devices act as route reflectors thus preventing a full-mesh of BGP sessions between spine devices (cluster-ID 1.1.1.1 in DC-1 and cluster-ID 2.2.2.2 in DC-2).

- Full-mesh of VXLAN tunnels (data-plane) exist between all the Layer 2/ Layer 3 gateways (leaf devices here) in each DC.
- Service interface:
 - VLAN aware EVPN service is in use (please refer to the appendix for more details on EVPN service interfaces).
- Host communication:
 - Inter-DC E-W traffic flows demonstrate both Layer 2 extension and Layer 3 communication between same tenant hosts (Intra-VRF) across data centers. Type 2 routes (asymmetric) are used to exchange reachability information for VLANs 10 and 20 (mapped to VNI 5010 and 5020, respectively) stretched across DCs, allowing for both intra-subnet and inter-subnet communication between these hosts.
 - Layer 2: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant) --- > Type 2
 - Layer 3: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant) --- > Type 2
 - N-S traffic flows demonstrate Layer 3 communication between northbound host across the WAN and tenant host in either data center. Within the data center, EVPN Type 5 routes are used to advertise host routes between leaf devices and DC edge devices. IP-VRFs are configured on leaf and fabric devices to exchange EVPN Type 5 host routes. WAN edge devices with interested CEs have the relevant IP-VRF configuration in which DC host routes are learned via L3VPN.
 - Both Type 2 and Type 5 routes have been used to demonstrate inter-subnet communication. However, that is not a requirement and either type can be used for the same purpose.

Integration with the WAN:

- EVPN unaware core:
 - WAN devices, interconnecting the different data centers are only used for transport. WAN does not run EVPN but only assists in distributing loopback addresses required for establishing the control and data plane connections across DCs. To achieve this, each DC-edge establishes an eBGP (family inet) session with its connecting WAN edge, thus acting as CEs to the L3VPN PEs. Relevant WAN edge PEs require only one IP-VRF ('VRF_DCI_lo0s' here) to exchange the loopback addresses of the connecting DC. Loopback addresses of the Layer 2/ Layer 3 gateways and DC gateways are exchanged to allow for the creation of the full mesh of EVPN/VXLAN tunnels across DCs. LDP is used in the L3VPN backbone with a full mesh of MP-iBGP inet-vpn sessions between WAN PEs. Tenant IP-VRFs are only needed on the EVPN PEs (leaf devices acting as Layer 3 gateways) for inter-DC E-W communication.
- Traffic optimization:
 - N-S traffic optimization is being demonstrated for VLANs that are stretched across DCs, with no traffic tromboning to reach a specific tenant host. This exact host location awareness exists since hosts routes are advertised by fabric devices in each DC to the WAN, for example, when H0 attached to PE-3 sends traffic to DC-1 to communicate with H1, there is no traffic tromboning via DC-2 despite the same subnet being stretched across data centers.

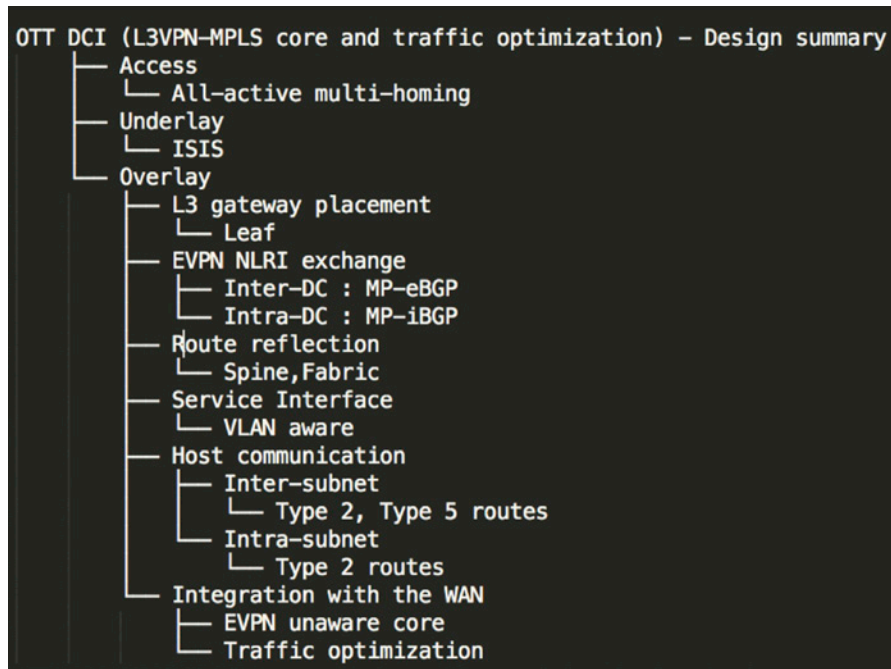


Figure 4.8 Design Summary for OTT DCI – L3VPN Core

Traffic Scenarios

The rest of this chapter demonstrates implementation details for two primary use cases:

1. East-West traffic for hosts in the same tenant (Intra-VRF; Intra-subnet and Inter-subnet)

- Configuration
- Verification
- Traffic flows

2. North-South traffic optimization for hosts in different tenants (Inter-VRF; Inter-subnet)

- Configuration
- Verification
- Traffic flows

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Figures 4.9 and 4.10 illustrate the logical topology for E-W traffic across PODs as summarized in the design highlights. The rest of the chapter details the illustrated configuration and verification steps.

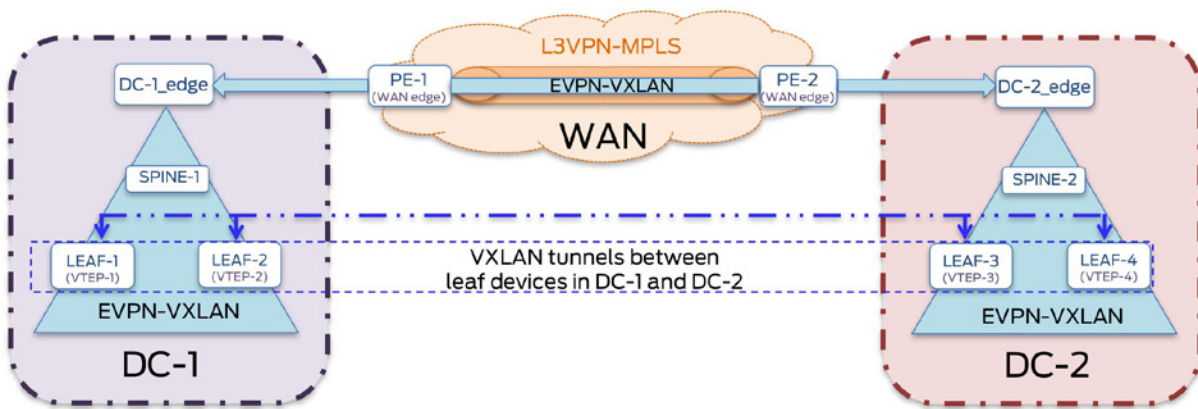


Figure 4.9 DCI Logical Topology (High Level Overview)

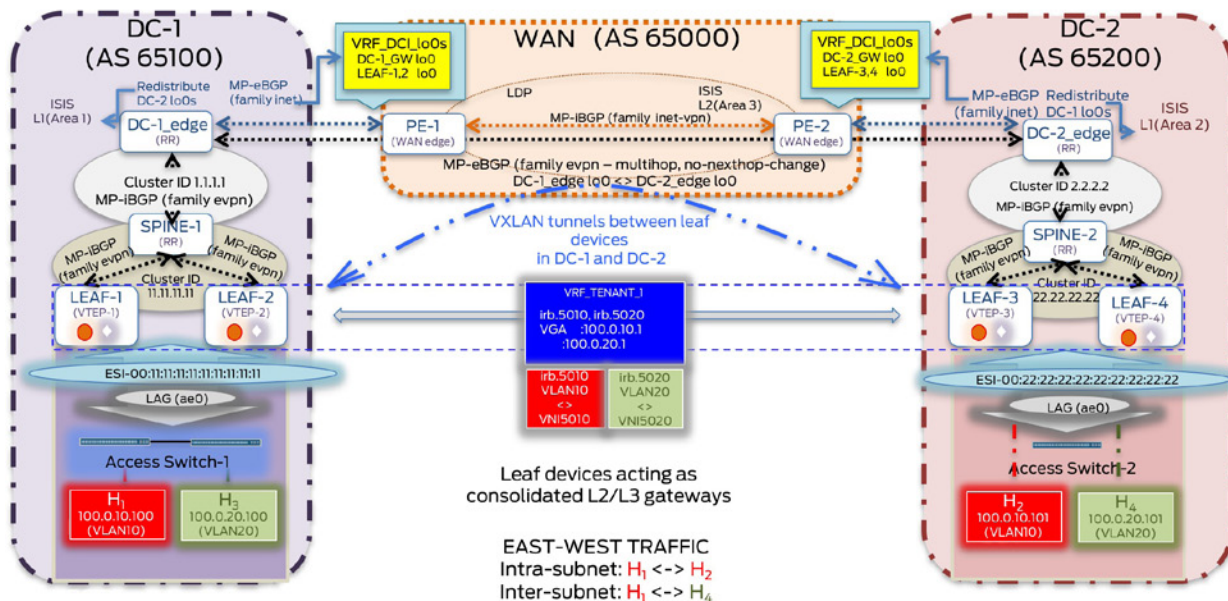


Figure 4.10 DCI Logical Topology (Detailed Overview)

Configuration

In this section, configuration is divided across multiple groups, which are applied to the different devices as needed. Configuration highlights have been outlined in cases where they differ from those as described in Chapter 2. Snippets below show configuration highlights for the leaf (LEAF-1), spine (SPINE-1) and fabric (DC-1_edge) layer devices across DC-1 and DC-2.

Leaf Devices

Configuration on the leaf devices is divided into three components – underlay, overlay (intra-DC), and access as shown in Figure 4.11. Three configuration groups have been applied to all leaf devices in both DC-1 and DC-2 – UC3-EW-Access, UC3-EW-Underlay, and UC3-EW-IntraDC.

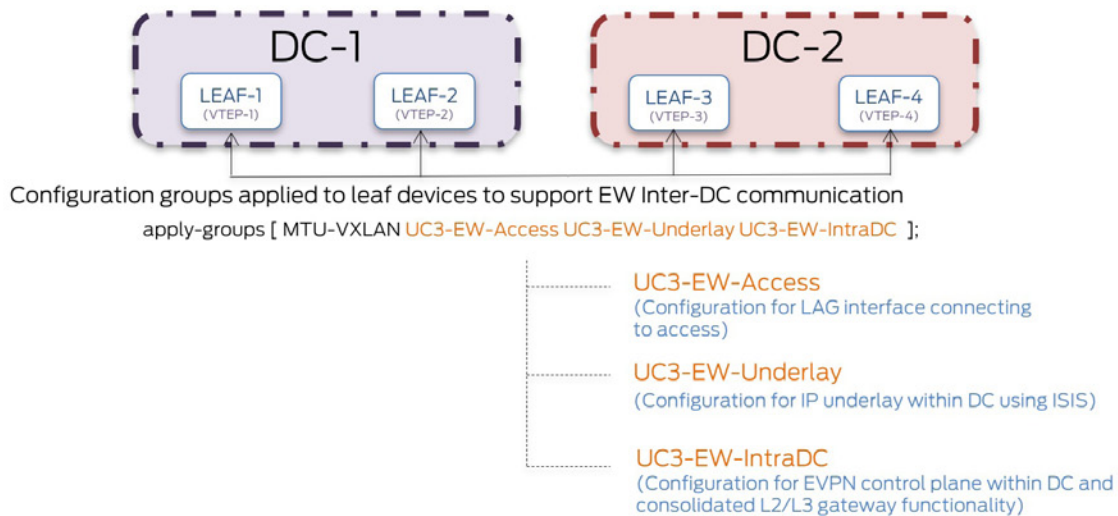


Figure 4.11 DCI Configuration – Leaf Layer

UC3-EW-Access: Configuration components for access remain unchanged from the previous case studies, shown here is the output on LEAF-1.

```
jnpr@LEAF-1> show configuration groups UC3-EW-Access
interfaces {
  et-0/0/30 {...}
  et-0/0/31 {...}
  ae0 {
    description „LAG VC <> LEAF-1”;
    mtu 9192;
    esi {
      00:11:11:11:11:11:11:11:11;
      all-active;
    }
    aggregated-ether-options {
      lacp {
        active;
        periodic fast;
        system-id 01:01:01:01:01:01;
      }
    }
  }
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
      vlan {
        members all;
      }
    }
  }
}
```

UC3-EW-Underlay: Devices in DC-1 are assigned AS#65100, while those in DC-2 are assigned AS#65200. Each data center is a separate ISIS Level 1 domain (DC-1 = ISIS L1 Area 49.001 or Area 1, DC-2 = ISIS L1 Area 49.0002 or Area 2).

```
jnpr@LEAF-1> show configuration groups UC3-EW-Underlay
interfaces {
  lo0 {
    unit 0 {
      family inet {...}
      family iso {
        address 49.0001.0010.0100.1001.00;
      }
    }
  }
}
```

```

    {...}
}
routing-options {
    autonomous-system 65100;
    {...}
}
protocols {
    isis {
        Layer 2 disable;
        interface all;
        interface lo0.0 {
            passive;
        }
    }
}
policy-options {...}
}

```

UC3-EW-IntraDC: MP-iBGP sessions (EVPN NLRI only) exist between leaf devices and spine/route reflector devices in each DC. Leaf/VTEP devices are acting as consolidated Layer 2/Layer 3 gateways. EVI/MAC-VRF and IP-VRF configuration remains the same as that discussed in the previous case studies.

```

jnpr@LEAF-1> show configuration groups UC3-EW-IntraDC
interfaces {
    irb {
        unit 5010 {...}
        unit 5020 {...}
    }
}
protocols {
    bgp {
        group IBGP-EVPN-DC1 {
            type internal;
            description "Leaf clients for Spine RR";
            local-address 10.1.1.1;
            local-as 65100;
            bfd-liveness-detection {
                minimum-interval 350;
                multiplier 3;
                session-mode automatic;
            }
            multipath;
            neighbor 10.1.1.5 {
                family evpn {
                    signaling;
                }
            }
        }
    }
}
evpn {...}
}
policy-options {
    policy-statement EVPN-IMPORT {...}
}
routing-instances {
    VRF_TENANT_1 {
        instance-type vrf;
        interface irb.5010;
        interface irb.5020;
        interface lo0.10;
        route-distinguisher 10.1.1.10:10;
        vrf-target target:10:10;
        vrf-table-label;
    }
}
switch-options {...}
vpls {
    bd5010 {...}
    bd5020 {...}
}
}

```

Spine Devices

Configuration on the spine devices is divided into two components – underlay and overlay (intra-DC) as shown in Figure 4.12. Two configuration groups have been applied to all leaf devices in both DC-1 and DC-2 – UC3-EW-Underlay, UC3-EW-IntraDC.

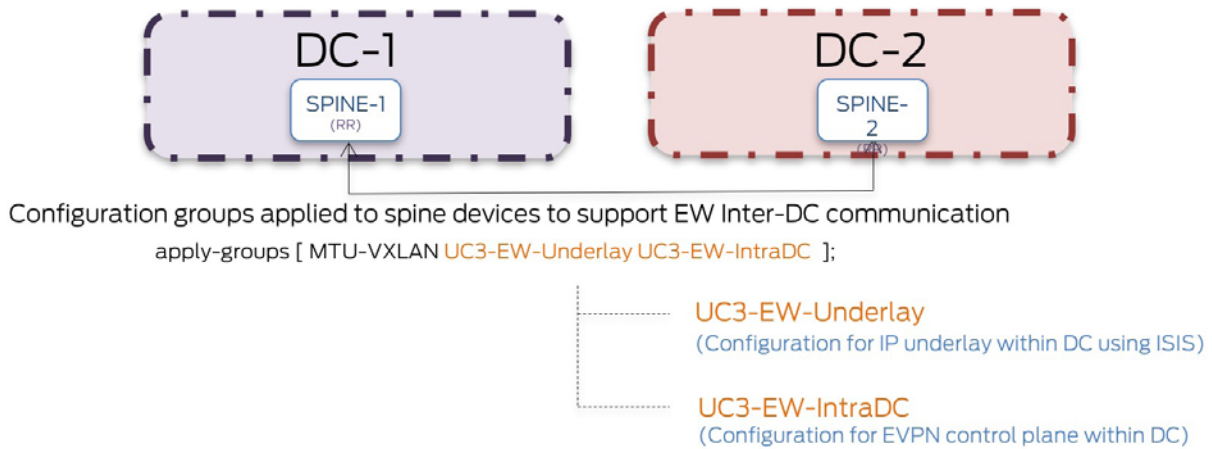


Figure 4.12 DCI Configuration – Spine layer

UC3-EW-Underlay: Devices in DC-1 are assigned AS#65100, while those in DC-2 are assigned AS#65200. Each data center is a separate ISIS Level 1 domain (DC-1 = ISIS L1 Area 49.0001 or Area 1, DC-2 = ISIS L1 Area 49.0002 or Area 2).

```
jnpr@DC-1_GW> show configuration groups UC3-EW-Underlay
interfaces {
  lo0 {
    unit 0 {
      family inet {...}
      family iso { address 49.0001.0070.0700.7007.00; }
      {...}
    }
  }
  routing-options {
    autonomous-system 65100;
    {...}
  }
  protocols {
    isis {
      interface et-0/0/0.0 {
        Layer 2 disable;
      }
      interface em0.0 {
        disable;
      }
      interface lo0.0 {
        passive;
        Layer 2 disable;
      }
    }
  }
}
policy-options {...}
}
```

UC3-EW-IntraDC: Spine devices in each DC act as overlay route-reflectors with leaf devices as clients. These further act as client to fabric devices for hierarchical route reflection. This can provide added redundancy when different cluster IDs are in use on spine devices.

```
jnpr@SPINE-1> show configuration groups UC3-EW-IntraDC
protocols {
  bgp {
    group IBGP-EVPN-DC1-RR {
      type internal;
      description "Spine RR for Leaf clients";
      local-address 10.1.1.5;
      family evpn {
        signaling;
      }
      cluster 11.11.11.11;
      local-as 65100;
      bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
      }
      multipath;
      neighbor 10.1.1.1;
      neighbor 10.1.1.2;
    }
  }
  group IBGP-EVPN-DC1 {
    type internal;
    description "Spine client for Fabric RR";
    local-address 10.1.1.5;
    family evpn {
      signaling;
    }
    local-as 65100;
    bfd-liveness-detection {
      minimum-interval 350;
      multiplier 3;
      session-mode automatic;
    }
    multipath;
    neighbor 10.1.1.7;
  }
}
```

Spine devices in each DC act as overlay route reflectors with leaf devices as clients. These further act as client-to-fabric devices for hierarchical route reflection, providing added redundancy when different cluster IDs are in use on spine devices.

Fabric Devices

Configuration on the fabric devices is divided into four components – underlay, overlay (intra-DC), overlay (inter-DC), and integration with L3VPN core. Four configuration groups have been applied to all fabric devices for the same in both DC-1 and DC-2 – UC3-EW-Underlay, UC3-EW-IntraDC, UC3-EW-L3VPNcore, and UC3-EW-DCI as shown in Figure 4.13.

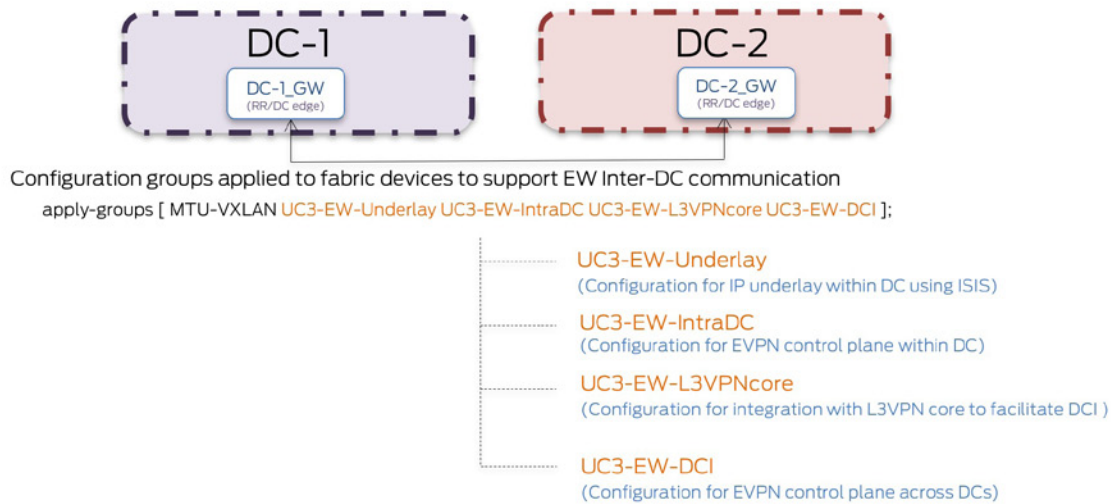


Figure 4.13 DCI Configuration – Fabric Layer

UC3-EW-Underlay: Devices in DC-1 are assigned AS#65100, while those in DC-2 are assigned AS#65200. Each DC edge participates only in ISIS Layer 1 domain of its respective data center (DC-1 = ISIS L1 Area 49.001 or Area 1, DC-2 = ISIS L1 Area 49.0002 or Area 2).

```
jnpr@DC-1_GW> show configuration groups UC3-EW-Underlay
interfaces {
  lo0 {
    unit 0 {
      family inet {...}
      family iso { address 49.0001.0070.0700.7007.00; }
      {...}
    }
  }
  routing-options {
    autonomous-system 65100;
    {...}
  }
  protocols {
    isis {
      interface et-0/0/0.0 {
        Layer 2 disable;
      }
      interface em0.0 {
        disable;
      }
      interface lo0.0 {
        passive;
        Layer 2 disable;
      }
    }
  }
}
policy-options {...}
}
```

UC3-EW-IntraDC: Each DC gateway acts as the overlay route-reflector with spine devices as clients.

```
jnpr@DC-1_GW> show configuration groups UC3-EW-IntraDC
protocols {
  bgp {
    group IBGP-EVPN-DC1-RR {
      type internal;
      description "Fabric RR for spine clients";
      local-address 10.1.1.7;
    }
  }
}
```

```

        family evpn {
            signaling;
        }
        inactive: export nh-self;
        cluster 1.1.1.1;
        local-as 65100;
        bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode automatic;
        }
        multipath;
        neighbor 10.1.1.5;
    }
}
}

```

UC3-EW-L3VPNcore: Each DC edge advertises loopback addresses of its connected data center through an EBGp session to the connected WAN edge device, thus acting as a CE to the L3VPN PE.

```

jnpr@DC-1_GW> show configuration groups UC3-EW-L3VPNcore interfaces {
    xe-0/0/34:0 {
        description „To PE-1 (WAN edge)”;
        vlan-tagging;
        unit 0 {
            description “To PE-1 (DCI EW)”;
            vlan-id 301;
            family inet {
                address 30.30.30.1/31;
            }
            family iso;
        }
        unit 1 {
            description “To PE-1 (NS)”;
            vlan-id 303;
            family inet {
                address 30.30.30.3/31;
            }
        }
    }
}
policy-options {
    prefix-list local-DC-lo0s {
        10.1.1.1/32;
        10.1.1.2/32;
        10.1.1.7/32;
    }
}
policy-statement adv-local-lo0s {
    term 1 {
        from {
            prefix-list local-DC-lo0s;
        }
        then accept;
    }
}
... }
protocols {
    bgp {
        group DC-GW_WAN-Edge {
            type external;
            description “EBGP peering DC GW <> WAN edge”;
            local-as 65100;
            neighbor 30.30.30.2 {
                family inet {
                    unicast;
                }
                export adv-local-lo0s;
                peer-as 65000;
            }
        }
    }
}
}

```

UC3-EW-DCI:

```

jnpr@DC-1_GW> show configuration groups UC3-EW-DCI
protocols {
  bgp {
    group EBG-P-EVPN-DCI {
      type external;
      description "EVPN session between DC-GWs";
      multihop {
        ttl 255;
        no-nexthop-change;
      }
      local-address 10.1.1.7;
      family evpn {
        signaling;
      }
      bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
      }
      multipath;
      neighbor 10.1.1.8 {
        peer-as 65200;
      }
    }
    ...
  }
  isis {
    export adv-remote-lo0s;
  }
}
policy-options {
  prefix-list remote-DC-lo0s {
    10.1.1.3/32;
    10.1.1.4/32;
  }
  policy-statement adv-remote-lo0s {
    term 1 {
      from {
        prefix-list remote-DC-lo0s;
      }
      then accept;
    }
  }
}

```

The EVPN-VXLAN domain is extended by means of multihop MP-eBGP EVPN session between DC gateways to allow for the full mesh of VXLAN tunnels to be created across DCs. The Junos OS `no-nexthop-change` knob ensures that the original protocol next hop on the EVPN NLRI is preserved across DC boundaries. For example, the protocol next hop of an EVPN route originated by Leaf-1 in DC-1 will be unchanged when advertised across DCs and will be seen by leaf devices in DC-2. This is to ensure that the mass withdraw mechanism does not break as it is now possible to identify the originating EVPN PE even across DC boundaries (see [draft-ietf-bess-evpn-overlay-07](#)).

Since the next hop of a route needs to be reachable for it to be actively used, in addition to the lo0s of the DC-gateways, lo0 addresses of EVPN PEs (leaf devices here) are transported through the WAN (learned through the EBGp session between DC and WAN edge devices) between DC-1 and DC-2. Each DC-gateway further re-advertises these into the underlay for the respective data center (using ISIS). As elaborated upon in previous sections, VTEP discovery with the use of EVPN is control-plane based and is influenced by the protocol next hop of the received EVPN NLRI from the remote BGP neighbor. In this case, VTEPs are located only on each leaf node and full-mesh of VXLAN tunnels exist between all VTEPs in both DC-1 and DC-2.

WAN Devices

Configuration on the WAN devices is that of a typical L3VPN core. To support Inter-DC EW communication, one configuration group has been applied to PE devices connecting both DC-1 and DC-2 – UC3-EW-WAN, as shown in Figure 4.14.

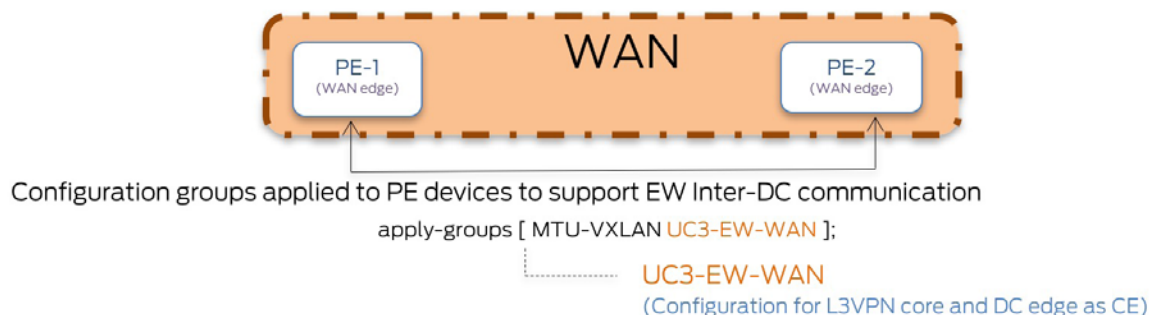


Figure 4.14 WAN Configuration (Inter-DC EW)

UC3-EW-WAN: WAN devices do not run EVPN and are configured for a typical L3VPN core. To allow for the full mesh of EVPN-VXLAN tunnels across DCs, each connecting WAN edge device (PE-1 and PE-2) terminates the EBGp session for DC edge devices acting as CEs to transport data center loopback addresses.

```
jnpr@PE-1> show configuration groups UC3-EW-WAN
interfaces {
  xe-0/0/0 {
    description "To DC-1_GW";
    vlan-tagging;
    unit 1 {
      description "To exchange lo0s for DCI - EW";
      vlan-id 303;
      family inet {
        address 30.30.30.2/31;
      }
    }
  }
  {...}
}
protocols {
  bgp {
    group L3VPN-PEs {
      type internal;
      local-address 10.1.1.9;
      family inet-vpn {
        unicast;
      }
      bfd-liveness-detection {...}
      multipath;
      neighbor 10.1.1.10 {
        export nhs;
      }
      neighbor 10.1.1.11;
    }
    group DC-GW_WAN-Edge {
      type external;
      peer-as 65100;
      neighbor 30.30.30.1;
    }
  }
}
isis {
  interface xe-0/1/0.0 {
    level 1 disable;
  }
  interface lo0.0 {
    passive;
  }
}
```

```

        level 1 disable;
    {... }
}
l3dp {
    interface xe-0/1/0.0;
}
routing-options {...}
policy-options {...}
routing-instances {
    VRF_DCI_lo0s {
        instance-type vrf;
        interface xe-0/0/0.1;
        route-distinguisher 10.1.1.9:1;
        vrf-target target:65120:1;
        protocols {
            bgp {
                group CE_L3VPN {
                    type external;
                    neighbor 30.30.30.3 {
                        family inet { unicast; }
                    }
                }
            }
        }
    }
}
{...}

```

Verification

Leaf-1 and Leaf-2 in DC-1 learn MAC/IP addresses for local hosts H1 and H3. Similarly, MAC address for remote hosts H2 and H4 are learned from the remote VTEPs Leaf-3 and Leaf-4 in DC-2, that advertise Type 2 routes for the same.

```

jnpr@LEAF-1> show ethernet-switching table
Vlan name      MAC address      MAC flags  Logical interface  Active source
bd5010         00:00:1e:63:c8:7c  DLR        ae0.0
bd5010        00:00:1e:63:d1:fd  DR        esi.1751         00:22:22:22:22:22:22:22:22
bd5010         00:00:5e:00:01:01  DR         esi.1765          05:00:00:fe:4c:00:00:13:92:00
bd5020         00:00:00:93:3c:f3  DR         ae0.0
bd5020        00:00:00:93:3c:f4  DR        esi.1751         00:22:22:22:22:22:22:22:22
bd5020         00:00:5e:00:01:01  DR         esi.1764          05:00:00:fe:4c:00:00:13:9c:00

```

DC-2 host MAC addresses are learned by DC-1 VTEPs and vice versa (output for Leaf-1 shown here).

Protocol next hop on routes is preserved across DC boundaries.

```

jnpr@LEAF-1> show route table bgp.evpn.0 evpn-mac-address 00:00:00:93:3c:f4 detail
bgp.evpn.0: 38 destinations, 38 routes (38 active, 0 holddown, 0 hidden)
2:10.1.1.3:100::5020::00:00:00:93:3c:f4/304 (1 entry, 0 announced)
    *BGP      Preference: 170/-101
              Route Distinguisher: 10.1.1.3:100
              Next hop type: Indirect, Next hop index: 0
              Address: 0xa5bd5f0
              Next-hop reference count: 30
              Source: 10.1.1.5
              Protocol next hop: 10.1.1.3
              <...>

2:10.1.1.4:100::5020::00:00:00:93:3c:f4/304 (1 entry, 0 announced)
    *BGP      Preference: 170/-101
              Route Distinguisher: 10.1.1.4:100
              Next hop type: Indirect, Next hop index: 0
              Address: 0xa5bdd70
              Next-hop reference count: 26
              Source: 10.1.1.5
              Protocol next hop: 10.1.1.4
              <...>

```

VTEPs in DC-1 see originating VTEPs as the protocol next hop for routes received from DC-2 (here, Leaf-1 sees Leaf-3 and Leaf-4, as the protocol next hop for host H2 route received from DC-2).

Full mesh of VXLAN tunnels exist across VTEPs in both DC-1 and DC-2.

```
jnpr@LEAF-1> show interfaces vtep | match "Logical interface|Endpoint Type"
Logical interface vtep.32768 (Index 552) (SNMP ifIndex 534)
  VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.1.1.1, L2 Routing Instance:
Logical interface vtep.32771 (Index 565) (SNMP ifIndex 549)
  VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.2, L2 Routing Instance:
Logical interface vtep.32769 (Index 563) (SNMP ifIndex 547)
  VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.3, L2 Routing Instance:
Logical interface vtep.32770 (Index 564) (SNMP ifIndex 548)
  VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.4, L2 Routing Instance:
```

Traffic Flows

For this use case, both PODs are situated in different data centers (Inter DC) and all hosts belong to the same tenant (Intra VRF). Hosts H1 and H2 belong to VLAN 10 mapped to VNI 5010, while hosts H3 and H4 belong to VLAN 20 mapped to VNI 5020. All hosts share the same IP-VRF (VRF_TENANT_1).

```
Host 1 (H1) : IP1 = 100.0.10.100, MAC1 = 00:00:1e:63:c8:7c is multihomed to Leaf-1 and Leaf-2
              (ESI-1 = 00:11:11:11:11:11:11:11:11:11:11:11)
Host 2 (H2) : IP2 = 100.0.10.101, MAC2 = 00:00:1e:63:d1:fd is multihomed to Leaf-3 and Leaf-4
              (ESI-2 = 00:22:22:22:22:22:22:22:22:22:22:22)
Host 3 (H3) : IP3 = 100.0.20.100, MAC3 = 00:00:00:93:3c:f3 is multihomed to Leaf-1 and Leaf-2
              (ESI-1 = 00:11:11:11:11:11:11:11:11:11:11:11)
Host 4 (H4) : IP4 = 100.0.20.101, MAC4 = 00:00:00:93:3c:f4 is multihomed to Leaf-3 and Leaf-4
              (ESI-2 = 00:22:22:22:22:22:22:22:22:22:22:22)
Default gateway : 100.0.10.1 (virtual-gateway-address for irb.5010 Layer 3 interface for VLAN 10)
                  : 100.0.20.1 (virtual-gateway-address for irb.5020 Layer 3 interface for VLAN 20)
```

All E-W (Inter DC – Intra VRF: Intra VNI and Inter VNI) traffic flows are tunneled through the L3VPN core across the two data centers as shown in Figure 4.15. No E-W traffic transits the remote PE-3.

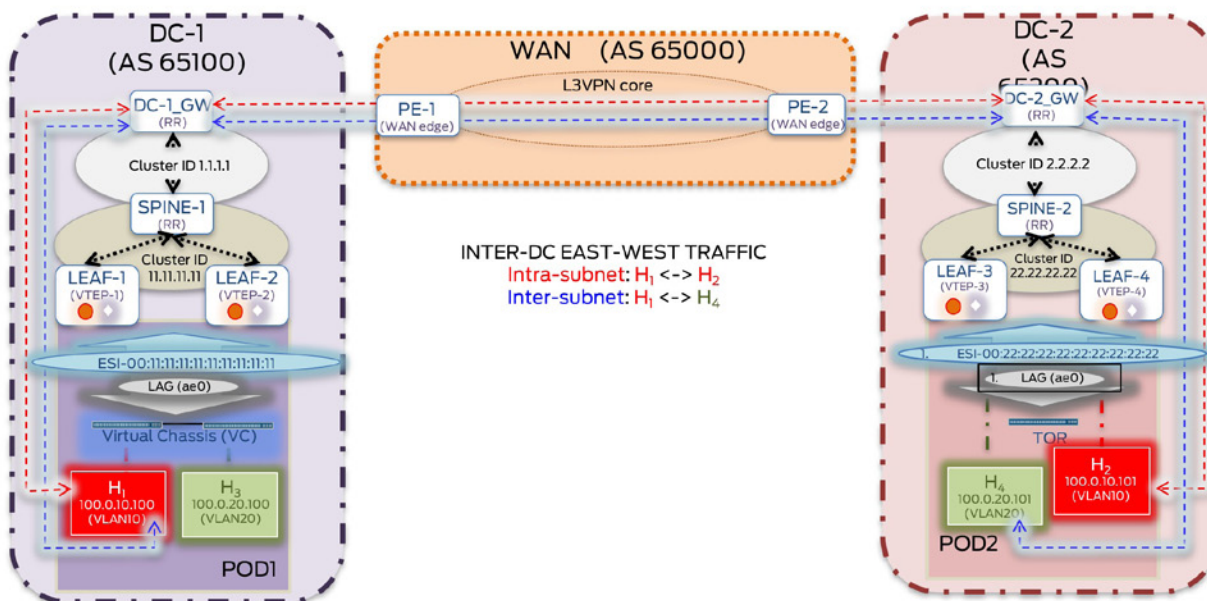


Figure 4.15 Inter-DC EW – Intra-VRF Intra/inter Subnet Traffic Flows

IXIA Statistics – Inter-DC EW: Intra-VRF Intra-Subnet(VNI 5010 <-> VNI 5010)

(H1 <-> H2 1000 flows @ 1000 pps)

IXIA Statistics – Inter-DC EW: Intra-VRF Intra-Subnet(VNI 5010 <-> VNI 5010)

(H1 <-> H2 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC3: Inter-DC EW: Intra-VRF - Intra-subnet - H1 <-> H2	35,744	35,744	0	0.000	2,000.000	2,000.000

IXIA Statistics – Inter-DC EW: Intra-VRF Inter-Subnet(VNI 5010 <-> VNI 5020)

(H1 <-> H4 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC3: Inter-DC EW: Intra-VRF - Inter-subnet - H1 <-> H4	43,750	43,750	0	0.000	2,000.000	2,000.000

North-South Traffic Optimization for Hosts in Different Tenants (Inter-VRF: Inter-subnet)

When Layer 2 is stretched across PODs in different data centers, tromboning might occur for N-S traffic. Egress traffic tromboning (South --> North) from the DC to the WAN, is avoided by means of the virtualized distributed default gateway. Tromboning of ingress traffic (North --> South) from the WAN can be seen as a result of sub-optimal routing, when specific routes indicating the current location of a host are not exchanged.

As seen in Figures 4.16 and 4.17, when DC gateways only advertise summary routes, a remote host across the WAN cannot identify the exact location of the destination host. As a result, traffic intended for a host H1 in DC-1 may trombone through DC-2, following a sub-optimal forwarding path. To alleviate this issue, specific host routes are exchanged to convey location awareness for DC hosts resulting in the creation of an optimal forwarding path.

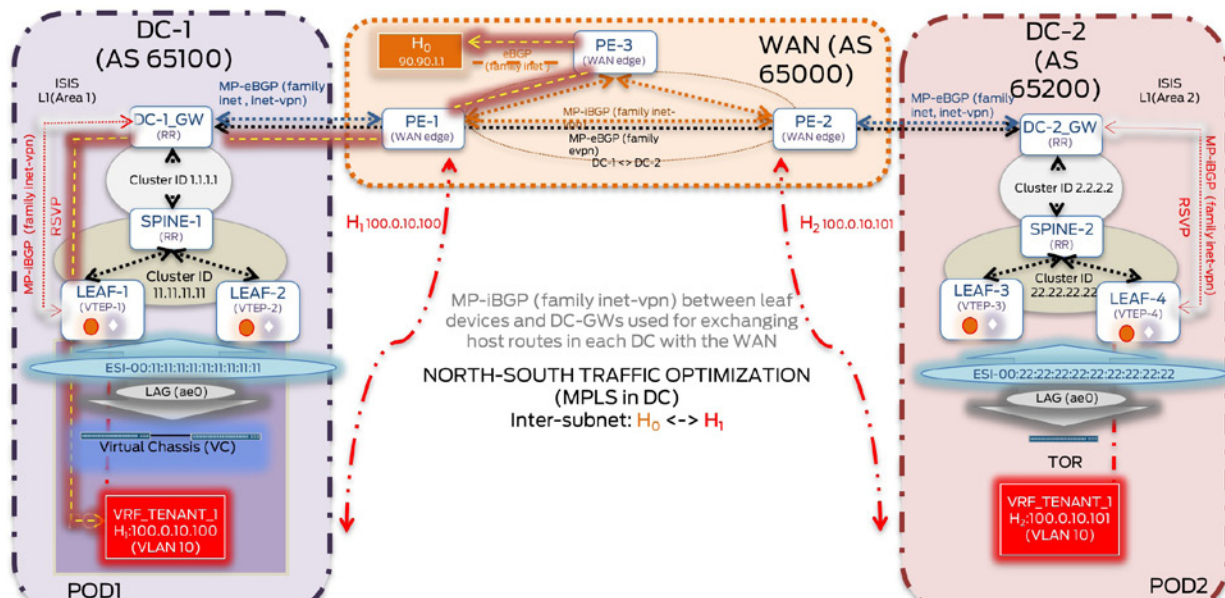


Figure 4.16 North – South Traffic Optimization

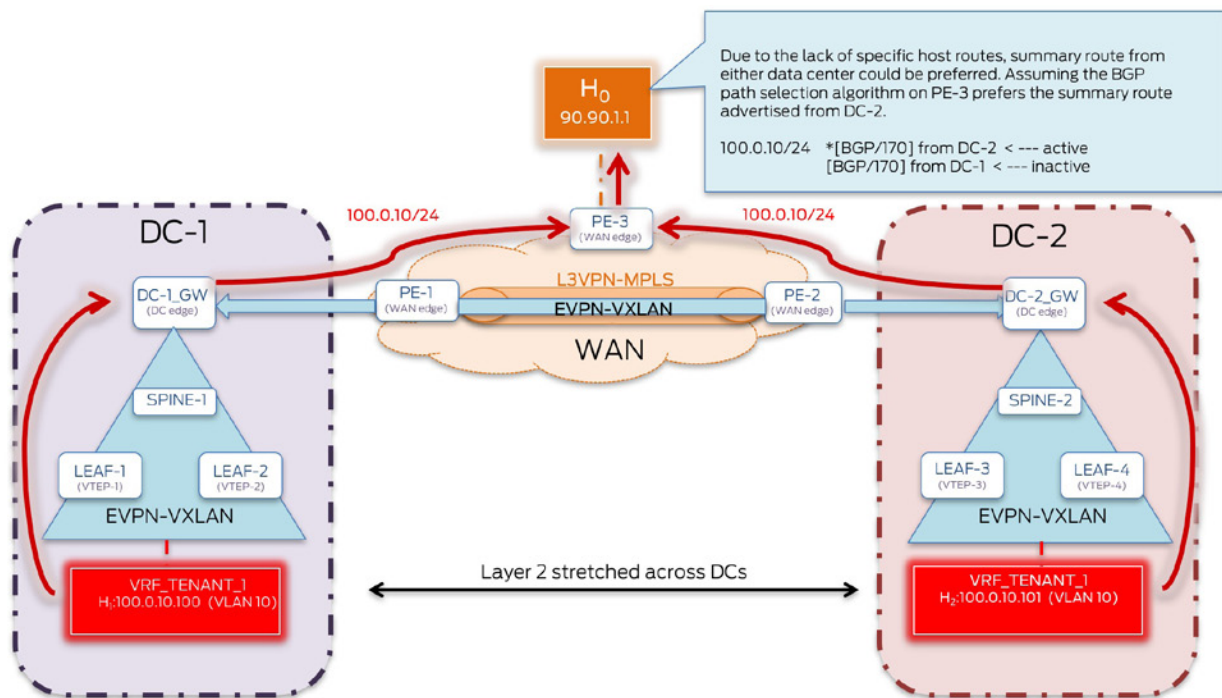


Figure 4.17 Deployment Use Case 3: Ingress Traffic Trombone Effect for N-S Traffic

Configuration

All configuration details to enable Inter-DC EW communication discussed in the previous sections are applied and remain unchanged. Configuration highlights have been outlined to demonstrate N-S traffic optimization using this design alternative that does not use MPLS inside the data center.

Leaf Devices

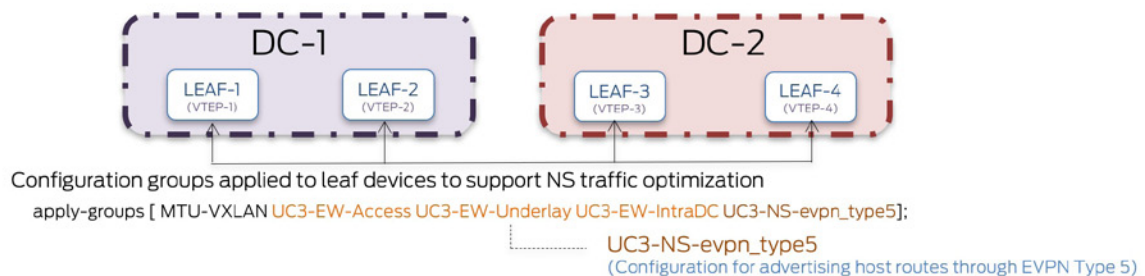


Figure 4.18 N-S Traffic Optimization (No MPLS in DC) Configuration – Leaf Layer

UC3-NS-evpn_type5:

```

jnpr@LEAF-2> show configuration groups UC3-NS-evpn_type5 routing-instances {
  VRF_TENANT_1 { <<< Globally unique VNI 9090 is used to identify VRF_TENANT_1
Export policy to advertise host routes using EVPN Type 5 NLRI
  vrf-target target:10:10;
  protocols {

```



```

    evpn {
        ip-prefix-routes {
            advertise direct-nexthop;
            encapsulation vxlan;
            vni 9090;
            export EVPN-host-routes;
        }
    }
...}
policy-options {
    policy-statement EVPN-host-routes {
        term 1 {
            from {
                protocol evpn;
                route-filter 0.0.0.0/0 prefix-length-range /32-/32;
            }
            then {
                accept;
            }
        }
    }
}
}

```

Spine Devices

Spine devices only act as transport. No additional configuration is needed on spine devices to enable the exchange of EVPN Type 5 routes between leaf and fabric devices.

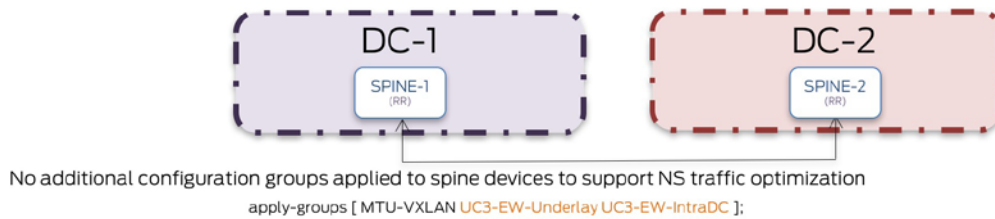


Figure 4.19 N-S Traffic Optimization (No MPLS in DC) Configuration – Spine Layer

Fabric Devices

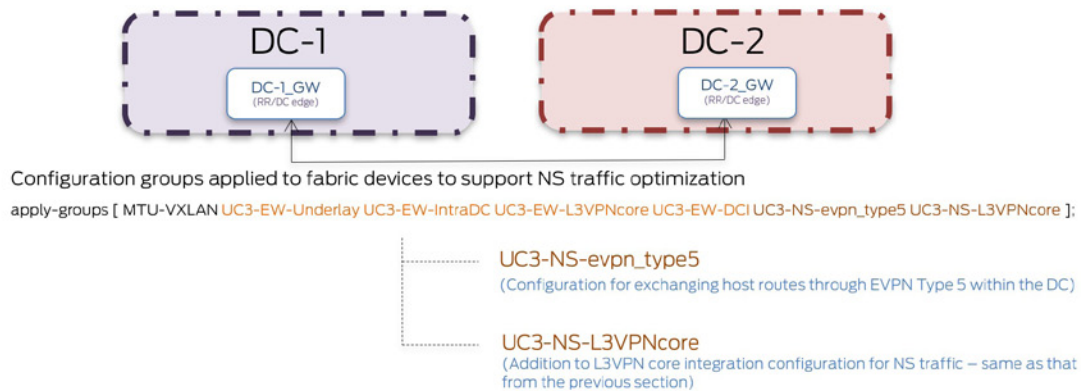


Figure 4.20 N-S traffic Optimization (No MPLS in DC) Configuration – Fabric Layer

```

jnpr@SPINE> show configuration groups UC3-NS-evpn_type5 routing-instances {
    VRF_TENANT_1 { <<< Globally unique VNI 9090 is used to identify VRF_TENANT_1
        instance-type vrf;
        interface lo0.10;
        route-distinguisher 10.1.5.10:10;
    }
}

```

```

vrf-target target:10:10;
vrf-table-label;
protocols {
    evpn {
        ip-prefix-routes {
            advertise direct-nexthop;
            encapsulation vxlan;
            vni 9090;
        }
    }
}
}
}
}
}
}

```

WAN Devices

Configuration on the WAN devices remains unchanged from previous sections.

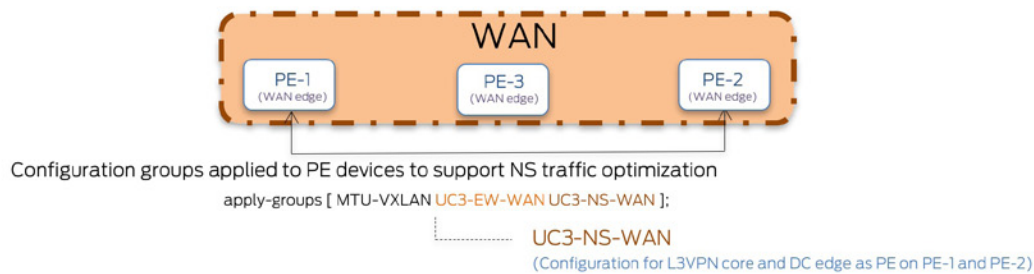


Figure 4.21 N-S Traffic Optimization (No MPLS in DC) WAN Configuration

Verification

Leaf-2 learns host route for H1 in DC-1 and the same is installed in the RIB. This host route is further propagated via `bgp.l3vpn.0` to PE-3. Similarly, H0 route from PE-3 is received by Leaf-2. Though VLAN 10 (VNI 5010) is stretched across DC-1 and DC-2, PE-3 receives specific host routes for each host and is thus aware of the intended host's (H1) location in DC-1.

```

jnpr@PE-3> show route 100.0.10.100 detail
VRF_L3VPN_TENANT-1.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
100.0.10.100/32 (2 entries, 1 announced)
  *BGP    Preference: 170/-101 <<< H1 reachable via DC-1 (PE-1 NH on PE-3)
Route Distinguisher: 10.1.7.10:10
  Next hop type: Indirect
  Address: 0x2a851c8
  Next-hop reference count: 8
  Source: 10.1.1.9
  Next hop type: Router, Next hop index: 605
  Next hop: 50.50.50.4 via xe-0/3/0.0, selected
  Label operation: Push 301504, Push 299984(top)
  Label TTL action: prop-ttl, prop-ttl(top)
  Load balance label: Label 301504: None; Label 299984: None;
  Session Id: 0x182
  Protocol next hop: 10.1.1.9 <<< 10.1.1.9 is the lo0 address of
PE-1 which leads to DC-1
<...>
jnpr@PE-3> show route 100.0.10.101 detail
VRF_L3VPN_TENANT-1.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
100.0.10.101/32 (2 entries, 1 announced)
  *BGP    Preference: 170/-101 <<< H2 reachable via DC-2 (PE-2 NH on PE-3)
Route Distinguisher: 10.1.8.10:10
  Next hop type: Indirect
  Address: 0x2a853c0
  Next-hop reference count: 8
  Source: 10.1.1.10
  Next hop type: Router, Next hop index: 607

```


Next hop: 50.50.50.4 via xe-0/3/0.0, selected
 Label operation: Push 302080, Push 299936(top)
 Label TTL action: prop-ttl, prop-ttl(top)
 Load balance label: Label 302080: None; Label 299936: None;
 Session Id: 0x182
 Protocol next hop: 10.1.1.10

<...>

Traffic Flows

Ingress traffic from H0 ---> H1 is sent across the WAN from remote host H0 directly to DC-1 and no tromboning through DC-2 is seen.

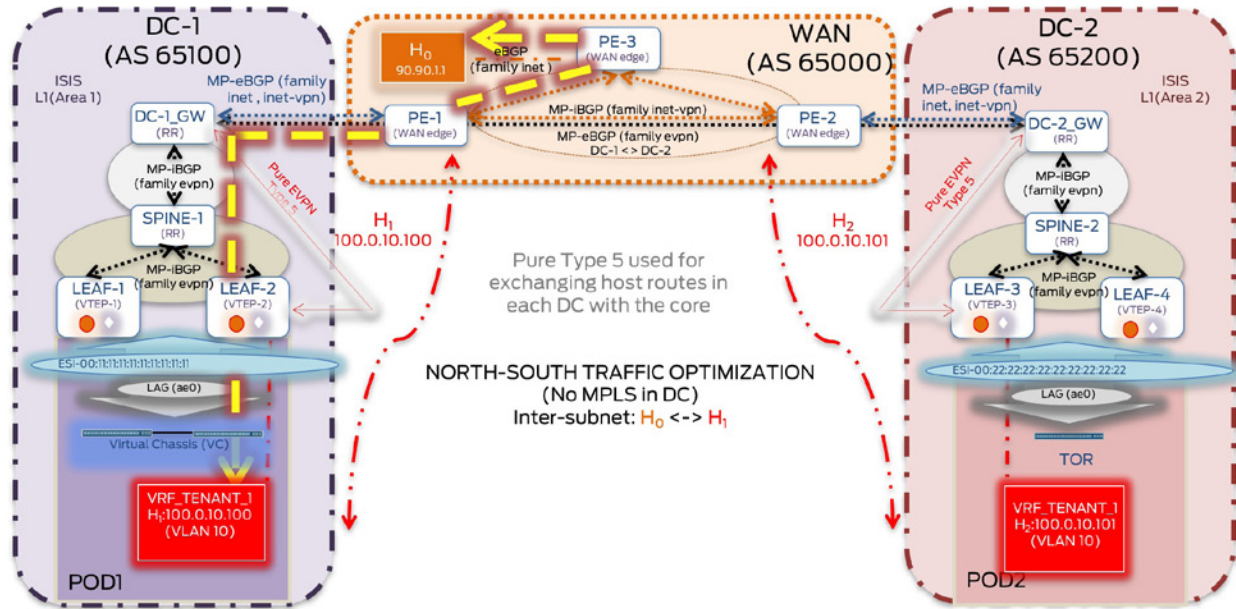


Figure 4.22 N-S Traffic Optimization (No MPLS in DC) – No Ingress Traffic Tromboning

IXIA Statistics – NS Traffic Optimization – No MPLS in DC (H0 <-> H1)

(H0 <-> H1 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC3: NS traffic optimization (MPLS in DC): Remote 90.90.1.1 <-> H1 100.0.10.100	1,011,744	1,011,744	0	0.000	2,000.000	2,000.000

No traffic hair pinning at DC-2_GW:

PE-3			Seconds: 14		Time: 01:28:08
Interface	Link	Input packets	(pps)	Output packets	(pps)
xe-0/2/0	Up	1776642557	(1000)	1563821500	(1000)
xe-0/3/0	Up	1573677460	(1006)	1725526947	(1006)
DC-1_GW			Seconds: 79		Time: 01:29:19
Interface	Link	Input packets	(pps)	Output packets	(pps)
et-0/0/0	Up	178929749	(1002)	155359265	(1002)
xe-0/0/34:0	Up	158190327	(1003)	149100098	(1003)
DC-2_GW			Seconds: 2		Time: 01:30:02
Interface	Link	Input packets	(pps)	Output packets	(pps)
et-0/0/0	Up	91256446	(2)	91214223	(2)
xe-0/0/34:0	Up	91229572	(2)	91229044	(3)
LEAF-2			Seconds: 2		Time: 01:30:22
Interface	Link	Input packets	(pps)	Output packets	(pps)
et-0/0/0	Up	154217198	(1003)	156899254	(1004)
et-0/0/30	Up	78758453	(500)	75092916	(489)
et-0/0/31	Up	78330760	(501)	77894908	(512)
ae0	Up	157089213	(1001)	152987824	(1001)

Chapter 5

Data Center Interconnect – OTT DCI (Public Internet)

Contents

<i>OTT DCI Case Study</i>	<i>184</i>
<i>Topology Description.....</i>	<i>185</i>
<i>Design Summary</i>	<i>186</i>
<i>East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet):</i>	<i>189</i>
<i>Manual Service Chaining of E-W Inter-VRF Traffic Through DC Firewall</i>	<i>204</i>



In this design approach, similar to Chapter 2, routing is done at the spine layer within each data center thus making it possible to reduce functionality and scale requirements at the leaf layer. Spine devices act as Layer 3 gateways as well as overlay route reflectors. Chapter 2 demonstrated the use of fabric devices that act as the DC edge connecting the data center to the WAN. In this case study, the DC edge devices terminate GREoIPsec tunnels across the Public Internet. EVPN-VXLAN tunnels extend end-to-end from each data center over these GREoIPsec tunnels. Due to the higher scale and functional requirements, MX devices (fabric/super-spine) are acting as the DC edge.

As described in Chapter 2, Juniper Networks implementation provides the flexibility to place the Layer 3 gateway function on either the leaf, spine, or fabric to meet the specific requirements being addressed. Chapter 2 demonstrated the use of border leaf, and fabric devices, respectively, acting as service nodes for each POD to direct both E-W and N-S traffic for manual service chaining by the DC firewall. This case study demonstrates the use of spine nodes and service nodes to facilitate manual service chaining of E-W inter-DC traffic, with the local DC firewall being the preferred service block.

For brevity, manual service chaining for N-S traffic has not been demonstrated but the same building blocks as those described in Chapter 2 can be extended to use the spine or fabric as service nodes for N-S traffic as well. A typical deployment example could use QFX 5100/5200 as leaf devices, QFX 10002/8/16 as spine devices, and the MX Series as DC edge devices.

OTT DCI Case Study

This case study focuses on East-West (E-W) traffic flows across tenant hosts in different data centers (Inter-DC) separated by the public Internet. EVPN-VXLAN is set up between data centers over the top (OTT) of an existing WAN network to demonstrate both intra-subnet (Layer 2 domain stretched across data centers) and inter-subnet communication between tenant hosts. Additionally, this section addresses manually service chaining E-W traffic between tenant VRFs across DCI through SRX Series devices for policy enforcement.

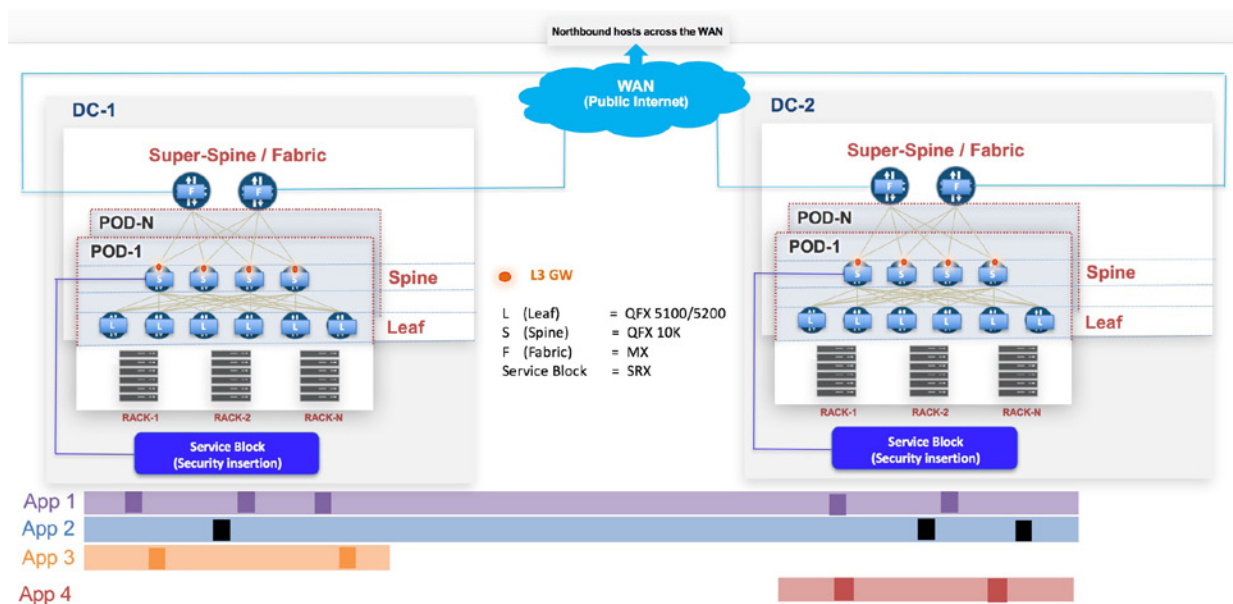


Figure 5.1 Chapter 5's OTT DCI Case Study

Topology Description

Leaf devices: The QFX5100-24Q-2P are used as leaf devices. Each POD consists of two leaf devices.

Spine devices: Three QFX10002-72Qs have been used as the spine devices in each POD in every DC. One spine device per POD has been used as the service node to direct traffic from the IP fabric towards the service block.

Fabric devices: One MX-104 has been used as fabric/super-spine device to connect all PODs in each DC.

Service block: Due to resource constraints, the service block consists of only a single firewall (SRX) in each data center. The designated spine device in every POD can be used as service nodes to connect to the DC firewall.

CE devices: Access switches (QFX5100-48S-6Q – acting as VCs in POD-1 and standalone switches in POD-2) and IXIA simulate CE devices in both PODs in both DCs. Servers can be directly plugged into the TOR leaf devices. To demonstrate the use of LACP, intermediate switches have been used here due to resource constraints. The term host (simulated on IXIA) implies any application entity (VM or container) that consumes an IP/MAC address.

WAN: One MX-104 router has been used to simulate an Internet router.

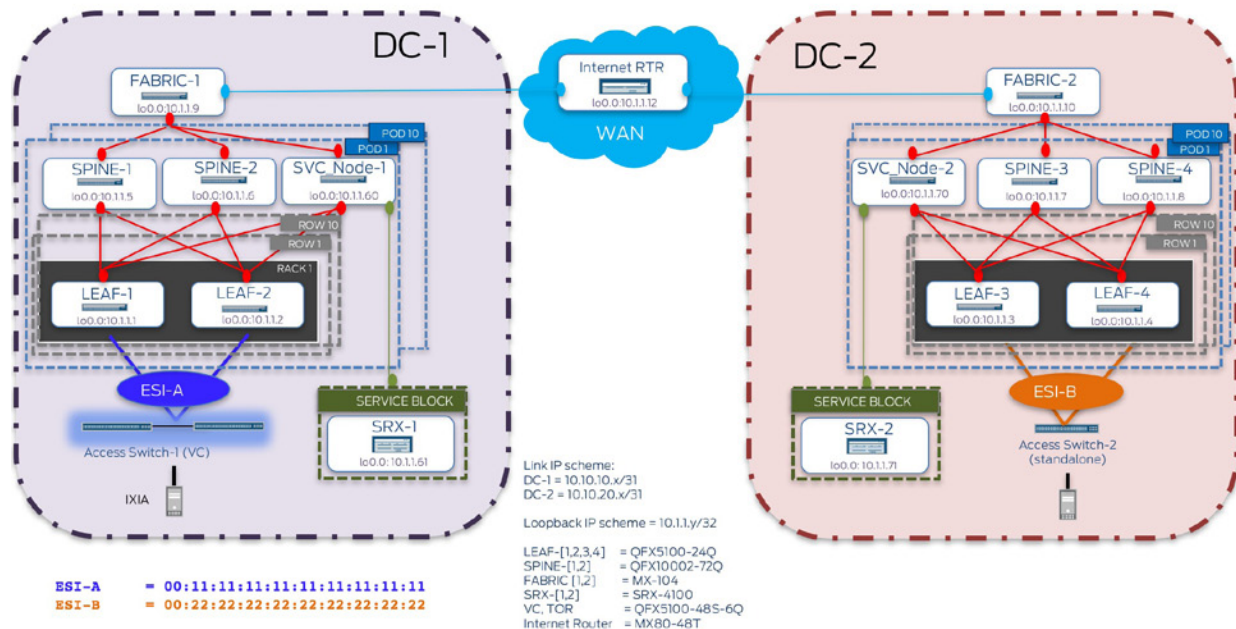


Figure 5.2 Physical Topology

Design Summary

Traffic flows demonstrated:

- East-West (traffic across PODs within the DC):
 - Layer 2 (Intra-VRF Intra-subnet): Traffic between hosts in the same subnet for a given tenant.
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Traffic between hosts in different subnets for different tenants and is manually service-chained through the SRX (service block).
- North-South (traffic between hosts across the WAN and within the DC):
 - Not demonstrated in this case study, but the same building blocks as those described in previous case studies can be used.

Access considerations:

- Two leaf devices are acting as a pair of redundant Top of Rack switches in each DC. CE-1 represents groups of hosts (e.g. VMs/containers residing on servers) mapped to ESI-A (00:11:11:11:11:11:11:11:11:11) on ae0 for LEAF-1 and LEAF-2 in DC-1. Similarly, hosts on CE-2 are mapped to ESI-B (00:22:22:22:22:22:22:22:22:22) on ae0 for LEAF-3 and LEAF-4 in DC-2.

Underlay considerations:

- EBGp has been used to achieve underlay IP reachability (lo0 address reachability between VTEPs). EBGp with unique AS number per device is used to achieve underlay IP reachability.
- MTU has been set on all physical interfaces for devices in a data center to account for VXLAN encapsulation.

Overlay considerations:

- L3 gateway placement:
 - VXLAN routing in this use case is done on the spine devices. Each spine device acts as a Layer 3 gateway for VNIs residing in that POD. Fabric devices in this example do not act as Layer 3 gateways.
 - Routing is done only at the spine layer (including service nodes), while leaf devices acting only as Layer 2 gateways. Fabric devices act only as transit.
- EVPN NLRI exchange:
 - MP-iBGP sessions have been used to exchange EVPN NLRI within the data center.
 - To support EW Inter-DC communication, the EVPN-VXLAN domain within each data center is extended by means of a full mesh EVPN MP-iBGP session between DC core (spine) devices to exchange routes learned in each data center. Fabric devices, interconnecting the different data centers, provide both DC edge and WAN edge functionality and are only used for transport. These fabric devices

are acting as endpoints for the GRE over IPsec tunnels that connect the two data centers over the public network.

■ Route reflection:

■ To avoid full-mesh of control-plane connections (EVPN MP-iBGP sessions within each data center), MP-iBGP EVPN sessions have been created between the leaf devices functioning as clients and spine devices being overlay route-reflectors, responsible for the exchange of EVPN NLRI. In DC-1 (POD-1), LEAF-1 and LEAF-2 act as clients to SPINE-1,2 serving as RRs (cluster ID 11.11.11.11 in DC-1). In DC-2 (POD-1), LEAF-3 and LEAF-4 act as clients to SPINE-3 serving as RRs (cluster ID 22.22.22.22 in DC-2). (Not discussed here, but spine devices with different cluster IDs can provide for added redundancy.)

■ Not repeated here, but hierarchical route reflection with fabric devices acting as overlay route-reflectors can be used. This example shows another variation wherein a full mesh of EVPN control-plane sessions exists between spine devices across DCs. Fabric devices act only as transit in this case study.

■ A full mesh of VXLAN tunnels (data-plane) exist with VTEPs located on each leaf and spine (including service nodes) node in each DC.

Service interface:

■ VLAN aware EVPN service is in use (please refer to the Appendix for more details on EVPN service interfaces).

Host communication:

■ Inter-DC EW traffic flows demonstrate both intra-subnet (Layer 2 extension) and inter-subnet communication between same tenant hosts (Intra-VRF) across data centers, using asymmetric IRB forwarding (using Type 2 routes) for Tenant-1 (VRF_TENANT_1). Use of Type 5 routes for inter-subnet communication has also been demonstrated for Tenant-8 and Tenant-9 (VRF_TENANT_8 and VRF_TENANT_9) hosts within the same tenant (Intra-VRF) and between different tenants (Inter-VRF):

```
Layer 2: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant - Tenant-1) --- > Type 2
Layer 3: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant - Tenant-1) --- > Type 2
Layer 3: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant - Tenant-8) --- > Type 5 (no L2 stretch)
Layer 3: DC-1 POD-1 <> DC-2 POD-1 (Intra-VRF/same tenant - Tenant-9) --- > Type 5 (no L2 stretch)
Layer 3: DC-1 POD-1 <> DC-2 POD-1 (Inter-VRF/different tenants - Tenant-8 and 9) --- > Type 5
```

Integration with the WAN:

■ EVPN unaware core:

■ WAN does not run EVPN but only assists in distributing loopback addresses required for establishing the control and data plane connections across DCs. To achieve this, each DC edge/fabric device terminates a GRE over IPsec tunnel, running OSPF across the same to exchange loopback addresses of leaf and spine devices for each connected data center. This allows for the creation of the full mesh of EVPN/VXLAN tunnels across DCs.

Manual service-chaining orchestration for EW inter-tenant inter-subnet traffic:

■ Service Nodes:

■ Here, spine devices (one per POD) act as service nodes.

- The term service node is used for devices that direct traffic from the IP-fabric towards the service block.
- Service Block:
 - Layer 3 security insertion has been demonstrated for inter-tenant EW traffic. The service block in this use case consists of a firewall that services all PODs. It is not shown here, but SRX clusters can be used one for each POD or the entire DC, depending on scale and high-availability requirements.
 - SRX is physically connected to each service node device. Hair pinning through the SRX is only treated as an IP path problem, therefore, getting intended traffic to the SRX.
- Manual service-chaining orchestration for E-W inter-tenant inter-subnet traffic:
 - For Inter-VRF communication, manual service chaining has been demonstrated by hair pinning traffic across service block (SRX devices) in the originating data center. SRX devices are connected to a spine device acting as service node that essentially hosts Type 5 routes for tenant hosts that need to communicate with each other and directs traffic to the service block.

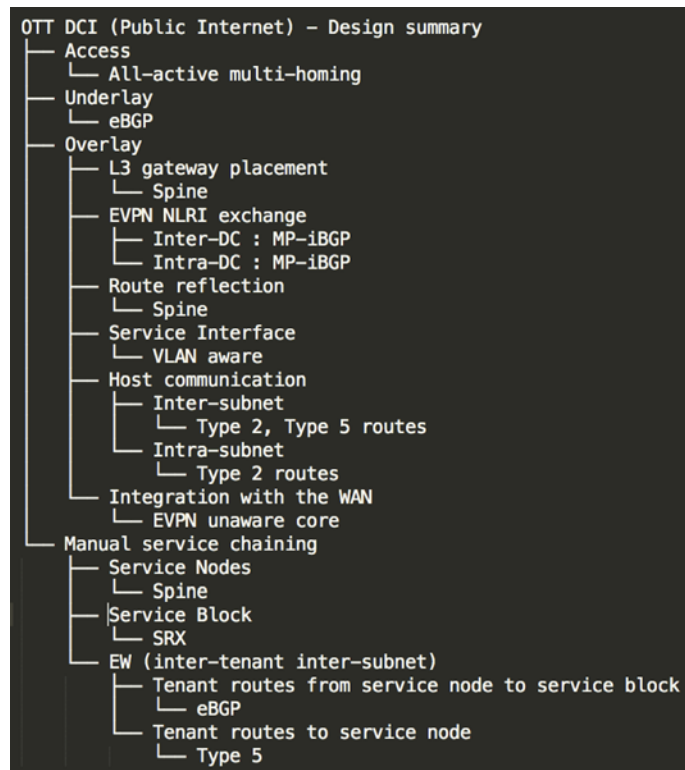


Figure 5.3 Design Summary for OTT DCI – Public Internet

Traffic Scenarios

The following sections will demonstrate implementation.

1. East-West traffic for hosts in the same tenant (Intra-VRF: Intra-subnet and Inter-subnet):

- Configuration
- Verification
- Traffic flows

2. Manual service chaining of East-West traffic across DCs for hosts in different tenants (Layer 3 security insertion: Inter-tenant Inter-subnet):

- Configuration layout
- Route exchange between service nodes and service blocks
- Configuration
- Verification
- Traffic flows

East-West Traffic (Intra-VRF: Intra-subnet and Inter-subnet)

Figures 5.4 and 5.5 highlight the logical topology for E-W traffic across PODs as summarized by the design highlights. Details on the configuration and verification steps will be provided.

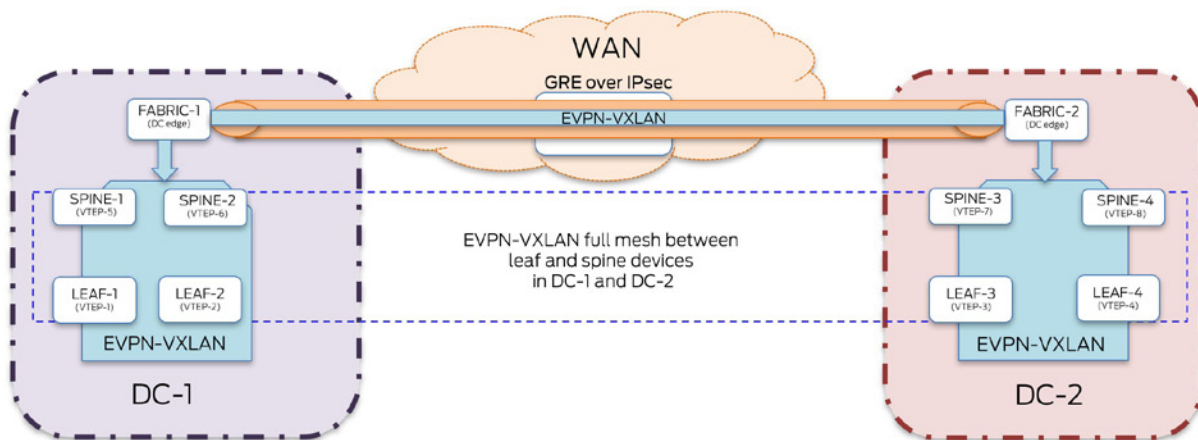


Figure 5.4 DCI Logical Topology High Level Overview

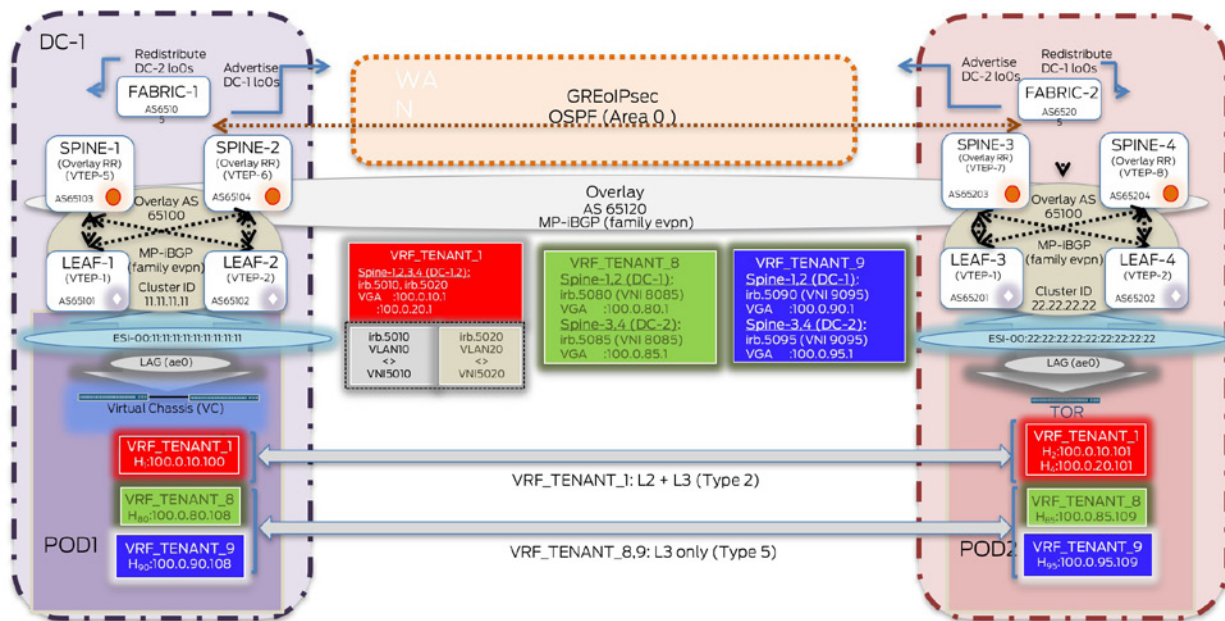


Figure 5.5 DCI Logical Topology Detailed Overview

Configuration

Configuration is divided across multiple groups, which are applied to the different devices as needed. Configuration highlights have been outlined in cases where they differ from those as described in the previous use cases. Snippets will show configuration highlights for the leaf (LEAF-1), spine (SPINE-1) and fabric (FABRIC-1) layer devices across DC-1 and DC-2.

Leaf Devices

Configuration on the leaf devices is divided into three components – underlay, overlay (intra-DC), and access. Three configuration groups have been applied to all leaf devices in both DC-1 and DC-2 – UC4-EW-Access, UC4-EW-Underlay, UC4-EW-IntraDC as shown in Figure 5.6.

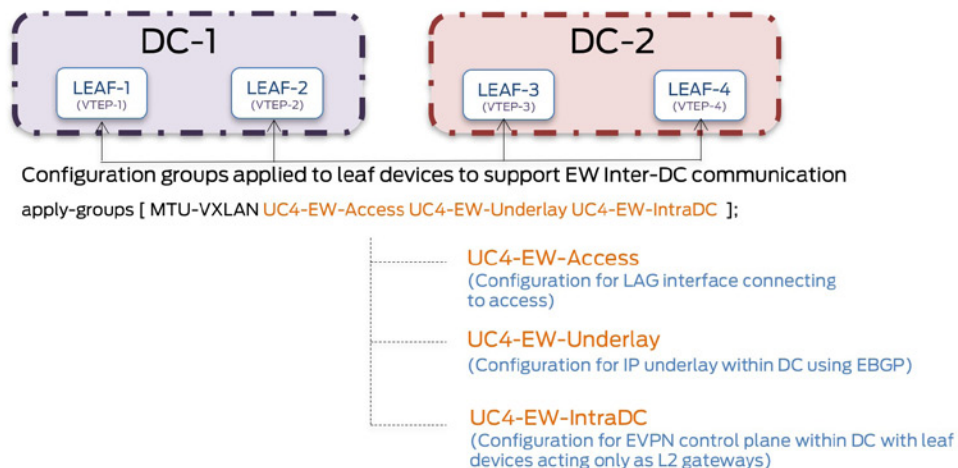


Figure 5.6 DCI Configuration – Leaf layer

jnpr@LEAF-1> show configuration groups UC4-EW-Access

```

interfaces {
  et-0/0/20 {...}
  et-0/0/21 {...} <<< Configuration components for access remain unchanged from the previous use cases,
shown here is the output on LEAF-1

```

```

ae0 {
  description "LAG VC <> LEAF-1";
  mtu 9192;
  esi {
    00:11:11:11:11:11:11:11:11;
    all-active;
  }
  aggregated-ether-options {
    lacp {
      active;
      periodic fast;
      system-id 01:01:01:01:01:01;
    }
  }
  unit 0 {
    family ethernet-switching {
      interface-mode trunk;
      vlan {
        members all;
      }
    }
  }
}

```

jnpr@LEAF-1> show configuration groups UC4-EW-Underlay

```

interfaces {
  lo0 {...}
  et-0/0/22 {...}
  et-0/0/23 {...}
  {...}
}
routing-options {
  {...}
}

protocols {
  bgp {
    group EBGp-Underlay {
      type external;
      mtu-discovery;
      import underlay-in;
      export underlay-out;
      local-as 65101;
      bfd-liveness-detection {...}
      multipath multiple-as;
      neighbor 10.10.10.8 { <<< Each leaf device exchanges loopback addresses over the EBGp
session with each connecting spine (including service node).
local-address 10.10.10.9;
peer-as 65103;
      }
      neighbor 10.10.10.10 {
        local-address 10.10.10.11;
        peer-as 65104;
      }
      neighbor {...}
    }
  }
}

policy-options {
  policy-statement LB {...}
  policy-statement underlay-in {
    term acpt-remote-lo0 {
      from {
        route-filter 10.1.1.0/24 orlonger;
      }
      then accept;
    }
  }
}

```

```

}
}
policy-statement underlay-out {
  term adv-local-lo0 {
    from {
      protocol direct;
route-filter 10.1.1.0/24 orlonger;          }
    then {
      next-hop self;
      accept;
    }
  }
  term default {
    then reject;
  }
}
}

jnpr@LEAF-1> show configuration groups UC4-EW-IntraDC
protocols {
  bgp {
    group IBGP-EVPN-DC1 {
      type internal;
      description «Leaf clients for Spine RR»;
import overlay-in; <<< Import policy configured on leaf devices for EW traffic optimization across
DCs. As a result, leaf devices will prefer routes from local DC Layer 3 gateways, else tromboning might
happen since same virtual-gateway (VRF_TENANT_1) address is configured on spine devices across DCs.
      local-address 10.1.1.1;
      family evpn {
        signaling;
      }
      local-as 65120;
      bfd-liveness-detection {...}
      multipath;
      neighbor 10.1.1.5 {
        peer-as 65120;
      }
      neighbor 10.1.1.6 {
        peer-as 65120;
      }
      neighbor { ... }
    }
  }
  evpn {
    encapsulation vxlan;
    extended-vni-list all;
  }
}

policy-options {
  policy-statement EVPN-IMPORT {
term ESI_IN {
    from community comm-esi-in;
    then accept;
  }
  term default {
    then reject;
  }
}
  policy-statement overlay-in {
term reject-remote-gw {
from {
    family evpn;
    next-hop [ 10.1.1.7 10.1.1.8 10.1.1.70];
    nlri-route-type [ 1 2 ];
    then reject;
  }
  term default {
    then accept;
  }
}
}
community comm-esi-in members target:1:100;
}

```

```

switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.1.1.1:100;
    vrf-import EVPN-IMPORT;    <<< Manual route targets have been used
    vrf-target {
        target:1:100;
    ...}
}

vllans {
    bd5010 {
        vllan-id 10;
        vxlan {
            vni 5010;
        }
    }
    bd5020 {
        vllan-id 20;
        vxlan {
            vni 5020;
        }
    }
    bd5080 {
        vllan-id 80;
        vxlan {
            vni 5080;
        }
    }
    bd5090 {
        vllan-id 90;
        vxlan {
            vni 5090;
        }
    }
}

```

Spine Devices

Configuration on the spine devices is divided into four components as shown in Figure 5.7 – underlay, overlay (intra-DC), overlay (Inter-DC), and Layer 3 DCI using Pure Type 5 routes. Four configuration groups have been applied to all spine devices in both DC-1 and DC-2: UC4-EW-Underlay, UC4-EW-IntraDC, UC4-EW-DCI, and UC4-EW-T5_L3DCI.

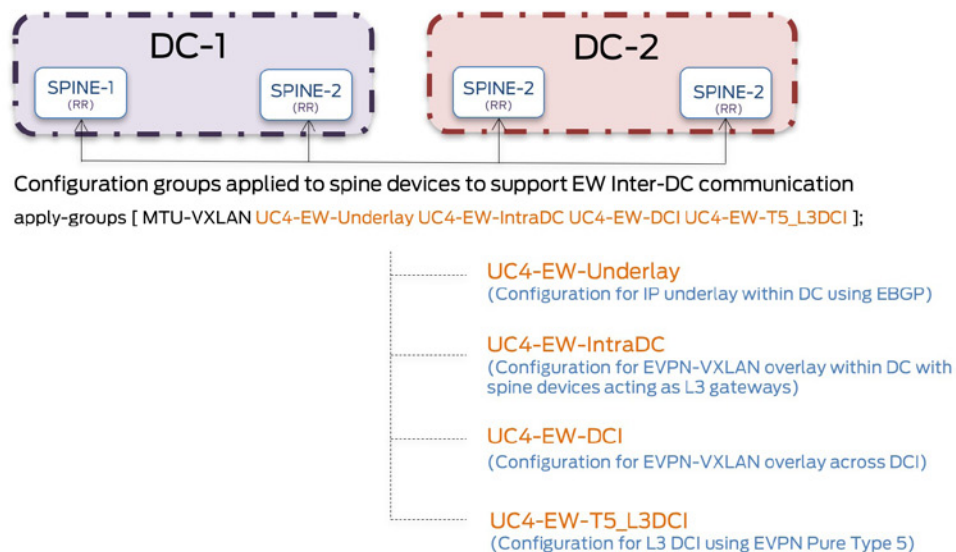


Figure 5.7 DCI Configuration – Spine Layer

```

jnpr@SPINE-1> show configuration groups UC4-EW-Underlay
interfaces {
    lo0      {...}
    et-0/0/12 {...}
    et-0/0/13 {...}
    xe-0/0/24:0 {...}
    ...}
routing-options {...}
}

protocols {
    bgp {
        group EBG-Underlay {
            type external;
            mtu-discovery;
            import underlay-in; <<< Each spine device exchanges loopback addresses over the EBG session
with each connecting leaf and fabric device.
            export underlay-out;
            local-as 65103;
            bfd-liveness-detection {...}
            multipath multiple-as;
            neighbor 10.10.10.9 {
local-address 10.10.10.8;
peer-as 65101;
            }
            neighbor 10.10.10.13 {
local-address 10.10.10.12;
peer-as 65102;
            }
            neighbor 10.10.10.0 {
local-address 10.10.10.1;
peer-as 65105;
            }
        }
    }
}

policy-options {
    policy-statement LB {...}
    policy-statement underlay-in {
        term acpt-remote-lo0 {
            from {
route-filter 10.1.1.0/24 orlonger; }
            then accept;
        }
    }
    policy-statement underlay-out {
<<< All remote VTEPs appear to be directly connected. In cases when there is a link failure between a leaf
and connected spine, if next-hop reachability to the selected spine is available even through a non-
optimal path, no path recomputation to use an alternate spine is done. For example, if the link between
LEAF-1 and SPINE-1 fails, it is possible that LEAF-1 still continues to use SPINE-1, as reachability is
available through SPINE-2 which causes traffic to traverse the path LEAF-1 > SPINE-2 > LEAF-2 > SPINE-1.
To avoid this and to allow for EW traffic optimization within a DC, routes are rejected on spine nodes with
as-path > 2/3 (3/5 stage clos). This prevents sub-optimal routing to reach remote VTEP via another leaf.

        term adv-local-lo0 {
            from {
                protocol direct;
                route-filter 10.1.1.0/24 orlonger; }
            then {
                community add comm_spine-lo0;
                next-hop self;
                accept;
            }
        }
    }
}
/* To prevent peer spine loopback learned from leaf from being re-advertised */
term rej-as-path {
    from {
        as-path asPathLength2;

```

```

    community comm_spine-lo0; }
    then reject;
  }
}
community comm_spine-lo0 members target:12345:999;
as-path asPathLength2 «.{2,}»;}
}

```

UC4-EW-IntraDC:

```

jnpr@SPINE-1> show configuration groups UC4-EW-IntraDC
protocols {
  bgp {
    group IBGP-EVPN-DC1-RR {
      type internal;
      description "Spine Overlay RR for Leaf clients";
      local-address 10.1.1.5;
      family evpn {
        signaling;
      }
      export no-adv-type5;
      vpn-apply-export;
      cluster 11.11.11.11;
      local-as 65120;
      bfd-liveness-detection {...}
      multipath;
      neighbor 10.1.1.1;
      neighbor 10.1.1.2;
    }
    evpn {
      encapsulation vxlan;
      extended-vni-list all;
      default-gateway no-gateway-community;
    }
  }
  policy-options {
    policy-statement no-adv-type5 {
      term no-t5 {
        from {
          family evpn;
          nlri-route-type 5;
        }
        then reject;
      }
      term default {
        then accept;
      }
    }
  }
}
<<< EVPN Type 5 NLRI are not advertised to the leaf (QFX5100) devices.
  policy-statement EVPN-IMPORT {
    term ESI_IN {
      from community comm-esi-in;
      then accept;
    }
    term default {
      then reject;
    }
  }
}
community comm-esi-in members target:1:100;
}

<<< All spine device (including service nodes) act as L3 gateways
interfaces {
  irb {
    unit 5010 {
      proxy-macip-advertisement;
    }
  }
  description "Tenant 1 - vlan 10 - vni5010";
  family inet {
    address 100.0.10.2/24 {
      virtual-gateway-address 100.0.10.1;
    }
  }
}

```



```

...}
  unit 5020 {
    proxy-macip-advertisement;
    description "Tenant 1 - vlan 20 - vni5020";
    family inet {
      address 100.0.20.2/24 {
virtual-gateway-address 100.0.20.1;
...}

```

<<< Intra-subnet (layer 2 stretch) and inter-subnet communication for Tenant 1 is achieved using asymmetric forwarding with Type 2 routes.

```

routing-instances {
  VRF_TENANT_1 {
    instance-type vrf;
    interface irb.5010;
    interface irb.5020;
    interface lo0.10;
    route-distinguisher 10.1.5.10:10;
    vrf-target target:10:10;
    vrf-table-label;
    ...}
  switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.1.1.5:100;
    vrf-import EVPN-IMPORT;
    vrf-target {
      target:1:100;
    }
  }
  ...}

vllans {
  bd5010 {
    vlan-id 10;
    vxlan {
      vni 5010;
    }
  }
  bd5020 {
    vlan-id 20;
    vxlan {
      vni 5020;
    }
  }
  ...}

```

UC4-EW-T5_L3DCI:

```

jnpr@SPINE-1> show configuration groups UC4-EW-T5_L3DCI
interfaces {
  irb {
    unit 5080 {
      proxy-macip-advertisement;
      description «Tenant 8 - vlan 80 - vni5080»;
      family inet {
        address 100.0.80.2/24 {
virtual-gateway-address 100.0.80.1;
...}
    unit 5090 {
      proxy-macip-advertisement;
      description «Tenant 9 - vlan 90 - vni5090»;
      family inet {
        address 100.0.90.2/24 {
virtual-gateway-address 100.0.90.1;
...}
  lo0 {
    unit 80 {
      family inet {

```

```

        address 10.1.5.80/32;
...}
    unit 90 {
        family inet {
            address 10.1.5.90/32;
        }
    }
...}

vllans {
    bd5080 {
        vllan-id 80;
        l3-interface irb.5080;
        vxlan {
            vni 5080;
        }
    }
    bd5090 {
        vllan-id 90;
        l3-interface irb.5090;
        vxlan {
            vni 5090;
        }
    }
...}

routing-instances {
    VRF_TENANT_8 {
        instance-type vrf;
        interface irb.5080;
        interface lo0.80;
        route-distinguisher 10.1.5.80:85;
        vrf-export vrf-export-pol_VRF-8;
        vrf-target target:80:85;
        vrf-table-label;
        protocols {
            evpn {
                ip-prefix-routes {
                    advertise direct-nexthop;
                    encapsulation vxlan;
                    vni 8085;
                }
            }
        }
    }
...}

VRF_TENANT_9 {
    instance-type vrf;
    interface irb.5090;
    interface lo0.90;
    route-distinguisher 10.1.5.90:95;
    vrf-export vrf-export-pol_VRF-9;
    vrf-target target:90:95;
    vrf-table-label;
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop;
                encapsulation vxlan;
                vni 9095;
            }
        }
    }
...}

```

Tenants 8 and 9 (VRF_TENANT_8,9) have been set up for Layer 3 only and use Pure Type 5 for the same.

Fabric Devices

Configuration on the fabric devices is divided into two components as shown in Figure 5.8: underlay (intra-DC) and transport for DCI. Two configuration groups have been applied to all fabric devices in both DC-1 and DC-2: UC4-EW-Underlay and UC4-EW-DCI.

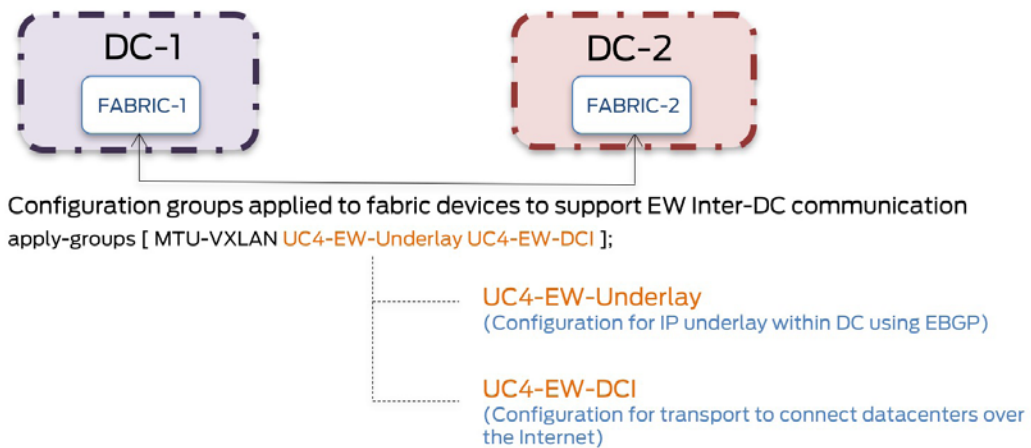


Figure 5.8 DCI Configuration – Fabric Layer

```
jnpr@FABRIC-1> show configuration groups UC4-EW-Underlay
interfaces {
    lo0      {...}
    xe-2/0/0 {...}
    xe-2/0/1 {...}
    ...}
routing-options {
    {...}
}

protocols {
    bgp {
        group EBGp-Underlay {
            type external;
            mtu-discovery;
            import underlay-in; <<< Each fabric device exchanges loopback addresses (within the data
center) over EBGp session with each connecting spine device.
```

```
        export underlay-out;
        local-as 65105;
        bfd-liveness-detection {...}
        multipath multiple-as;
        neighbor 10.10.10.1 {
local-address 10.10.10.0;
peer-as 65103;
        }
        neighbor 10.10.10.3 {
local-address 10.10.10.2;
peer-as 65104;
        }
        neighbor {...}
    ...}
}
```

```
policy-options {
    policy-statement LB {...}
    policy-statement underlay-in {
        term acpt-remote-lo0 {
            from {
route-filter 10.1.1.0/24 orlonger;
            }
        then accept;
    }
}
policy-statement underlay-out {
    term adv-local-lo0 {
```

```

        from {
            protocol direct;
        }
        route-filter 10.1.1.0/24 orlonger;
    }
    then {
        next-hop self;
        accept;
    }
}
}

```

UC4-EW-DCI:

```

jnpr@FABRIC-1> show configuration groups UC4-EW-DCI
chassis {
    fpc 0 {
        pic 3 {
            tunnel-services;
        }
    }
    ...}
services {
    service-set DC-2_VPN_SET {
        next-hop-service {
            inside-service-interface ms-1/2/0.1;
            outside-service-interface ms-1/2/0.2;
        }
        ipsec-vpn-options {
            local-gateway 30.30.30.0;
            tunnel-mtu 5000;
        }
        ipsec-vpn-rules DC-2_VPN_RULE;
    }
}

ipsec-vpn {
    rule DC-2_VPN_RULE {
        term 1 {
            then {
                remote-gateway 40.40.40.0;
                dynamic {
                    ike-policy IKE-POLICY;
                    ipsec-policy IPSEC-POLICY;
                }
            }
            match-direction input;
        }
        ipsec {
            proposal IPSEC-PROPOSAL {
                protocol esp;
                authentication-algorithm hmac-sha1-96;
                encryption-algorithm aes-256-cbc;
                lifetime-seconds 3600;
            }
            policy IPSEC-POLICY {
                perfect-forward-secrecy {
                    keys group2;
                }
                proposals IPSEC-PROPOSAL;
            }
        }
    }
    ...}

```

<<< IPsec configuration (IKE and IPsec proposals and policies) between fabric devices

```

ike {
    proposal IKE-PROPOSAL {
        authentication-method pre-shared-keys;
        dh-group group2;
        authentication-algorithm sha1;
        encryption-algorithm aes-256-cbc;
        lifetime-seconds 21600;
    }
    policy IKE-POLICY {
        version 2;
    }
}

```

```

        proposals IKE-PROPOSAL;
        pre-shared-key ascii-text "$9$cPLlEW4oGk.5oJGiqfn68X7-s2oJGiqm";
        ## SECRET-DATA
    }
}
establish-tunnels immediately;
}
}

```

<<< GRE tunnel created between fabric devices is enabled to process IPv4 packets

```

interfaces {
    ms-1/2/0 {
        mtu 7000;
        unit 1 {
            description "IPsec interface to DC-2";
        family inet {
            address 192.0.10.9/30;
        }
        service-domain inside;
    }
    unit 2 {
        family inet;
        service-domain outside;
    }
}
lo0 { ... }
xe-0/3/0 { ... }
gr-0/3/0 {
    unit 0 {
        description "GRE interface to DC-2";
    }
}
tunnel {
    source 198.30.10.9;
    destination 198.40.10.10;
}
family inet;
...}
...}

```

<<< Static routes to direct traffic to the remote GRE tunnel endpoint through the IPsec tunnel that traverses the intermediate public network

```

routing-options {
    static {
        route 40.40.40.0/24 next-hop 30.30.30.1;
        route 198.40.10.10/32 next-hop ms-1/2/0.1;
    }
}

policy-options {
    prefix-list local-DC-lo0s {
        10.1.1.1/32;
        10.1.1.2/32;
        10.1.1.5/32;
        10.1.1.6/32;
    }
    prefix-list remote-DC-lo0s {
        10.1.1.3/32;
        10.1.1.4/32;
        10.1.1.7/32;
        10.1.1.8/32;
    }
}

policy-statement adv-remote-lo0s {
    term lo0 {
        from {
            prefix-list remote-DC-lo0s;
        }
        then accept;
    }
}

policy-statement adv-local-lo0s {
    term lo0 {
        from {
            prefix-list local-DC-lo0s;
        }
    }
}

```

```

        then accept;
    }
...}

```

<<< OSPF is enabled on the GRE tunnel connecting the DC edge devices over which DC loopback addresses are exchanged so that MP-iBGP (EVPN) sessions can be created between spine devices across the different data centers.

```

protocols {
    bgp {
        group EBGp-Underlay {
            export adv-remote-lo0s;
        }
    }
    ospf {
        export adv-local-lo0s;
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface gr-0/3/0.0;
        }
    }
...}

```

Verification

IPsec tunnels are operational and traffic is carried over the GRE tunnel between fabric devices.

```

jnpr@FABRIC-1> show services ipsec-vpn ipsec security-associations
Service set: DC-2_VPN_SET, IKE Routing-instance: default
Local gateway: 30.30.30.0, Remote gateway: 40.40.40.0
IPSec inside interface: ms-1/2/0.1, Tunnel MTU: 5000

```

Direction	SPI	AUX-SPI	Mode	Type	Protocol
inbound	3463568178	0	tunnel	dynamic	ESP
outbound	4059263108	0	tunnel	dynamic	ESP

```

jnpr@FABRIC-1> show services ipsec-vpn ipsec statistics
PIC: ms-1/2/0, Service set: DC-2_VPN_SET
ESP Statistics:
  Encrypted bytes:      1517468976
  Decrypted bytes:      1558772160
  Encrypted packets:    1013988
  Decrypted packets:    1034291

```

```

jnpr@FABRIC-1> show interfaces statistics gr-0/3/0
Physical interface: gr-0/3/0, Enabled, Physical link is Up
  Interface index: 140, SNMP ifIndex: 528
  Type: GRE, Link-level type: GRE, MTU: Unlimited, Speed: 100000mbps
  Device flags      : Present Running
  Interface flags: Point-To-Point SNMP-Traps
  Statistics last cleared: Never
  Input rate       : 171007920 bps (14227 pps)
  Output rate      : 172053600 bps (14443 pps)

```

```

Logical interface gr-0/3/0.0 (Index 327) (SNMP ifIndex 540)
  Description: GRE interface to DC-2
  Flags: Up Point-To-Point SNMP-Traps 0x0 IP-
Header 198.40.10.10:198.30.10.9:47:df:64:0000000000000000 Encapsulation: GRE=NULL
  Copy-tos-to-outer-ip-header: Off
  Gre keepalives configured: Off, Gre keepalives adjacency state: down
  Input packets : 2009287
  Output packets: 2004026
  Protocol inet, MTU: 9000
  Flags: Sendbcst-pkt-to-re, User-MTU

```

Protocol next-hop on routes is preserved across DC boundaries. For Tenant 1, Type 2 routes are exchanged to advertise DC-1 (H1) and DC-2 (H2, H4) host routes. For Tenants 8 and 9, Type 5 routes are exchanged to advertise DC-1 (H80, H90) and DC-2 (H85, H95) host routes. Full mesh of VXLAN tunnels exist across VTEPs (leaf and spine devices) in both DC-1 and DC-2.

```
jnpr@LEAF-1> show evpn database mac-address 00:00:1e:63:d1:fd extensive
Instance: default-switch
VN Identifier: 5010, MAC address: 00:00:1e:63:d1:fd
  Source: 00:22:22:22:22:22:22:22:22:22:22:22, Rank: 1, Status: Active
    Remote origin: 10.1.1.3
    Remote origin: 10.1.1.4
<...>
jnpr@LEAF-1> show evpn database mac-address 00:00:00:93:3c:f4 extensive
Instance: default-switch
VN Identifier: 5020, MAC address: 00:00:00:93:3c:f4
  Source: 00:22:22:22:22:22:22:22:22:22:22:22, Rank: 1, Status: Active
    Remote origin: 10.1.1.3
    Remote origin: 10.1.1.4
<...>
jnpr@SPINE-1> show evpn ip-prefix-database l3-context VRF_TENANT_8
L3 context: VRF_TENANT_8
```

IPv4->EVPN Exported Prefixes

Prefix	EVPN route status
100.0.80.0/24	Created

EVPN->IPv4 Imported Prefixes

Prefix	Etag	IP route status
100.0.80.0/24	0	Created
Route distinguisher	St	VNI/Label
10.1.6.80:85	A	8085
Router MAC	54:4b:8c:62:cb:f6	10.1.1.6
100.0.85.0/24	0	Created
Route distinguisher	St	VNI/Label
10.1.7.80:85	A	8085
Router MAC	ec:3e:f7:84:e3:fa	10.1.1.7
10.1.8.80:85	A	8085
Router MAC	54:4b:8c:cd:b4:38	10.1.1.8

```
jnpr@SPINE-1> show evpn ip-prefix-database l3-context VRF_TENANT_9
L3 context: VRF_TENANT_9
```

IPv4->EVPN Exported Prefixes

Prefix	EVPN route status
100.0.90.0/24	Created

EVPN->IPv4 Imported Prefixes

Prefix	Etag	IP route status
100.0.90.0/24	0	Created
Route distinguisher	St	VNI/Label
10.1.6.90:95	A	9095
Router MAC	54:4b:8c:62:cb:f6	10.1.1.6
100.0.95.0/24	0	Created
Route distinguisher	St	VNI/Label
10.1.7.90:95	A	9095
Router MAC	ec:3e:f7:84:e3:fa	10.1.1.7
10.1.8.90:95	A	9095
Router MAC	54:4b:8c:cd:b4:38	10.1.1.8

Traffic Flows

For this case study, both PODs are situated in different data centers (Inter DC) and all hosts belong to the same tenant (Intra VRF). Hosts H_1 and H_2 belong to VLAN 10 mapped to VNI 5010 while hosts H_3 and H_4 belong to VLAN 20 mapped to VNI 5020. These hosts share the same IP-VRF (VRF_TENANT_1). Hosts H_{80} and H_{85} belong to VLAN 80 and VLAN 85 respectively and are mapped to Layer 3 VNI 8085 (Pure Type 5). These share the same IP-VRF (VRF_TENANT_8). Hosts H_{90} and H_{95} belong to VLAN 90 and VLAN 95 respectively and are mapped to Layer 3 VNI 9095 (Pure Type 5). These share the same IP-VRF (VRF_TENANT_9).

CE-1 hosts multi-homed to LEAF-1 and LEAF-2 (ESI-A = 00:11:11:11:11:11:11:11:11:11):
 Host 1 (H₁) : VLAN 10 <-> VNI 5010, IPv4 = 100.0.10.100, MAC = 00:00:1e:63:c8:7c
 Host 3 (H₃) : VLAN 20 <-> VNI 5020, IPv4 = 100.0.20.100, MAC = 00:00:00:93:3c:f3
 Host 80 (H₈₀) : VLAN 80 <-> VNI 8085, IPv4 = 100.0.80.108, MAC = 00:00:0e:a9:d8:9f
 Host 90 (H₉₀) : VLAN 90 <-> VNI 9095, IPv4 = 100.0.90.109, MAC = 00:00:0e:a9:d8:a0

CE-2 hosts multi-homed to LEAF-3 and LEAF-4 (ESI-B = 00:22:22:22:22:22:22:22:22:22):
 Host 2 (H₂) : VLAN 10 <-> VNI 5010, IPv4 = 100.0.10.101, MAC = 00:00:1e:63:d1:fd
 Host 4 (H₄) : VLAN 20 <-> VNI 5020, IPv4 = 100.0.20.101, MAC = 00:00:00:93:3c:f4
 Host 85 (H₈₅) : VLAN 85 <-> VNI 8085, IPv4 = 100.0.85.108, MAC = 00:00:0e:a9:d8:a1
 Host 95 (H₉₅) : VLAN 95 <-> VNI 9095, IPv4 = 100.0.95.109, MAC = 00:00:0e:a9:d8:a2

Default gateway : 100.0.10.1 (virtual-gateway-address for irb.5010 L3 interface for VLAN 10)
 : 100.0.20.1 (virtual-gateway-address for irb.5020 L3 interface for VLAN 20)
 : 100.0.80.1 (virtual-gateway-address for irb.5080 L3 interface for VLAN 80)
 : 100.0.85.1 (virtual-gateway-address for irb.5085 L3 interface for VLAN 85)
 : 100.0.90.1 (virtual-gateway-address for irb.5090 L3 interface for VLAN 90)
 : 100.0.95.1 (virtual-gateway-address for irb.5095 L3 interface for VLAN 95)

All East-West (Inter DC – Intra VRF: Intra VNI and Inter VNI) traffic flows are tunneled through the GRE over IPsec tunnel connecting the two data centers as shown in Figure 5.9. E-W traffic for hosts in the same tenant does not transit the SRX data center firewall. Spine devices acting as service nodes are used as Layer 3 gateways in this scenario.

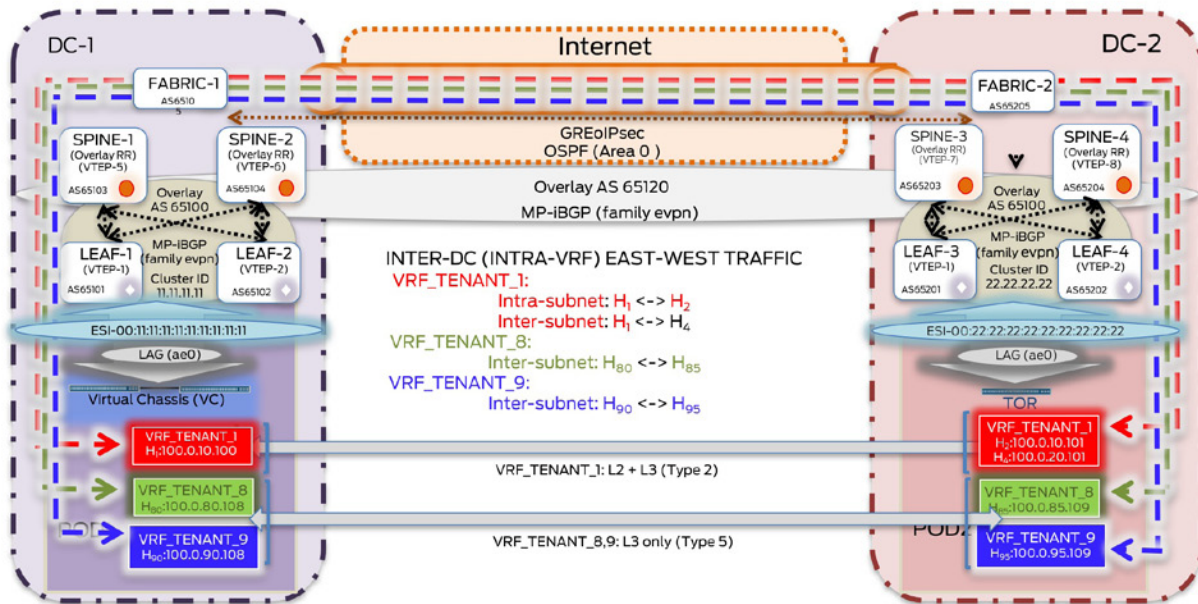


Figure 5.9 Inter-DC EW – Intra-VRF Intra/Inter Subnet Traffic Flows

IXIA Statistics – Inter-DC EW: Intra-VRF (Tenant-1) Intra-Subnet (VNI 5010 <-> VNI 5010)

(H1 <-> H2 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC4: Inter-DC EW: Intra-VRF-VRF-1 - Intra-subnet - H1 <-> H2	35,918	35,918	0	0.000	2,000.000	2,000.000

IXIA Statistics – Inter-DC EW: Intra-VRF (Tenant-1) Inter-Subnet (VNI 5010 <-> VNI 5020)

(H1 <> H4 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames ▲	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC4: Inter-DC EW: Intra-VRF-VRF-1 - Inter-subnet - H1 <> H4	31,708	31,708	0	0.000	2,000.000	2,000.000

IXIA Statistics – Inter-DC EW: Intra-VRF (Tenant-8) Inter-Subnet (VNI 5080 <-> VNI 5085)

(H80 <> H85 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames ▲	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC4: Inter-DC EW: Intra-VRF-VRF-8 - Inter-subnet - H80 <> H85	43,746	43,746	0	0.000	2,000.000	2,000.000

IXIA Statistics – Inter-DC EW: Intra-VRF (Tenant-9) Inter-Subnet (VNI 5090 <-> VNI 5095)

(H90 <> H95 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames ▲	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
UC4: Inter-DC EW: Intra-VRF-VRF-9 - Inter-subnet - H90 <> H95	35,744	35,744	0	0.000	2,000.000	2,000.000

Manual Service Chaining of E-W Inter-VRF Traffic Through DC Firewall

For traffic between hosts in different tenants, manual service chaining is demonstrated by hair pinning traffic through the local DC firewall for policy enforcement before forwarding it to the intended destination. In this use case, the DC firewall (SRX) is connected to a spine device, acting as the designated service node per POD. This service node is essentially a spine node with configuration similar to those of the other spine devices. It is not shown here, but the SRX can be connected to one or more of the other spine nodes, as desired, to provision for service node redundancy. Similarly, the service (spine) nodes could be connected to SRX clusters to provision for DC firewall redundancy. As elaborated previously, service chaining is treated as an IP path problem by getting traffic to the firewall. Details on firewall rules and policy enforcement are outside the scope of this book.

Let's elaborate upon the control plane and data plane details used to achieve the Layer 3 security insertion for E-W inter-tenant traffic across data centers.

Configuration

All configuration details as explained in the previous sections remain. Figure 5.10 illustrates the logical topology on how the firewall is inserted in the DC to process traffic between different tenants across DCs.

One physical link exists between SRX and service nodes: SVC_Node-1 and SV_Node-2 in DC-1 and DC-2, respectively.

DC-1 POD-1 devices have been provisioned for hosts in VLAN 80, IP routes for which reside in VRF_TENANT_8. Similarly, DC-2 POD-2 devices have been provisioned for hosts in VLAN 95, IP routes for which reside in VRF_TENANT_9. It is required for this inter-subnet traffic (VLAN 80 <> VLAN 95) between hosts for different tenants residing in two different data centers to traverse the SRX.

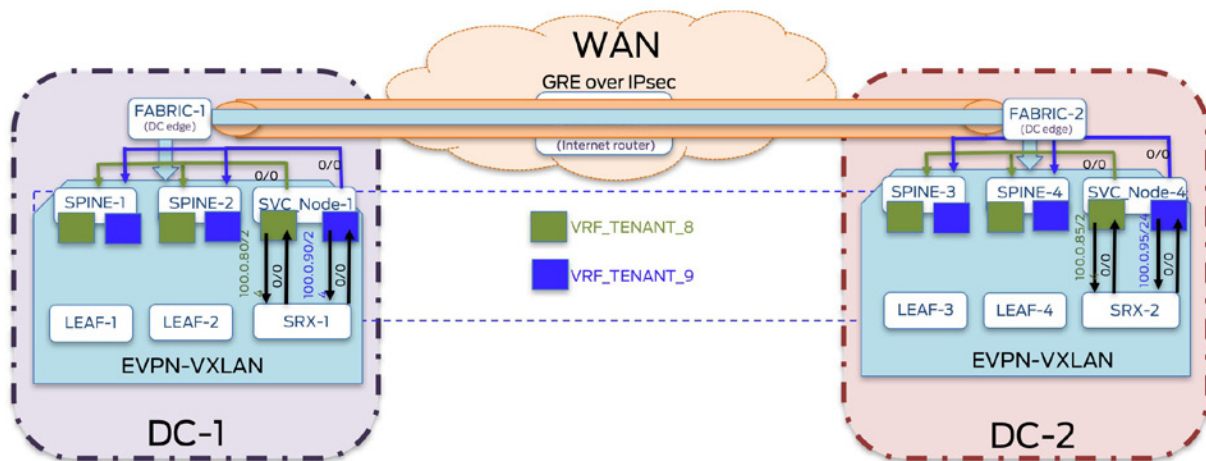


Figure 5.10 Layer 3 Security Insertion – Logical Topology

Similar to the building blocks explored in Chapter 1, in this case study, EVPN Type 5 routes have been used to get IP fabric tenant routes to the service node, which are further exchanged with the service block over eBGP sessions that terminate over IFLs residing in each tenant.

Route Exchange Between Service Nodes and Service Block

To achieve traffic hair pinning, service nodes in each data center establish an eBGP session from the tenant IP-VRF (here, Tenant-8 and Tenant-9) advertising VRF summary routes into the global route table on the SRX Series.

Each DC SRX advertises a default route to the service nodes in the respective data center, which is further distributed to all spine nodes.

VRF import policies on spine nodes ensure that the default route from only the local data center SRX is used. Thus, traffic from Tenant-8 in DC-1 to Tenant-9 in DC-2, will hair pin through SRX-1 before exiting DC-1.

Configuration

All configurations in the previous sections remain unchanged. For manual service chaining, an additional group 'UC4-EW-SRX-Svc_Chaining' has been applied on spine, fabric, and service block devices. Configuration highlights have been outlined in cases where they differ from those as described in the previous section. Snippets below show configuration highlights for service node (SVC_NODE-1), firewall (SRX-1), spine (SPINE-1), and fabric (FABRIC-1) devices in DC-1.

Service Node

```
jnpr@ SVC_Node-1 > show configuration groups
UC4-EW-SRX-Svc_Chaining
interfaces {
  xe-0/0/31:0 { ... }
  lo0 { ... }
  xe-0/0/30:2 {
    description «To SRX-1»;
    vlan-tagging;
    unit 8 {
      description "IFL in VRF_TENANT_8";
```

```

        vlan-id 80;
        family inet {
            address 10.10.10.80/31;
        }
    }
    unit 9 {
        description "IFL in VRF_TENANT_9";
        vlan-id 90;
        family inet {
            address 10.10.10.90/31;
        }
    }
...}
routing-options {...}
}

protocols {
    bgp {
        group EBGp-Underlay {
            type external;
            mtu-discovery;
            import underlay-in;
            export underlay-out;
            local-as 65060;
            bfd-liveness-detection {...}
            multipath multiple-as;
            neighbor 10.10.10.4 {
                local-address 10.10.10.5;
                peer-as 65105;
            }
            neighbor {...}
        } {...}
    }
    group IBGP-EVPN-DCI {
        type internal;
        mtu-discovery;
        local-address 10.1.1.60;
    }
    family evpn {
        signaling;
    }
    local-as 65120;
    bfd-liveness-detection {...}
    multipath multiple-as;
    neighbor 10.1.1.5 {
peer-as 65120;
    }
    neighbor 10.1.1.6 {
peer-as 65120;
    }
    neighbor 10.1.1.7 {
peer-as 65120;
    }
    neighbor 10.1.1.8 {
peer-as 65120;
    }
    neighbor 10.1.1.70 {
peer-as 65120;
    }
    }
...}

```

A full mesh of MP-iBGP connections between spine and service nodes across DCs exchange EVPN NLRI. Each IP-VRF is populated with Type 5 routes that include summary routes for each tenant. These summary routes for tenant IP-VRFs are exchanged over EBGp sessions in each VRF with DC firewall. Since the same AS (65060 above) is used for EBGp peering, the SRX does not advertise a summary route for Tenant-8 to Tenant-9, only the default route is advertised. Any Inter-VRF traffic is thus directed towards the SRX.

```

routing-instances {
  VRF_TENANT_8 {
    instance-type vrf;
    interface xe-0/0/30:2.8;
    interface lo0.80;
    route-distinguisher 10.1.60.80:85;
    vrf-export vrf-export-pol_VRF-8;
    vrf-target target:80:85;
    vrf-table-label;
    protocols {
      bgp {
        group EBGp-to-SRX {
          type external;
          export adv-vrf-summary-rt;
          local-as 65060;
          neighbor 10.10.10.81 {
            peer-as 65061;
          }
        }
      }
      evpn {
        ip-prefix-routes {
          advertise direct-nexthop;
          encapsulation vxlan;
          vni 8085;
          ...}
        }
      }
    }
  }
  VRF_TENANT_9 {
    instance-type vrf;
    interface xe-0/0/30:2.9;
    interface lo0.90;
    route-distinguisher 10.1.60.90:95;
    vrf-export vrf-export-pol_VRF-9;
    vrf-target target:90:95;
    vrf-table-label;
    protocols {
      bgp {
        group EBGp-to-SRX {
          type external;
          export adv-vrf-summary-rt;
          local-as 65060;
          neighbor 10.10.10.91 {
            peer-as 65061;
          }
        }
      }
      evpn {
        ip-prefix-routes {
          advertise direct-nexthop;
          encapsulation vxlan;
          vni 9095;
          ...}
        }
      }
    }
  }
}

policy-options {
  policy-statement LB { ... }
  policy-statement underlay-in { ... }
  policy-statement underlay-out { ... }
  policy-statement adv-vrf-summary-rt {
    term summ-rt {
      from protocol evpn;
    }
  }
  policy-statement vrf-export-pol_VRF-8 {
    term prefer-SRX1 {
      as-path orig-in-SRX1;
    }
    then {
      from {

```

```

        community add SRX1-default;
        community add vrf8-rt;
    }
}
term add-comm {
    then {
        community add DC1-comm;
        community add vrf8-rt;
        accept;
    }
}
...}

policy-statement vrf-export-pol_VRF-9 {
    term prefer-SRX1 {
        from {
            as-path orig-in-SRX1;
        }
        then {
            community add SRX1-default;
            community add vrf9-rt;
        }
    }
    term add-comm {
        then {
            community add DC1-comm;
            community add vrf9-rt;
            accept;
        }
    }
}
...}
community vrf8-rt members target:80:85;
community vrf9-rt members target:90:95;
community SRX1-default members target:10.1.1.60:0;
as-path asPathLength2 "{2,}";
as-path orig-in-SRX1 ".* 65061";
}

```

UC4-EW-UC4-EW-SRX-Svc_Chaining:

```

jnpr@SRX-1> show configuration groups
UC4-EW-SRX-Svc_Chaining
interfaces {
    lo0 { ... }
    xe-0/0/1 {
        description «To SVC_Node-1»;
        vlan-tagging;
        unit 8 {
            description «IFL in VRF_TENANT_8»;
            vlan-id 80;
            family inet {
                address 10.10.10.81/31;
            }
        }
        unit 9 {
            description "IFL in VRF_TENANT_9";
            vlan-id 90;
            family inet {
                address 10.10.10.91/31;
            }
        }
    }
}
...}

```

DC firewall (SRX)

```

routing-options {
    autonomous-system 65061;
    static {

```

```

        route 0.0.0.0/0 {
            reject;
            install;
        }
    }
    {...}
}
protocols {
    bgp {
        group EBGp-to-SRX {
            type external;
            local-as 65061;
            neighbor 10.10.10.80 {
                peer-as 65060;
            }
            neighbor 10.10.10.90 {
                peer-as 65060;
            }
        }
    }
    ...}

```

SRX injects a default route (configured statically) into the EBGp sessions terminating in the tenant IP-VRFs. This allows for Inter-VRF traffic to be hair-pinned through the DC firewall.

Spine Devices

Spine devices reject the default route from the remote DC SRX, so there is no sub-optimal forwarding as local DC firewall is preferred.

```

jnpr@SPINE-1 > show configuration groups
UC4-EW-SRX-Svc_Chaining
routing-instances {
    VRF_TENANT_8 {
        vrf-import vrf-import-pol_VRF-8;
    }
    VRF_TENANT_9 {
        vrf-import vrf-import-pol_VRF-9;
    }
}

policy-options {
    policy-statement vrf-import-pol_VRF-8 {
    term prefer-SRX1 {
        from community SRX2-default;
        then reject;
    }
    term add-comm {
        from community vrf8-rt;
        then accept;
    }
    }
    policy-statement vrf-import-pol_VRF-9 {
    term prefer-SRX1 {
        from community SRX2-default;
        then reject;
    }
    term add-comm {
        from community vrf9-rt;
        then accept;
    }
    }
    community vrf8-rt members target:80:85;
    community vrf9-rt members target:90:95;
    community SRX2-default members target:10.1.1.70:0;
    community SRX1-default members target:10.1.1.60:0;
}

```


Fabric Devices

```

jnpr@SRX-1> show configuration groups
UC4-EW-SRX-Svc_Chaining
interfaces {
  lo0          { ... }
  xe-0/0/1 {
    description "To SVC_Node-1";
    vlan-tagging;
    unit 8 {
      description "IFL in VRF_TENANT_8";
      vlan-id 80;
      family inet {
        address 10.10.10.81/31;
      }
    }
    unit 9 {
      description "IFL in VRF_TENANT_9";
      vlan-id 90;
      family inet {
        address 10.10.10.91/31;
      }
    }
  }
...}

```

Fabric devices exchange only service node loopbacks (not SRXs) to enable creation of full mesh of MP-iBGP sessions to exchange EVPN NLRI across DCs.

Verification

Tenant IP-VRFs on spine devices prefer the default route advertised by the local data center firewall:

```

jnpr@SPINE-1> show route table VRF_TENANT_8.inet.0 detail
VRF_TENANT_8.inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
0.0.0.0/0 (1 entry, 1 announced)
<<< 0/0 received by spine devices from local DC service nodes that direct traffic to connected SRX (not show here, but similar output seen for VRF_TENANT_9).

```

```

*EVPN    Preference: 170
        Next hop type: Indirect, Next hop index: 0
        Address: 0xa5bfff90
        Next-hop reference count: 4
        Next hop type: Router, Next hop index: 1704
        Next hop: 10.10.10.0 via xe-0/0/24:0.0, selected
        Session Id: 0x0
        Protocol next hop: 10.1.1.60
<...>
jnpr@SVC_Node-1> show route table VRF_TENANT_8.inet.0 detail
VRF_TENANT_8.inet.0: 7 destinations, 10 routes (7 active, 0 holddown, 0 hidden)
0.0.0.0/0 (2 entries, 1 announced)
*BGP     Preference: 170/-101
        Next hop type: Router, Next hop index: 1754
        Address: 0xa5ba650
        Next-hop reference count: 2
        Source: 10.10.10.81
        Next hop: 10.10.10.81 via xe-0/0/30:2.8, selected
        Session Id: 0x0
        State: <Active Ext>
        Peer AS: 65061
<...>

```

The SRX receives summary routes for tenant IP-VRFs from local DC service node.

```

jnpr@SRX-1> show route 100.0.80/24 detail
inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)

```

```

100.0.80.0/24 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
        Next hop type: Router, Next hop index: 549
        Address: 0x9db2f50
        Next-hop reference count: 6
        Source: 10.10.10.80
        Next hop: 10.10.10.80 via xe-0/0/1.8, selected
< ... >
jnpr@SRX-1> show route 100.0.95/24 detail
inet.0: 20 destinations, 20 routes (20 active, 0 holddown, 0 hidden)
100.0.95.0/24 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
        Next hop type: Router, Next hop index: 552
        Address: 0x9db3070
        Next-hop reference count: 6
        Source: 10.10.10.90
        Next hop: 10.10.10.90 via xe-0/0/1.9, selected
< ... >

jnpr@SRX-1> show interfaces descriptions
Interface      Admin Link Description
xe-0/0/1       up    up    To SVC_Node-1
xe-0/0/1.8     up    up    IFL in VRF_TENANT_8
xe-0/0/1.9     up    up    IFL in VRF_TENANT_9

```

Traffic Flows

Figure 5.11 illustrates the path traversed by traffic from host H80 (VRF_TENANT_8; IP 100.0.80.108) to host H95 (VRF_TENANT_8; IP 100.0.95.109). Traffic in the reverse direction is hair-pinned through the firewall (SRX-2) of DC-2.

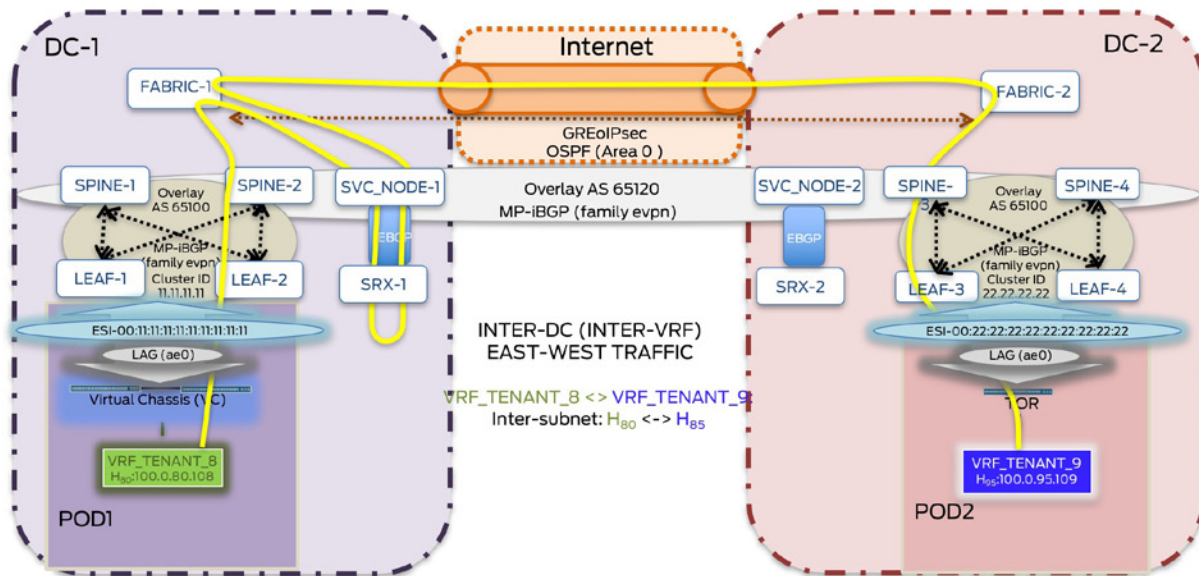


Figure 5.11 Manual Service Chaining for Inter-DC EW – Inter-VRF Inter Subnet Traffic ($H_{80} > H_{95}$)

Inter-VRF traffic from host in DC-1 to DC-2 is hair-pinned only through the firewall in DC-1.

SRX-1			Seconds: 62	Time: 03:03:15
Interface	Link	Input packets	(pps)	Output packets (pps)
xe-0/0/1	Up	2611872409	(1009)	2604884270 (1008)
SRX-2			Seconds: 148	Time: 03:05:11
Interface	Link	Input packets	(pps)	Output packets (pps)
xe-0/0/1	Up	2603726926	(0)	2590714762 (0)

Chapter 6

Data Center Interconnect – Layer 3 DCI (L3VPN-MPLS Core)

Contents

<i>High-Level Summary.....</i>	<i>214</i>
<i>Topology Description.....</i>	<i>215</i>
<i>East-West and North-South Traffic for Hosts in the Same Tenant (Intra-VRF: Inter-subnet).....</i>	<i>218</i>



This chapter discusses the last listed DCI option that orchestrates a simplified design for only Layer3 connectivity between data centers. Each case study presented here has three main sections:

- A high-level summary with an overview on the projected design and products being positioned.
- A design summary with technical details on the involved design components.
- Traffic scenarios that delve into configuration and verification details on common traffic use cases.

High-Level Summary

In previous design approaches, EVPN benefits can be leveraged inside a data center if there is a requirement for Layer 2 extension for Layer 2 workloads *inside* the DC but *not between* DCs. Each data center running EVPN-VXLAN can be integrated with the WAN (for example, L3VPN-MPLS core) to exchange tenant IP routes over the Layer 3 DCI that supports Layer 3 workloads only. In this example, routing is done at the spine layer within each data center. Spine devices also act as the DC edge connecting each DC to the WAN. All building blocks described in the previous case studies can be applied here to achieve manual service chaining for E-W and N-S traffic and have not been repeated again for the sake of brevity. A typical example for a smaller deployment (for example, 3-stage Clos with no fabric or super-spine devices) could use the QFX 5100/5200 Series as leaf devices and the QFX 10000 Series as spine and DC edge devices.

This case study focuses only on the DCI aspect of how the different data centers connect to the WAN, enabling E-W traffic flows across tenant hosts in different data centers (Inter-DC), separated by a L3VPN-MPLS backbone. EVPN-VXLAN is set up inside each DC and not extended over the existing WAN network. Pure EVPN Type 5 (VRF-to-VRF model) is used to exchange Layer 3 reachability information. Only inter-subnet communication between hosts across DCs is possible with this design. Additionally, this section also demonstrates N-S traffic between hosts across the WAN entering and exiting the data center boundary.

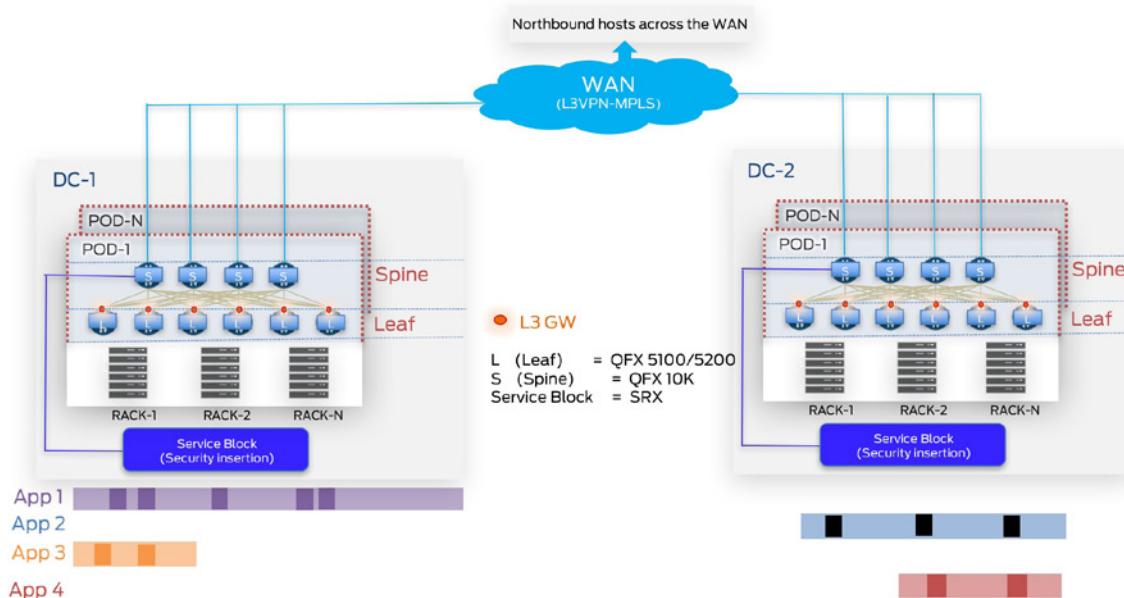


Fig 6.1 High Level Design Representation

Topology Description

Leaf devices: The QFX5100-24Q-2P have been used as leaf devices. Each POD consists of two leaf devices in each DC.

Spine devices: The QFX10002-72Q has been used as spine devices. Each POD consists of two spine devices in each DC.

Service block: Due to resource constraints, service block (firewall) has not been simulated in this example. However, design considerations from previous case studies can be re-used here. As described previously, designated spine devices in every POD can be used as service nodes to connect to the DC firewall.

CE devices: Access switches (QFX5100-48S-6Q – acts as a VC in POD-1 and a standalone switch in POD-2) and IXIA simulates CE devices in both PODs in both DCs. Servers can be directly plugged in to the TOR and leaf devices. To demonstrate the use of LACP, intermediate switches have been used here due to resource constraints. The term *host* (simulated on IXIA) implies any application entity (VM or container) that consumes an IP/MAC address.

WAN: Three MX104 routers have been used to simulate WAN PEs, while the QFX5100-48S-6Q has been used as a P device.

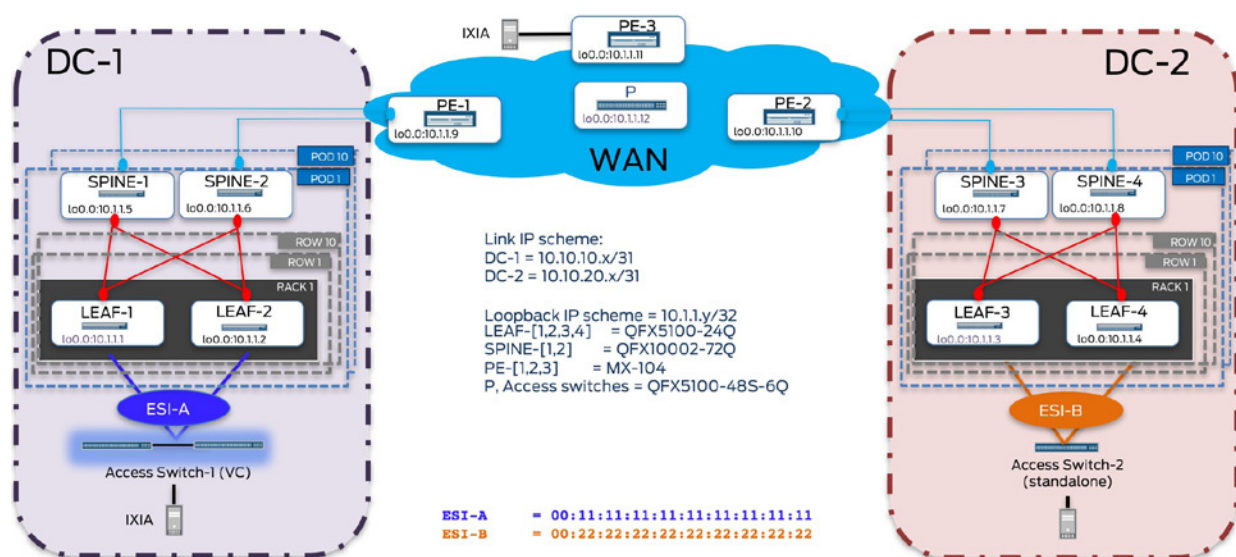


Figure 6.2 Chapter 6 Topology Overview

Design Summary

Traffic flows demonstrated are listed here.

- East-West (traffic between PODs across DCs):
 - Layer 2 (Intra-VRF Intra-subnet): Not applicable in this design approach (no Layer 2 extension across data centers).
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Not demonstrated in this case study, but the same building blocks as those described in the previous case studies can be used.

- North-South (traffic between hosts across the WAN and within the DC):
 - Layer 3 (Intra-VRF Inter-subnet): Traffic between hosts in different subnets for a given tenant, within data centers and across the WAN.
 - Layer 3 security insertion (Inter-VRF Inter-subnet): Not demonstrated in this case study, but the same building blocks as those described in the previous case studies can be used.

Access considerations:

- Two leaf devices are acting as a pair of redundant Top of Rack switches in each DC. CE-1 represents groups of hosts (for example, VMs/containers residing on servers) mapped to ESI-A (00:11:11:11:11:11:11:11) on ae0 for LEAF-1 and LEAF-2 in DC-1. Similarly, hosts on CE-2 are mapped to ESI-B (00:22:22:22:22:22:22:22) on ae0 for LEAF-3 and LEAF-4 in DC-2.

Underlay considerations:

- EBGp has been used to achieve underlay IP reachability (lo0 address reachability between VTEPs). A unique AS number is used per device. IP reachability information for devices in a specific data center is confined by the underlay and not propagated out of the data center boundary, by default.
- Spine or DC edge devices use eBGP-LU as the underlay to connect to the WAN (MPLS enabled on core-facing interfaces on the DC edge devices).
- MTU has been set on all physical interfaces for devices in a data center to account for VXLAN encapsulation.

Overlay considerations:

- L3 gateway placement:
 - VXLAN routing in this use case is done on the spine devices. Each spine device acts as a L3 gateway for VNIs residing in that POD.
 - Leaf devices acting only as Layer 2 gateways.
- EVPN NLRI exchange:
 - MP-iBGP sessions have been used to exchange EVPN NLRI within the data center.
 - To support E-W Inter-DC communication, summary routes inside a data center are further re-originated as IP routes into L3VPN towards the WAN. The spine devices acting as DC gateways also establish MP-BGP (family inet-vpn) sessions with the WAN devices (EVPN unaware), thus advertising tenant summary routes for the native data center domain into L3VPN. This allows L3 reachability exchange between tenants in different data centers. No additional functionality support is required to achieve this.
- Route reflection:
 - To avoid full-mesh of control-plane connections (EVPN MP-iBGP sessions within each data center), MP-iBGP EVPN sessions have been created between the leaf devices functioning as clients and spine devices being overlay route-reflectors, responsible for the exchange of EVPN NLRI. In DC-1 (POD-1), LEAF-1 and LEAF-2 act as clients to SPINE-1, SPINE-2 serving as RRs (cluster ID 11.11.11.11 in DC-1). In DC-2 (POD-1), LEAF-3 and LEAF-4 act as clients to

SPINE-3, SPINE-4 serving as RRs (cluster ID 22.22.22.22 in DC-2). Not shown here, but spine devices with different cluster IDs can provide for added redundancy.

- Full mesh of VXLAN tunnels (data-plane) exists with VTEPs located on each leaf and spine (including service nodes) node in each DC.

■ Service interface:

- VLAN aware EVPN service is in use (please refer to the Appendix for more details on EVPN service interfaces).

■ Host communication:

- Within the data center, EVPN Type 2 routes are exchanged between the leaf and spine devices to populate the tenant IP VRF on the DC edge devices, in each respective data center. Though not demonstrated in this case study, Type 5 routes can be used inside the DC to exchange L3 reachability information for data center hosts.

Integration with the WAN:

■ EVPN unaware core:

- WAN does not run EVPN – L3VPN-MPLS *only* core.

■ N-S traffic:

- To support N-S traffic, summary routes are advertised by DC edge devices in each DC to the WAN. (for example, H0 attached to PE-3 sends traffic to DC-1 to communicate with H1).

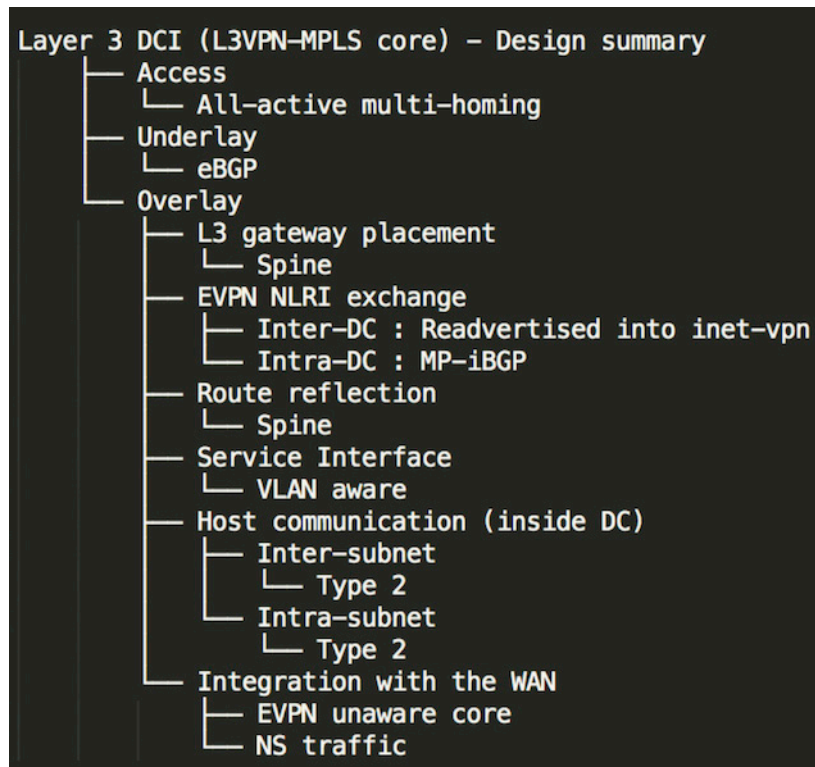


Figure 6.3 Design Summary for Layer 3 DCI

Traffic Scenarios

The rest of this chapter will demonstrate implementation details for East-West and North-South traffic for hosts in the same tenant (Intra-VRF: Inter-subnet):

- Configuration
- Verification
- Traffic flows

East-West and North-South Traffic for Hosts in the Same Tenant (Intra-VRF: Inter-subnet)

Figures 6.4 and 6.5 illustrate the logical topology for E-W traffic across PODs between DCs with details on the configuration and verification steps to follow.

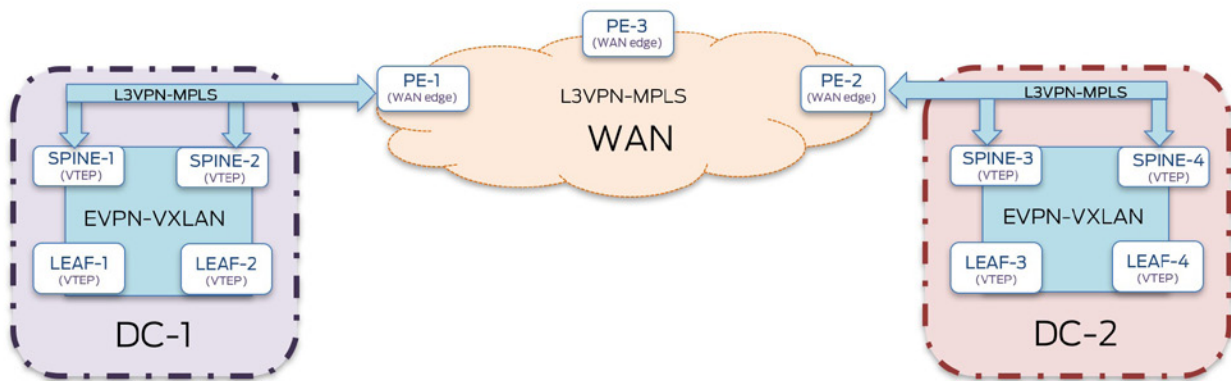


Figure 6.4 DCI Logical Topology (High-Level Overview)

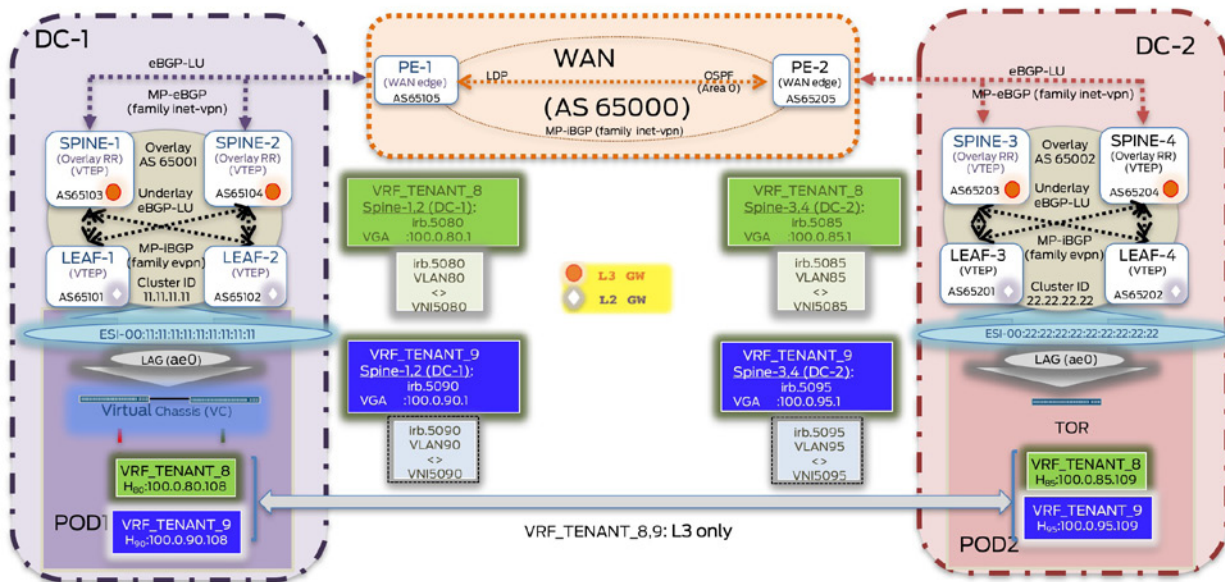


Figure 6.5 DCI Logical Topology (Detailed Overview)

Configuration

Configuration is divided across multiple groups, which are applied to the different devices as needed. Configuration highlights have been outlined in cases where they differ from those as described in the previous use cases. Snippets show configuration highlights for the leaf (LEAF-1), spine (SPINE-1) devices in DC-1, and WAN edge (PE-1) devices.

Leaf Devices

Configuration on the leaf devices is divided into three components: access, underlay, and overlay (intra-DC). Three configuration groups have been applied to all leaf devices in both DC-1 and DC-2: L3DCI_EW-Access, L3DCI_EW-Underlay, and L3DCI_EW-IntraDC.

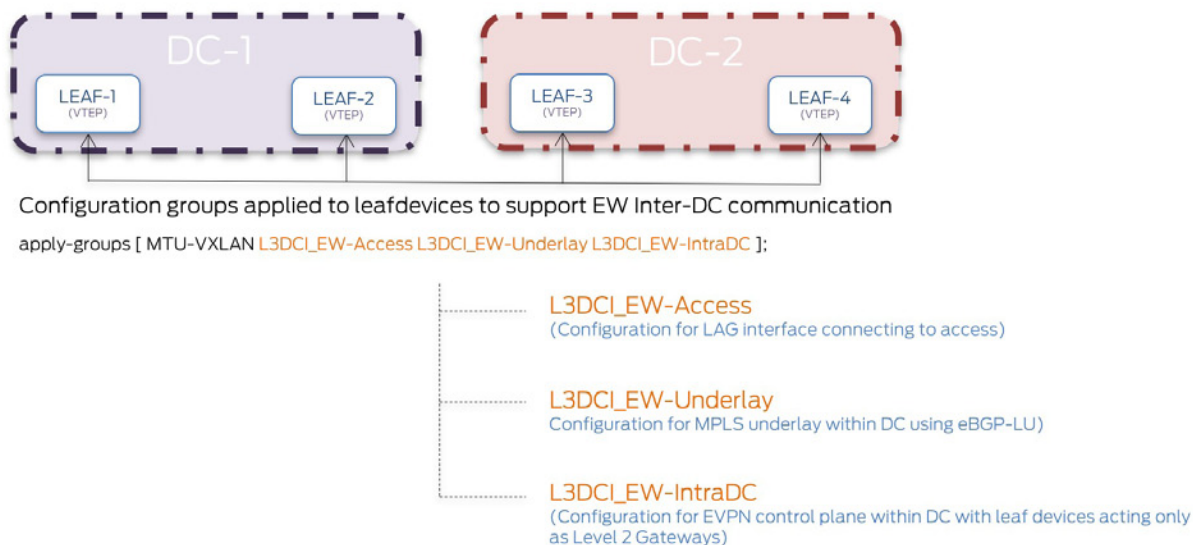


Figure 6.5 DCI Configuration – Leaf Layer

```
jnpr@LEAF-1> show configuration apply-groups
apply-groups [ MTU-VXLAN L3DCI_EW-Access L3DCI_EW-Underlay L3DCI_EW-IntraDC ];
```

jnpr@LEAF-1> show configuration groups L3DCI_EW-Access <<< <<< Configuration components for access remain unchanged from the previous use cases, shown here is the output on LEAF-1

```
interfaces {
  et-0/0/20 {
    description "LAG members to VC";
    hold-time up 180000 down 0;
    ether-options {
      802.3ad ae0;
    }
  }
  et-0/0/21 {
    description "LAG members to VC";
    hold-time up 180000 down 0;
    ether-options {
      802.3ad ae0;
    }
  }
}
ae0 {
  description "LAG VC <> LEAF-1";
  mtu 9192;
  esi {
    00:11:11:11:11:11:11:11:11:11:11:11;
    all-active;
  }
}
```

```

    }
    aggregated-ether-options {
        lacp {
            active;
            periodic fast;
            system-id 01:01:01:01:01:01;
        }
    }
    unit 0 {
        family ethernet-switching {
            interface-mode trunk;
            vlan {
                members all;
            }
        }
    }
}

jnpr@LEAF-1> show configuration groups L3DCI_EW-Underlay
interfaces {
    lo0 { ... }
    et-0/0/22 { ... }
    et-0/0/23 { ... }
}
routing-options { ... }
protocols {
    bgp {
        group EBG-Underlay {
            type external;
            mtu-discovery;
            import underlay-in; <<< eBGP sessions used in the underlay within a DC
            family inet {
                unicast;
            }
            export underlay-out;
            local-as 65101;
            bfd-liveness-detection {
                minimum-interval 350;
                multiplier 3;
                session-mode automatic;
            }
            multipath multiple-as;
            neighbor 10.10.10.8 {
                local-address 10.10.10.9;
                peer-as 65103;
            }
            neighbor 10.10.10.10 {
                local-address 10.10.10.11;
                peer-as 65104;
            }
        }
    }
}
policy-options {
    policy-statement LB { ... }
    policy-statement underlay-in { <<< Each leaf device exchanges loopback addresses over the EBGp session
with each connecting spine
        term acpt-remote-lo0 {
            from {
                route-filter 10.1.1.0/24 orlonger;
            }
            then accept;
        }
    }
    policy-statement underlay-out {
        term adv-local-lo0 {
            from {
                protocol direct;
            }
        }
    }
}

```

```

route-filter 10.1.1.0/24 orlonger;
    }
    then {
        next-hop self;
        accept;
    }
}
term default {
    then reject;
}
}
}

```

jnpr@LEAF-1> show configuration groups L3DCI_EW-IntraDC

```

protocols {
    bgp {
        group IBGP-EVPN-DC1 {
            type internal;
            description "Leaf clients for Spine RR";
            local-address 10.1.1.1; <<< Each leaf devices acts as a client establishing MP-iBGP (EVPN)
            session only with the spine devices in the local data center.
            import overlay-in;
            family evpn {
                signaling;
            }
            local-as 65001;
            bfd-liveness-detection {
                minimum-interval 350;
                multiplier 3;
                session-mode automatic;
            }
            multipath;
            neighbor 10.1.1.5 {
                peer-as 65001;
            }
            neighbor 10.1.1.6 {
                peer-as 65001;
            }
        }
    }
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
    }
}
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.1.1.1:100;
    vrf-import EVPN-IMPORT;
    vrf-target {
        target:1:100; <<< Manually defined route-targets are in use
    }
}
policy-options {
    policy-statement EVPN-IMPORT {
        term ESI_IN {
            from community comm-esi-in;
            then accept;
        }
        {...}
        term default {
            then reject;
        }
    }
    policy-statement overlay-in {...}
    community comm-esi-in members target:1:100;
}

```

```

vllans {
  bd5080 {
    vllan-id 80;
    vxllan {
      vni 5080;
    }
  }
  bd5090 {
    vllan-id 90;
    vxllan {
      vni 5090;
    }
  }
}

```

Spine Devices

Configuration on the spine devices is divided into three components as shown in Figure 6.7: underlay, overlay (intra-DC), and L3 DCI by re-originating summary routes into L3VPN. Three configuration groups have been applied to all spine devices in both DC-1 and DC-2: L3DCI_EW-Underlay, L3DCI_EW-IntraDC, and L3DCI_EW-L3VPN.

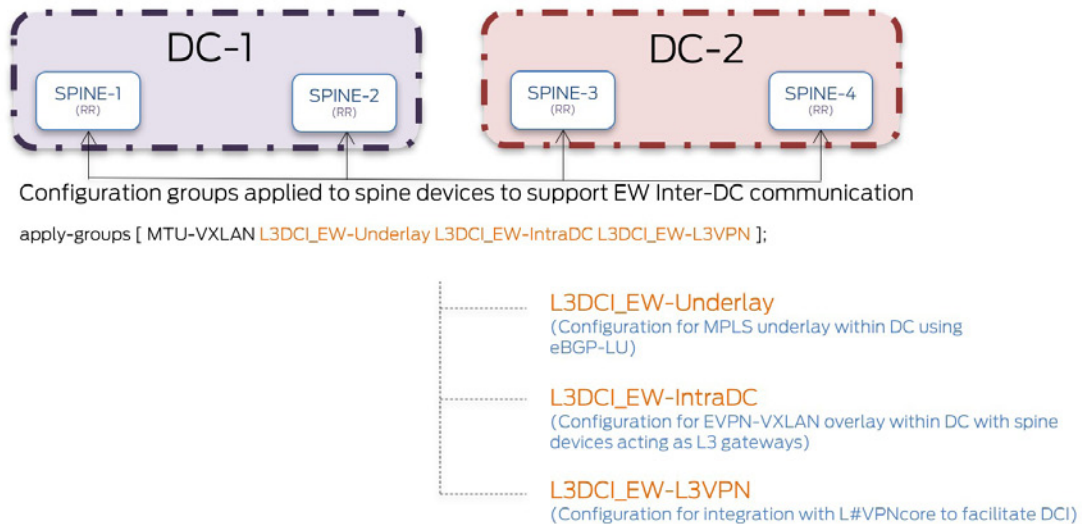


Figure 6.7 DCI Configuration – Spine Layer

```

jnpr@SPINE-1> show configuration apply-groups
apply-groups [ MTU-VXLAN L3DCI_EW-Underlay L3DCI_EW-IntraDC L3DCI_EW-L3VPN ];

```

```

jnpr@SPINE-1> show configuration groups L3DCI_EW-Underlay
interfaces {
  lo0 { ... }
  et-0/0/12 { ... }
  et-0/0/13 { ... }
  xe-0/0/24:0 { ... }
}
routing-options { ... }
protocols {
  bgp {
    group EBGp-Underlay {
      type external;
      mtu-discovery;
      import underlay-in;
    }
  }
}

```

```

    family inet { <<< eBGP sessions used in the underlay within a DC
        unicast;
    }
    export underlay-out;
    local-as 65103;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath multiple-as;
    neighbor 10.10.10.9 {
        local-address 10.10.10.8;
        peer-as 65101;
    }
    neighbor 10.10.10.13 {
        local-address 10.10.10.12;
        peer-as 65102;
    }
    neighbor 10.10.10.0 {
        local-address 10.10.10.1;
        family inet { <<< eBGP-LU enabled on core-facing interfaces on DC edge to connect to the
WAN. The 'resolve-vpn' knob installs a route in inet.3 for protocol next-hop allowing resolution of
received VPN routes
            labeled-unicast {
                resolve-vpn;
            }
        }
        peer-as 65105;
    }
...}
policy-options {
    policy-statement LB {...}
    policy-statement underlay-in {...}
    policy-statement underlay-out {...}
}

jnpr@SPINE-1> show configuration groups L3DCI_EW-IntraDC
interfaces {
    irb {
        unit 5080 { <<< Spine devices act as default layer 3 gateways for VNIs 5080 and 5090 for tenants 8
and 9 respectively
            proxy-macip-advertisement;
            description "Tenant 8 - vlan 80 - vni5080";
            family inet {
                address 100.0.80.2/24 {
                    virtual-gateway-address 100.0.80.1;
                }
            }
        }
        unit 5090 {
            proxy-macip-advertisement;
            description "Tenant 9 - vlan 90 - vni5090";
            family inet {
                address 100.0.90.2/24 {
                    virtual-gateway-address 100.0.90.1;
                }
            }
        }
    }
... }

lo0 {
    unit 80 {...}
    unit 90 {...}
}

vlangs {
    bd5080 {
        vlan-id 80;
        l3-interface irb.5080;
        vxlan {

```



```

        vni 5080;
    }
}
bd5090 {
    vlan-id 90;
    l3-interface irb.5090;
    vxlan {
        vni 5090;
    }
...}
protocols {
    bgp {
        group IBGP-EVPN-DC1-RR { <<< Spine devices act as overlay route-reflectors for leaf devices in
local data center
        type internal;
        local-address 10.1.1.5;
        family evpn {
            signaling;
        }
        export no-adv-type5;
        vpn-apply-export;
        cluster 11.11.11.11;
        local-as 65001;
        bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode automatic;
        }
        multipath;
        neighbor 10.1.1.1;
        neighbor 10.1.1.2;
    }
}
    evpn {
        encapsulation vxlan;
        extended-vni-list all;
        default-gateway no-gateway-community;
    }
}
policy-options {...}

routing-instances { <<< Type 2 routes are used to populate local DC tenant IP-VRFs
    VRF_TENANT_8 {
        instance-type vrf;
        interface irb.5080;
        interface lo0.80;
        route-distinguisher 10.1.5.80:85;
        vrf-target target:80:85;
        vrf-table-label;
    }
    VRF_TENANT_9 {
        instance-type vrf;
        interface irb.5090;
        interface lo0.90;
        route-distinguisher 10.1.5.90:95;
        vrf-target target:90:95;
        vrf-table-label;
    }
}
switch-options {
    vtep-source-interface lo0.0;
    route-distinguisher 10.1.1.5:100;
    vrf-import EVPN-IMPORT;
    vrf-target {
        target:1:100;
    }
}
...}

```

jnpr@SPINE-1> show configuration groups L3DCI_EW-L3VPN

```

protocols {
  bgp {
    group EBGP-L3VPN {
      type external;
      multihop;
      mtu-discovery;
      family inet-vpn { <<< Tenant summary routes are further re-advertised into L3VPN (MP-eBGP using
family inet-vpn). Here protocol next-hop on routes from remote DC are re-written to those of the WAN PEs
that can be resolved via inet.3 (using 'resolve-vpn')

      unicast;

    }
    local-as 65103;
    bfd-liveness-detection {
      minimum-interval 350;
      multiplier 3;
      session-mode automatic;
    }
    multipath;
    neighbor 10.1.1.9 {
      local-address 10.1.1.5;
      peer-as 65105;
    }
  }
...}

```

WAN Devices

Configuration on the WAN devices is that of a typical L3VPN core as shown in Figure 6.8. To support Inter-DC E-W communication, two configuration groups have been applied to PE devices connecting both DC-1 and DC-2: L3DCI_EW_L3VPN, and L3VPN-core.

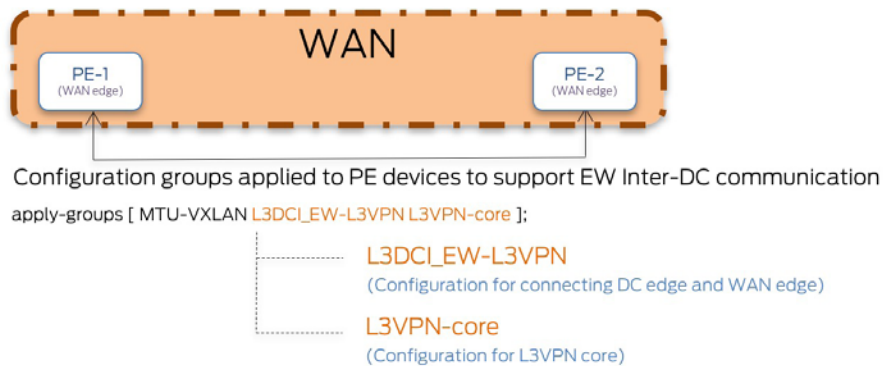


Figure 6.8 WAN Configuration (Inter-DC E-W)

```

jnpr@PE-1# show apply-groups
apply-groups [ MTU-VXLAN L3DCI_EW-L3VPN L3VPN-core ];

jnpr@PE-1# show groups L3DCI_EW-L3VPN
interfaces {
  xe-2/0/0 {...}
  xe-2/0/1 {...}
  lo0 {...}
}
routing-options {...}
protocols {
  bgp {
    group EBGP-Underlay {
      type external;

```

```

        mtu-discovery;
        import underlay-in;
        family inet { <<< Protocol next-hop on routes from remote DC are re-written to those of the WAN
PEs (MP-eBGP inet-vpn sessions) that can be resolved via inet.3 (using 'resolve-vpn' over eBGP LU used as
transport)
            labeled-unicast;
        }
        export underlay-out;
        local-as 65105;
        bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode automatic;
        }
        multipath multiple-as;
        neighbor 10.10.10.1 {
            local-address 10.10.10.0;
            family inet {
                labeled-unicast {
                    resolve-vpn;
                }
            }
            peer-as 65103;
        }
        neighbor 10.10.10.3 {
            local-address 10.10.10.2;
            family inet {
                labeled-unicast {
                    resolve-vpn;
                }
            }
            peer-as 65104;
        }
    }
}
group EBGP-L3VPN {
    type external;
    multihop;
    mtu-discovery;
    family inet-vpn {
        unicast;
    }
    local-as 65105;
    bfd-liveness-detection {
        minimum-interval 350;
        multiplier 3;
        session-mode automatic;
    }
    multipath;
    neighbor 10.1.1.5 {
        local-address 10.1.1.9;
        peer-as 65103;
    }
    neighbor 10.1.1.6 {
        local-address 10.1.1.9;
        peer-as 65104;
    }
}
}
}
policy-options {...}

jnpr@PE-1# show groups L3VPN-core
interfaces {
    xe-0/1/0 {...}
}
protocols {
    bgp {
        group IBGP-L3VPN {
            type internal;
            local-address 10.1.1.9;
            family inet-vpn { <<< WAN devices do not run EVPN and are configured for a typical L3VPN core.

```

EVPN-VXLAN tunnels are not extended across DCs. No routing instances are configured on WAN PEs. Summary routes for different tenants VRFs learned in the EVPN-VXLAN domain are propagated into L3VPN to facilitate DCI.

```
unicast;
}
  local-as 65000;
  neighbor 10.1.1.10 {
    peer-as 65000;
  }
  neighbor 10.1.1.11 {
    peer-as 65000;
  }
}
}
ospf {
  area 0.0.0.0 {
    interface xe-0/1/0.0;
    interface lo0.0;
  }
}
ldp {
  interface xe-0/1/0.0;
  interface lo0.0;
}
}
```

Verification

Spine devices in each data center (Spine-1 and Spine-3 outputs shown here) advertise tenant summary routes into L3VPN through ‘inet-vpn’ sessions with the WAN edge devices.

```
jnpr@SPINE-1> show route advertising-protocol bgp 10.1.1.9
VRF_TENANT_8.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
* 10.1.5.80/32          Self                  I
* 100.0.80.0/24         Self                  I
VRF_TENANT_9.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
* 10.1.5.90/32          Self                  I
* 100.0.90.0/24         Self                  I
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
10.1.5.80:85:10.1.5.80/32
*                          Self                  I
10.1.5.80:85:100.0.80.0/24
*                          Self                  I
10.1.5.90:95:10.1.5.90/32
*                          Self                  I
10.1.5.90:95:100.0.90.0/24
*                          Self                  I
```

```
jnpr@SPINE-3> show route advertising-protocol bgp 10.1.1.10
VRF_TENANT_8.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
* 10.1.7.85/32          Self                  I
* 100.0.85.0/24         Self                  I
VRF_TENANT_9.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
* 10.1.7.95/32          Self                  I
* 100.0.95.0/24         Self                  I
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
  Prefix                Nexthop              MED      Lclpref  AS path
10.1.7.80:85:10.1.7.85/32
*                          Self                  I
10.1.7.80:85:100.0.85.0/24
*                          Self                  I
10.1.7.90:95:10.1.7.95/32
*                          Self                  I
10.1.7.90:95:100.0.95.0/24
*                          Self                  I
```

Spine devices in each data center (Spine-1 and Spine-3 outputs shown here) learn the tenant summary routes for both tenants 8 and 9 from the remote DC via the WAN through L3VPN:

```
jnpr@SPINE-1> show route 100.0.85/24
VRF_TENANT_8.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.85.0/24      *[BGP/170] 1d 18:27:35, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65203 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302160
                  [BGP/170] 1d 18:27:35, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65204 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302208
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.1.7.80:85:100.0.85.0/24
                  *[BGP/170] 1d 18:27:35, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65203 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302160
10.1.8.80:85:100.0.85.0/24
                  *[BGP/170] 1d 18:27:35, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65204 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302208

jnpr@SPINE-1> show route 100.0.95/24
VRF_TENANT_9.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.95.0/24     *[BGP/170] 1d 18:58:04, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65204 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302192
                  [BGP/170] 1d 18:58:04, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65203 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302176
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.1.7.90:95:100.0.95.0/24
                  *[BGP/170] 1d 18:58:04, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65203 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302176
10.1.8.90:95:100.0.95.0/24
                  *[BGP/170] 1d 18:58:04, localpref 100, from 10.1.1.9
                  AS path: 65105 65000 65205 65204 I, validation-state: unverified
                  > to 10.10.10.0 via xe-0/0/24:0.0, Push 302192

jnpr@SPINE-3> show route 100.0.80/24
VRF_TENANT_8.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.80.0/24     *[BGP/170] 1d 18:58:54, localpref 100, from 10.1.1.10
                  AS path: 65205 65000 65105 65103 I, validation-state: unverified
                  > to 10.10.20.0 via xe-0/0/24:0.0, Push 301552
                  [BGP/170] 1d 19:17:19, localpref 100, from 10.1.1.10
                  AS path: 65205 65000 65105 65104 I, validation-state: unverified
                  > to 10.10.20.0 via xe-0/0/24:0.0, Push 301408
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.1.5.80:85:100.0.80.0/24
                  *[BGP/170] 1d 18:58:54, localpref 100, from 10.1.1.10
                  AS path: 65205 65000 65105 65103 I, validation-state: unverified
                  > to 10.10.20.0 via xe-0/0/24:0.0, Push 301552
10.1.6.80:85:100.0.80.0/24
                  *[BGP/170] 1d 19:17:19, localpref 100, from 10.1.1.10
                  AS path: 65205 65000 65105 65104 I, validation-state: unverified
                  > to 10.10.20.0 via xe-0/0/24:0.0, Push 301408

jnpr@SPINE-3> show route 100.0.90/24
VRF_TENANT_9.inet.0: 7 destinations, 9 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100.0.90.0/24     *[BGP/170] 1d 19:01:04, localpref 100, from 10.1.1.10
                  AS path: 65205 65000 65105 65103 I, validation-state: unverified
```

```

> to 10.10.20.0 via xe-0/0/24:0.0, Push 301536
[BGP/170] 1d 19:19:29, localpref 100, from 10.1.1.10
AS path: 65205 65000 65105 65104 I, validation-state: unverified
> to 10.10.20.0 via xe-0/0/24:0.0, Push 301424
bgp.l3vpn.0: 18 destinations, 18 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.1.5.90:95:100.0.90.0/24
*[BGP/170] 1d 19:01:04, localpref 100, from 10.1.1.10
AS path: 65205 65000 65105 65103 I, validation-state: unverified
> to 10.10.20.0 via xe-0/0/24:0.0, Push 301536
10.1.6.90:95:100.0.90.0/24
*[BGP/170] 1d 19:19:29, localpref 100, from 10.1.1.10
AS path: 65205 65000 65105 65104 I, validation-state: unverified
> to 10.10.20.0 via xe-0/0/24:0.0, Push 301424

```

Traffic Flows

For this use case, both PODs are situated in different data centers (Inter DC). For tenant 8 (VRF_TENANT_8), hosts H80 (DC-1) and H85 (DC-2) belong to VLAN 80 and 85, respectively. For tenant 9 (VRF_TENANT_9), hosts H90 (DC-1) and H95 (DC-2) belong to VLAN 90 and 95, respectively. In DC-1, VLAN 80 and 90 are mapped to VNI 5080 and 5090, respectively, while in DC-2 VLAN 85 and 95 are mapped to VNI 5085 and 5095, respectively.

CE-1 hosts multi-homed to LEAF-1 and LEAF-2 (ESI-A = 00:11:11:11:11:11:11:11):
 Host 80 (H80) : VLAN 80 <> VNI 5080, IPv4 = 100.0.80.108, MAC = 00:00:0e:a9:d8:9f
 Host 90 (H90) : VLAN 90 <> VNI 5090, IPv4 = 100.0.90.109, MAC = 00:00:0e:a9:d8:a0

CE-2 hosts multi-homed to LEAF-3 and LEAF-4 (ESI-B = 00:22:22:22:22:22:22:22):
 Host 85 (H85) : VLAN 85 <> VNI 5085, IPv4 = 100.0.85.108, MAC = 00:00:0e:a9:d8:a1
 Host 95 (H95) : VLAN 95 <> VNI 5095, IPv4 = 100.0.95.109, MAC = 00:00:0e:a9:d8:a2

Default gateway : 100.0.80.1 (virtual-gateway-address for irb.5080 L3 interface for VLAN 80)
 : 100.0.90.1 (virtual-gateway-address for irb.5090 L3 interface for VLAN 90)
 : 100.0.85.1 (virtual-gateway-address for irb.5085 L3 interface for VLAN 85)
 : 100.0.95.1 (virtual-gateway-address for irb.5095 L3 interface for VLAN 95)

All E-W (Inter DC – Intra VRF: Inter VNI) traffic flows are transit through the L3VPN WAN connecting the two data centers as shown in Figures 6.9 and 6.10.

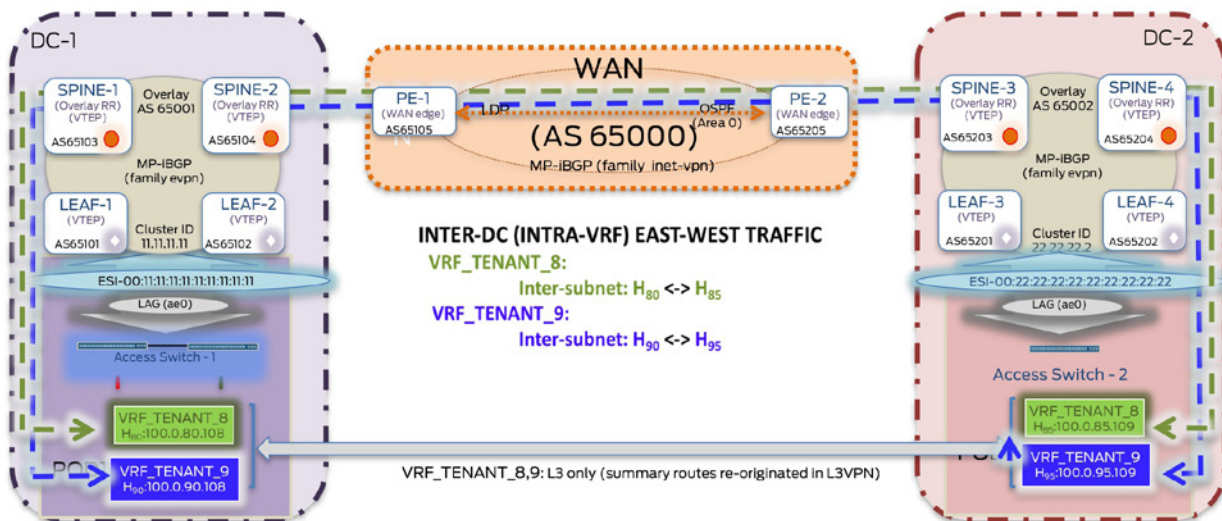


Figure 6.9 Inter-DC EW – Intra-VRF Inter Subnet Traffic Flows

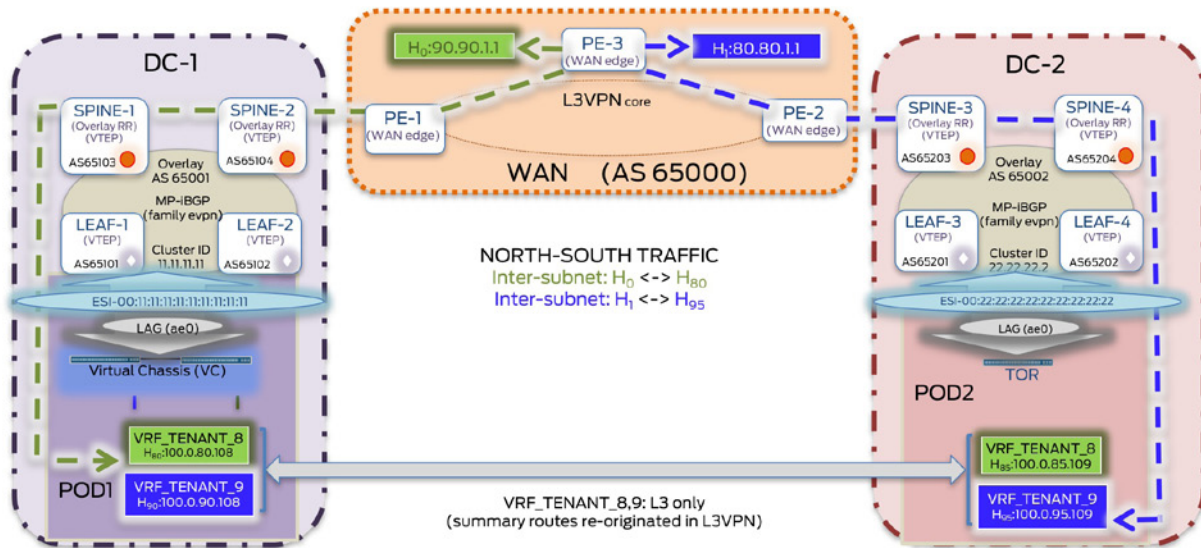


Figure 6.10 North-South Inter Subnet Traffic Flows

IXIA Statistics - Inter-DC EW: Intra-VRF (Tenant-8/9) Inter-Subnet
(VNI 5080 <-> VNI 5085)
(VNI 5090 <-> VNI 5095)

(H80 <-> H85 1000 flows @ 1000 pps)
(H90 <-> H85 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
L3DCI: Inter-DC EW: Intra-VRF-VRF-8 - Inter-subnet - H80 <-> H85	35,744	35,744	0	0.000	2,000.000	2,000.000
L3DCI: Inter-DC EW: Intra-VRF-VRF-9 - Inter-subnet - H90 <-> H95	35,744	35,744	0	0.000	2,000.000	2,000.000

IXIA Statistics - NS (H0 <-> H80, H1 <-> H95)

(H0 <-> H80 1000 flows @ 1000 pps)
(H1 <-> H95 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames	Frames Delta	Loss %	Tx Frame Rate	Rx Frame Rate
L3DCI: NS traffic optimization : Remote H0 90.90.1.1 <-> H80 100.0.80.108	35,740	35,740	0	0.000	2,000.000	2,000.000
L3DCI: NS traffic optimization : Remote H1 80.80.1.1 <-> H95 100.0.95.109	35,740	35,740	0	0.000	2,000.000	2,000.000

Appendix

Contents

Multicast/VXLAN Packet Walkthrough232

Understanding OVSDB234

OVSDB/VXLAN Packet Walkthrough235

EVPN Service Interfaces236

Building a Data Center - Routing at the Leaf Layer (Spine Devices Servicing Each POD)245



Multicast/VXLAN Packet Walkthrough

Figures A.1 and A.2 depict the MAC address learning process, while Figure A.3 illustrates data traffic forwarding over Multicast/VXLAN for end-hosts in the same VNI (intra-subnet communication). VMs A, C, and E belong to the same tenant and are connected through the same VXLAN segment identified by VNID red (for example, 5010). VTEPs 1, 2, and 3 that have end-hosts in this VNID red use the same multicast group Mcast_red (for example, IP = 224.9.9.9, MAC = 00:01:5e:09:09:09). This allows flooding of traffic for address learning and discovery to be limited to only interested VTEPs. Each VTEP learns about its own hosted VMs in the data plane (for example, GARP, etc.). In the absence of a control plane, when VTEPs 1 and 2 locally learn addresses for VMs A and C, respectively, they do not propagate this information to each other or to any other VTEP in the same VNI. As a result, address learning is done using flood and learn.

VM-A (IP_A = 10.10.10.100) needs to send traffic to VM-C (IP_C = 10.10.10.101) but does not know its MAC address. As a result, VM-A sends an ARP request to find the MAC address of VM-C associated with IP_C (Destination MAC = FF:FF:FF:FF:FF:FF).

This ARP request packet is received by VTEP-1 that hosts VM-A. VTEP-1 encapsulates this Ethernet broadcast frame into a UDP header (Source port = hash of inner packet L2/L3/L4 headers; Destination port = 4789 for VXLAN). It further adds an IP header (Source IP = VTEP-1 lo0 address; Destination IP = Multicast group address Mcast_red – 224.9.9.9).

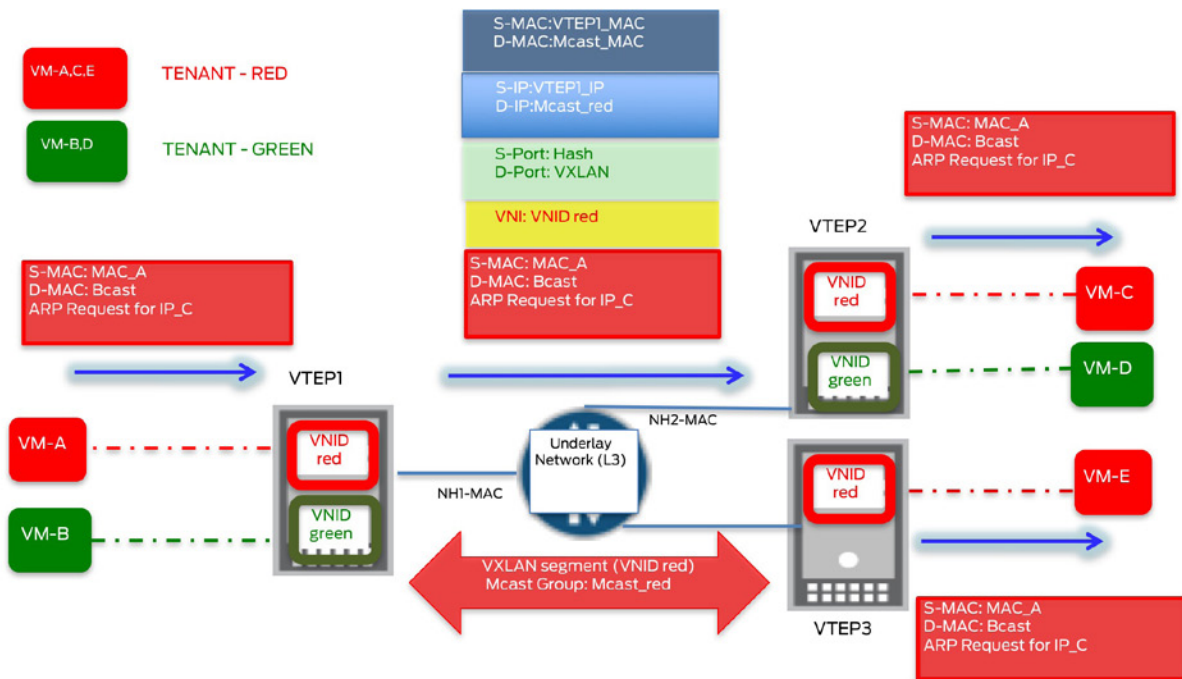


Figure A.1 ARP Request VM-A > VM-C

This ARP request packet is processed by all the VTEPs that are members of the distribution tree for this multicast group. Each VTEP will decapsulate the packet and evaluate the VXLAN header. If the contained VNID is locally configured, then the receiving VTEP forwards the decapsulated packet, therefore, ARP requests to VMs in that VNID. A forwarding table entry is created mapping the remote MAC (MAC_A for VM_A), VNID it was learned for (VNID red = 5010) and remote VTEP it was learned from (VTEP-IP lo0 address), by both VTEPs 2 and 3.

Since the received packet is for a locally-configured VNID, both VTEPs 2 and 3 forward the decapsulated ARP request packet to end-hosts in that VNID, therefore, VM-C and VM-E, respectively.

VM-C responds with a unicast ARP response packet which is received by VTEP-2.

Since VTEP-2 had created a forwarding table entry for MAC_A, it knows how to deliver the ARP reply packet. VTEP-2 adds VXLAN encapsulation and sends the ARP response as a unicast IP packet to VTEP-1 (Source IP address = VTEP-2 lo0 IP; Destination IP address = VTEP-1 lo0 IP). Since VTEPs 1 and 2 have created forwarding entries for remote VMs C and A respectively, they can allow for ARP suppression using proxy ARP to reduce flooding for future requests.

On receiving this encapsulated ARP response, VTEP-1 performs the VNID check, decapsulates it, and then forwards the same to VM-A.

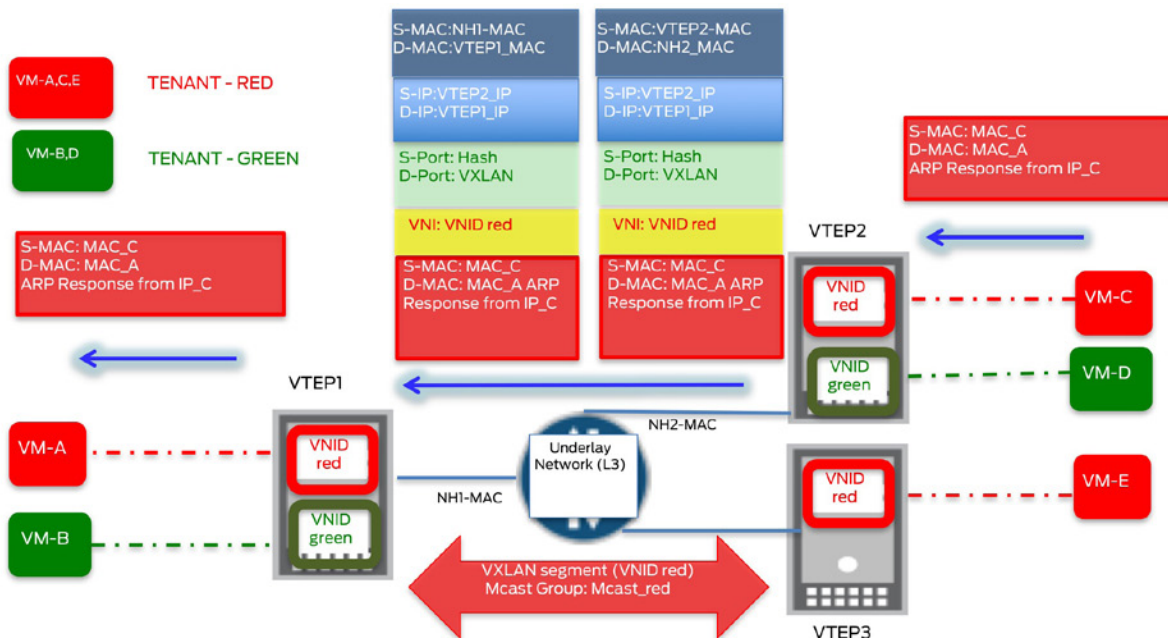


Figure A.2 ARP Response VM-C > VM-A

Traffic Forwarding (Data Plane)

Data packets from VM-A to VM-C are unicast and transported over the VXLAN tunnels. The IP underlay will only route these based on the outer IP header and do not have any visibility into the VXLAN header or inner packet.

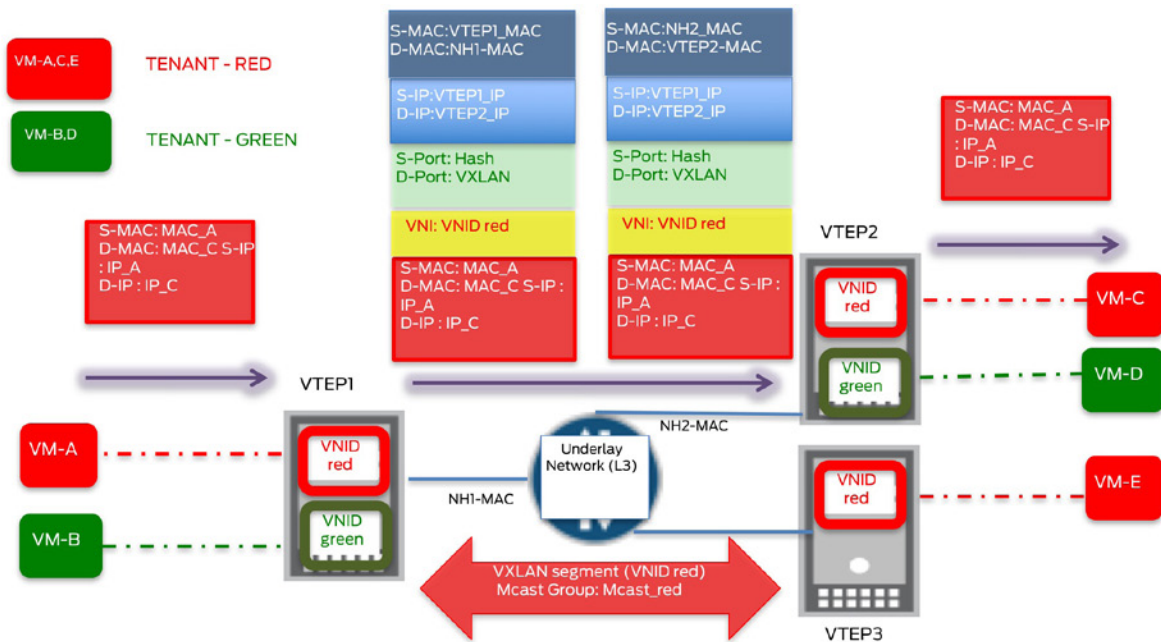


Figure A.3 Traffic Forwarding VM-A > VM-C

For Unicast (Static) VXLAN, MAC address learning is also done in the data plane. The difference being, instead of multicast for Multicast/VXLAN, the ingress VTEP-1 would be creating unicast copies of the ARP request and sending it to both VTEP-2 and VTEP-3 (ingress replication). All other steps for MAC learning and traffic forwarding as indicated above would be applicable. Next, let's describe the traffic flows for OVSDb/VXLAN and EVPN-VXLAN.

Understanding OVSDb

Open vSwitch Database Management Protocol (OVSDb) is a general purpose database management protocol that allows retrieval or modification of the configuration and operational state of a device from a centralized entity like a SDN controller. When a hardware VTEP gateway is used, OVSDb employs a client-server mechanism to create a communication channel, which a SDN controller can configure and manage VLAN to VNI mappings, retrieve locally learned MAC addresses from the VTEP, and push these MAC addresses to remote VTEPs. OVSDb thus provides control plane functionality to VXLAN while enabling the creation of programmatic networks.

OVSDb Essentials

OVSDb uses JSON RPCs, which is a simple lightweight remote procedure call protocol to define both the schema format and the wire protocol format. These RPC methods specify the different database operations that can be done (for example, Transact, Monitor, Update, etc.). A client-server model is used to establish a communication channel. The hardware VTEP gateway runs an instance of the OVSDb server (OSD – OVSDb-Server Daemon) and a local OVSDb client (VGD – VTEP Gateway Daemon) that acts as an intermediary between other Junos OS components and the OVSDb server. A separate OVSDb client runs on the SDN controller. Using the Junos OS CLI, local OVSDb client supplies controller IP information to the server, so that can initiate or receive a connection from the controller client. A database schema for hardware VTEPs consists of different tables and is a blueprint of how the OVSDb

database is constructed on the OVSDB server. Both OVSDB clients can query or update this database on the OVSDB server to retrieve or modify configuration or operational state.

- *Provisioning:* If a device supports the hardware VTEP schema, a controller can provision it for the relevant VLAN and VXLAN configuration using the respective database tables. For example, using the `Physical_Switch`, `Physical_Port`, and `Logical_Switch` tables, a controller can map a VLAN on a physical port to a VXLAN segment (VLAN to VNI mapping), which is reflected in the OVSDB database. On receiving this configuration change, local OVSDB client (VGD) installs it via `netconf` in the Junos OS CLI configuration.
- *Address learning:* In addition to management, OVSDB also provides control plane functionality for VXLAN. Locally-learned MAC addresses are published to the controller and remote MAC addresses are learned by the controller. For example, using the `Ucast_Macs_Local` table MACs learned locally from the hardware VTEP gateway are published to the controller as a local database update whereas the `Ucast_Macs_Remote` table allows the remote MAC, VNI, VTEP mapping to be pushed from the controller to the hardware VTEP. On receiving these routes from the controller via OVSDB, local OVSDB client (VGD) installs these using L2ALD.
- *BUM traffic handling:* Two types of multicast replication modes are supported – Ingress Replication (IR – headend replication directly to remote VTEPs) or Service Node replication (Ingress node sends BUM traffic to separate entities that handle replication).

OVSDB/VXLAN Packet Walkthrough

Figure A.4 illustrates the MAC address learning and data traffic forwarding over OVSDB/VXLAN for end-hosts (intra-subnet communication). VM-A resides on the server acting as a software VTEP (VTEP2). BMS-A is a bare-metal server that is VXLAN unaware and is connected to hardware VTEP (VTEP1 – for example, MX/QFX). To establish communication between VM-A and BMS-A (same tenant), a VXLAN tunnel identified by VNID red is created between VTEPs 1 and 2. Each VTEP learns about its own end hosts via data plane learning (for example, GARP). In the presence of a control plane, when VTEPs 1 and 2 locally learn addresses for BMS-A and VM-A, respectively, they propagate this information to the controller, which further relays this to VTEPs in the same VNI using OVSDB.

Remote MAC Learning (Control Plane)

VTEP-1 learns the MAC address of VM-A on a local non-VTEP IFL through data plane learning. VTEP-1 publishes this MAC address to the SDN controller using the `Ucast_Macs_Local` table. This provides a mapping of VM-A MAC address, associated VNI (VNID red), and VTEP-1 IP address (hypervisor address) as the physical locator. This conveys information on what VTEP needs to be used to reach this MAC. Optionally, IP address corresponding to the MAC for VM-A might also be included for ARP suppression.

The SDN controller relays this information to all VTEPs in the same VNI including VTEP-2. This is done using `Ucast_Macs_Remote` table that essentially provides a mapping of the MAC address to the VXLAN tunnel, therefore, remote MAC (MAC for VM-A), VNI (VNID red), and VTEP (VTEP-1 address) mapping is pushed from the controller to VTEP-2. On receiving these routes from the controller via OVSDB, the local OVSDB client (VGD) installs these using L2ALD. Due to the provisioning changes made by the controller, the Junos OS CLI will have the VLAN to VNI mapping configured, allowing VTEP-2 to determine which VLAN the MAC route belongs to.

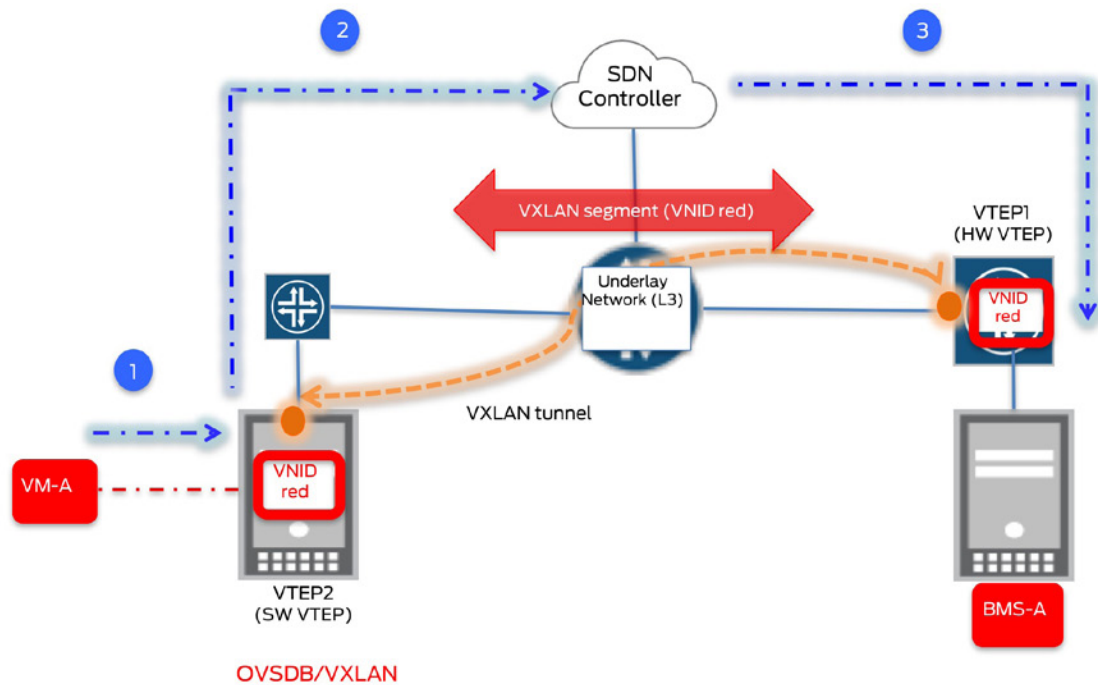


Figure A.4 Address Learning VM-A > BMS-A

Traffic Forwarding (Data Plane)

Destination address lookup to send traffic from BMS-A to VM-A is successful as a result. Similar to what is described in Figure A.3, traffic is VXLAN -ncapsulated by VTEP-2, and transported over the IP underlay to VTEP-1, which decapsulates and delivers the same to VM-A.

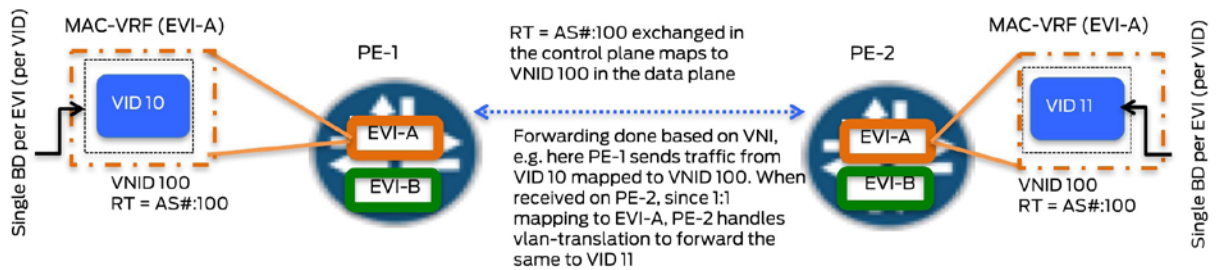
EVPN Service Interfaces

An EVPN instance (EVI) can span multiple PEs. Each PE maintains a separate MAC Virtual Routing and Forwarding table (MAC-VRF) for each EVI. Each EVI can contain one or more broadcast domains. A broadcast domain within an EVI is identified by an Ethernet Tag. This Ethernet Tag is encoded in the EVPN NLRI that is exchanged between PEs for control plane learning. When a PE receives an EVPN NLRI that contains an Ethernet Tag, it needs to be able to map this to the attached CE VLAN-ids (VIDs). The function of this mapping is defined by the EVPN service interface. By defining CE-VID-to-broadcast-domain assignment, service interfaces help create service contexts in the core. This allows for easy integration with the EVPN infrastructure and flexible service offerings (defines how a CE VLAN is plugged into an EVI, transported across the core using an Ethernet Tag and serviced at the receiving PE). There are three different types of service interfaces – VLAN Based, VLAN Bundle, and VLAN Aware. Table A.1 and Figures A.5, A.6, and A.7 compare these different service interfaces.

Table A.1 VLAN Service Interfaces

	VLAN Based Service Interface	VLAN Bundle Service Interface	VLAN Aware Service Interface
VLAN-ID to EVI mapping	1:1 - With this service interface, an EVI consists of only a single broadcast domain (e.g., a single VLAN). Therefore, there is a 1:1 mapping between a vlan-id and a MAC-VRF. When VXLAN is used as the data plane encapsulation for EVPN, VNI maps directly to the EVI for VLAN-based services.	N:1 - With this service interface, an EVI consists of multiple broadcast domains (e.g., multiple VLANs) with all VLANs sharing the same bridge table for a given EVI. Therefore, there is a N:1 mapping between vlan-ids and a MAC-VRF. When VXLAN is used as the data plane encapsulation for EVPN, for VLAN-bundle services, all vlan-ids share the same VNI that maps directly to the EVI.	N:1 - With this service interface, an EVI consists of multiple broadcast domains (e.g., multiple VLANs) with each VLAN having its own bridge table -- i.e., multiple bridge tables (one per VLAN) are maintained by a single MAC-VRF corresponding to the EVPN instance. Therefore, there is a N:1 mapping between vlan-ids and a MAC-VRF. When VXLAN is used as the data plane encapsulation for EVPN, for VLAN-aware services, VNI maps to a bridge table within the EVI.
Number of CE VLAN-IDs per EVI	1 - Each CE-VID will be individually mapped to a different EVI (single bridge domain per PE for the EVI). For example, for up to 4K CE-VIDs this means a total number of 4K MAC-VRFs/EVIs is required per PE.	N - All CE-VIDs are mapped to the same EVI (single bridge domain per PE for the EVI). For example, for up to 4K CE-VIDs, all 4K are mapped to the same EVI so only 1 MAC-VRF/EVI is required per PE.	N - Each CE-VID will be individually mapped to a different bridge table in the same EVI (multiple bridge domains per PE for the EVI). For example, for up to 4K CE-VIDs, all 4K are mapped to the same EVI to 4K bridge-domains, only 1 MAC-VRF/EVI is required per PE.
Number of bridge domains per EVI	1 - Since a MAC-VRF corresponds to a single VLAN, it consists of a single bridge table corresponding to that VLAN.	1 - MAC-VRF corresponds to multiple VLANs, but only a single bridge table is maintained per MAC-VRF, which means multiple VLANs share the same bridge table.	N - MAC-VRF corresponds to multiple VLANs and multiple bridge tables one for each vlan are maintained per MAC-VRF which means each VLAN is mapped to its corresponding bridge table.
VNID to EVI mapping	Ethernet Tag ID is set to 0 for VLAN-based mode, where there is a 1:1 mapping between EVI and VNI. Bridge Domain ID or Broadcast Domain ID can be derived from the Route Target (RT) associated with this route.	Ethernet Tag ID is set to 0 for VLAN-bundle mode, where there is a N:1 mapping between VNI and EVI. A single RT (per-EVI) is exchanged in the control plane. All VIDs are mapped to the same VNID (per-EVI) in the data plane. To uniquely identify a given VLAN, inner packet lookup is necessary to process the vlan-id.	Ethernet Tag ID is be set to the VNI, for VLAN-aware mode, where multiple VNIs are mapped to the same EVI, the Ethernet Tag. At the receiving PE, the Bridge Domain (BD) ID is derived from the combination of RT + VNI - e.g., the RT identifies the associated EVI on that PE and the VNI identifies the corresponding BD ID within that EVI. Multiple subnets each represented by a unique VNI are mapped to a single EVI e.g., if a tenant has multiple segments/ subnets each represented by a VNI then all the VNIs for that tenant are mapped to a single EVI. In this case, EVI represents the tenant and not the subnet.

	VLAN Based Service Interface	VLAN Bundle Service Interface	VLAN Aware Service Interface
Overlapping MACs across VLANs	Separation of traffic and MAC address advertisements is achieved at the per bridge-domain level.	Separation of traffic and MAC address advertisements is not achieved at the per bridge-domain level. As such, MAC addresses must be unique across all VLANs for that EVI.	Separation of traffic and MAC address advertisements is achieved at the per bridge-domain level.
VLAN translation	Forwarding decisions made based on the VNI associated to each broadcast domain, so a given broadcast domain can be represented by multiple VIDs, therefore, VLAN normalization can be supported.	All vlans share the same VNI so based on the VNID alone there is no unique way to identify a given broadcast-domain (inner packet lookup to read VID is needed). As a result, no VID translation is allowed. Different CEs connected to different PEs use the same CE-VIDs for the same EVI.	Forwarding decisions made based on the VNI associated to each broadcast domain, so a given broadcast domain can be represented by multiple VIDs, therefore, VLAN normalization can be supported.
Advantages	Granular control, efficient flooding scope, support for overlapping address space across VLANs and VLAN-normalization.	Low control plane overhead, ease of provisioning.	Control over the customer broadcast domain, efficient flooding scope, reduced control plane state and scale limitations (unlike VLAN-based), support for overlapping address space across vlans and vlan-normalization (unlike VLAN-bundle).
Disadvantages	Scaling limitations, provisioning overhead, increased control plane state.	No control over the customer broadcast domain, inefficient flooding (BUM traffic flooded to all PEs with same per EVI RT but receiving PEs may not necessarily have the intended VIDs), no support for overlapping address space across VLANs and VLAN-normalization.	Some control plane state and provisioning overhead.



(a) VLAN Based Service Interface

Figure A.5 VLAN Based Service Interface

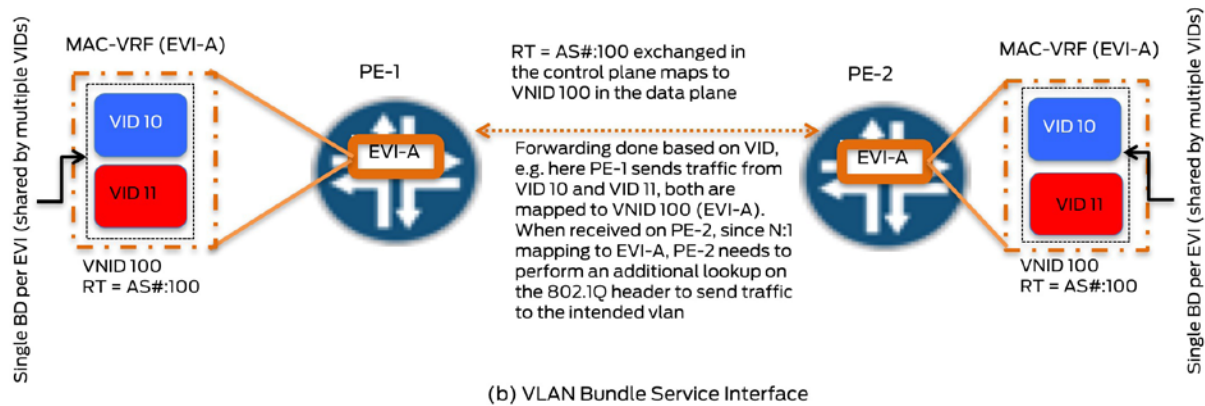


Figure A.6 VLAN Bundle Service Interface

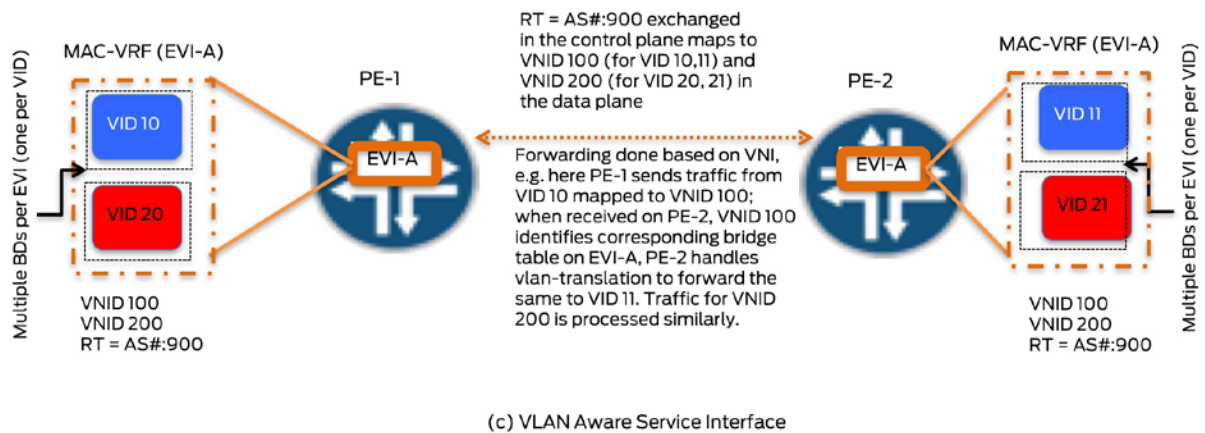


Figure A.7 VLAN Aware Service Interface

EVPN Route Types

MP-BGP (Multiprotocol extensions for BGP) enables one to carry routing information for multiple network layer protocols (for example, IPv6, L3VPN, etc.). The EVPN NLRI is carried in BGP using BGP Multiprotocol Extensions with an Address Family Identifier (AFI) of 25 (L2VPN) and a Subsequent Address Family Identifier (SAFI) of 70 (EVPN). In order for two BGP speakers to exchange EVPN NLRI, they must use BGP Capabilities Advertisements to ensure that they both are capable of properly processing such NLRI. This is done by using capability code 1 (multiprotocol BGP) with an AFI of 25 (L2VPN) and a SAFI of 70 (EVPN).

Similar to L3VPN, the EVPN address family uses Route Distinguishers (RDs) and Route Target (RTs). RDs (unique 8B number prepended to each route) allow for overlapping address space in different VRF instances by allowing identical routes in different VRFs to be treated as unique and processed as such by the BGP path selection algorithm. PEs participating in the same VRF often use different RDs, which helps in troubleshooting as the originating PE can be immediately identified by looking at the RD. It also allows for better load balancing wherein a route reflector can advertise all VPN prefixes as they are unique (same network but different RDs). RTs (extended community) are used to define VPN membership which dictates that route information is shared among interested VRFs. Both RDs and RTs can either be manually configured or automatically generated.

Table A.2 highlights the different EVPN NLRI route types, usage, and format when VXLAN is used as the data plane encapsulation. (Please refer to RFC 7432 for details on EVPN/MPLS.)

Table A.2 Various EVPN NLRI Routes

Route Types and Extended Communities	Format	Functionality and Usage	Benefit
Type 1 - Ethernet Auto-Discovery (A-D) Route per ES	Route Distinguisher (8B) Route = Not EVI specific (PE lo0 IP:#) Ethernet Segment ID (10B) = ESI value of multihomed PEs Ethernet Tag ID (4B) = MAX-ET {0xFFFFFFFF} MPLS label (3B) = 0 Route Target (8B) = Of all EVIs for this ESI BGP encapsulation (8B) = VXLAN encapsulation ESI Label (8B) • Flag = 0 (Single-Active), 1 (Active-Active) • ESI Label = null	Mandatory route advertised by both multi-homed PEs for a given ES. Upon a failure in connectivity to the attached segment, the PE withdraws the corresponding Ethernet A-D per ES routes. Since this route carries RTs of all EVIs for which this ES belongs, its withdrawal is reflected in all relevant EVI tables. This triggers all PEs that receive the withdrawal to update their next-hop for all MAC addresses associated with that ES. Flag field of the ESI label extended community is used to convey the multi-homing mode. If single-homed, ESI value is 0. Unlike MPLS, ESI label is not used for split-horizon filtering with VXLAN as the data plane encapsulation.	Fast Convergence MAC Mass Withdraw Multi-homing mode advertisement
Type 1 - Ethernet Auto-Discovery (A-D) Route per EVI	Route Distinguisher (8B) = EVI specific (PE lo0 IP:#) Ethernet Segment ID (10B) = ESI value of multihomed PEs Ethernet Tag ID (4B) = 0 (without VID translation) MPLS label (3B) = 0 Route Target (8B) = Per EVI BGP encapsulation (8B) = VXLAN encapsulation	In an all-active multi-homed scenario this route is used to implement the EVPN aliasing or load balancing feature. Remote PEs which receive MAC advertisement routes for a given ESI should consider the MAC address as reachable via all PEs that advertise reachability to the relevant ES using Ethernet A-D per EVI routes. The association of MAC routes, and the corresponding aliasing route, is determined by the same RD and RT. In single-active multi-homed mode this route is used to implement a similar backup-path feature. In this case, a remote PE sends traffic to the multi-homed PE that is the DF and installs a backup forwarding entry pointing to the non-DF PE. Unlike MPLS, Aliasing label is not used with VXLAN as the data plane encapsulation.	Aliasing Efficient load-balancing

Route Types and Extended Communities	Format	Functionality and Usage	Benefit
Type 2 - MAC/IP Advertisement Route	Route Distinguisher (8B) = EVI specific (PE lo0 IP:#) Ethernet Segment ID (10B) = ESI value of multihomed PEs Ethernet Tag ID (4B) = VNID (to identify BD) MAC Address Length (1B) = 48 (value in bits) MAC Address (6B) = MAC address of end-host IP Address Length (1B) = 32 or 128 (value in bits) IP Address (0, 4 or 16B) = IP address of end host	<p>An EVPN PE advertises a MAC route as soon as it learns a MAC address from an access interface, propagating MAC address reachability (VNI RTs to confine propagation). If this EVPN PE has an L3 interface for the VLAN in which this MAC address is learned, then it is capable of learning the host IP address (for example ARP snooping), and advertises this MAC/IP binding by generating another Type 2 route that contains both the MAC and IP addresses. MAC route conveys L2 information and is installed in the EVI BD table while the MAC/IP route conveys ARP like mappings (L3) present in the IP VRF ARP table (host IP route in the IP VRF route table).</p>	Advertise reachability to Unicast MAC Addresses (MAC/IP bindings)
	MPLS Label1(3B) = L2 VNI (VLAN) MPLS Label2(3B) = L3 VNI (VRF)	<p>Default gateway extended community is added to the MAC/IP advertisement of the default gateway (e.g. IRB MAC/IP route), to allow for default gateway synchronization, as explained in the previous section. Receiving PEs on processing type 2 routes with this community will attempt to route traffic on behalf of the advertising PEs, which prevents egress traffic tromboning.</p>	Default gateway synchronization
	Route Target (8B) = per EVI/VNI BGP encapsulation (8B) = VXLAN encapsulation Default Gateway (8B) = 0x0d (Sub-Type= Default GW) MAC Mobility (8B) <ul style="list-style-type: none"> • Flag = 1 (Sticky MAC can not move), 0 (can move) • Sequence Number 	<p>MAC mobility extended community prevents traffic black holing on MAC moves wherein the old PE has not yet withdrawn its route and the host has moved to a new segment. When the new PE learns a local MAC address for which it had previously received a Type 2 route with a different ESI, it will advertise the host MAC address by generating a Type 2 route tagged with the MAC mobility extended community. This acts as a trigger for the old PE to withdraw its route. The sequence number field increases with each MAC move and ensures that the most recent route version is processed. Based on a timer, MAC move occurrences can be accounted, to prevent the sequence number incrementing to infinity due to incessant MAC flapping and corrective action can be taken once this limit is reached. Spoofed MAC addresses can be detected using the flag field, as in this case Type 2 routes with the MAC mobility community will be received for MACs identified as static/sticky, which should not move.</p>	MAC mobility

Route Types and Extended Communities	Format	Functionality and Usage	Benefit
Type 3 - Inclusive Multicast Ethernet Tag Route	<p>Route Distinguisher (8B) = EVI specific (PE lo0 IP:#)</p> <p>Ethernet Tag ID (4B) = VNID</p> <p>IP Address Length (1B) = 32 or 128 (value in bits)</p> <p>Originating Router's IP Address (4 or 16B) = PE's lo0 IP address</p> <p>PMSI Tunnel Attribute</p> <ul style="list-style-type: none"> • Flags = 0 (No Leaf information required) • Tunnel Type = 3-6 (6 = Ingress Replication) • MPLS Label = VNID • Tunnel Identifier = Sender PE's lo0 IP address <p>Route Target (8B) = per EVI/VNI</p> <p>BGP encapsulation (8B) = VXLAN encapsulation</p>	<p>Type 3 routes are used to discover the multicast tunnels among the endpoints associated with a given VNI in an EVI (VNI RTs to confine propagation). They allow PEs to send Broadcast, Unknown Unicast, and Multicast (BUM) traffic from a CE for a VLAN in a given EVI, to all other PEs for that contain the given VLAN in that EVI. The PMSI Tunnel Attribute (MVPN defined construct) is used to encode the type of multicast tunnel to be used as well as the multicast tunnel identifier. Using Ingress Replication (current implementation) to flood a BUM frame ingress PE sends a unicast replica to each of the egress PEs individually over their respective unicast VXLAN tunnels. With IR, there is minimal operational complexity involved and the core will not need to maintain any states for the multicast trees. JUNOS platforms have inherent support for improved multicast throughput optimization algorithms with binary tree replication logic that provides for distributed replication load across PFEs. This can alleviate the processing overhead associated with IR at the ingress PE.</p>	Set up path for BUM traffic

Route Types and Extended Communities	Format	Functionality and Usage	Benefit
Type 4 - Ethernet Segment Route	Route Distinguisher (8B) = Not EVI specific (PE lo0 IP:#)	Type 4 routes are used for automatic discovery of PEs supporting the same ESI. When a PE discovers the ESI of the attached Ethernet segment, it advertises an Ethernet Segment route with the associated ES-Import extended community attribute. As this community encodes 6B of the 9B ESI value, shared by PEs multihomed to the same segment, it is imported only by those PEs.	Ethernet Segment Discovery
	Ethernet Segment ID (10B) = ESI value of multihomed PEs	Need for Split-Horizon: For a CE multi-homed to two or more PEs (represented by an ESI) and operative in an All-Active redundancy mode, there is a possibility of BUM traffic looping. The BUM frame sent from the CE to one PE could be replicated back to the CE again by the other multi-homed PE. The filtering mechanism on the PE to prevent such loop and packet duplication is called "split horizon filtering".	Loop avoidance
	IP Address Length (1B) = 32 or 128 (value in bits)	DF role and election: The designated forwarder (DF) is needed only when a CE is multi-homed to one or more PEs and is responsible for forwarding BUM traffic towards the CE for a given ESI. After advertising its Type 4 route that indicates the ESI membership of a PE, it starts a timer (default value = 3 seconds) to allow the reception of Ethernet Segment routes from other PEs connected to the same Ethernet segment. This timer value should be the same across all PEs connected to the same ES. The DF election algorithm then constructs an ordered list in increasing numeric value of IP addresses extracted from the originating router's IP address field of the Type 4 routes. DFs are then elected for a given ES, using a module function that takes the VLAN bundle into account as well. Though the DF election algorithm locally executed, it is deterministic as the same result is computed by all PEs.	Automated DF Election for multi-homing to allow for service carving mechanisms
	Originating Router's IP Address (4 or 16B) = PE's lo0 IP address	Service Carving: A CE can be multi-homed to one or more PEs, operating in All-Active or Single-Active redundancy mode. With All-Active mode, CE connects to all PEs using a single LAG bundle, all operating in the same ESI. This allows for per-flow load balancing, since traffic for the same VLAN can be handled by all PEs in the same ES. Per-VLAN load balancing can be achieved with Single-Active mode. Here, the CE connects to all PEs using separate LAG bundles, all operating in the same ESI. The granularity of the DF election algorithm allows electing multiple DFs for a given ES on a per VLAN basis. As a result, different VLANs are handled by each PE and traffic forwarding responsibilities can be divided.	
	BGP encapsulation (8B) = VXLAN encapsulation		
	ES-Import Route Target (8B) = Encodes 6B (high-order) of the 9B ESI value (value for this community is auto-derived)		
		Split horizon rules with EVPN-VXLAN and Local Bias:	
		1. For BUM traffic from the access (from CEs): a.) If received by DF: traffic is forwarded to other local CEs (same VNI as source BUM traffic) will flood this only. It performs ingress replication and sends copies to interested PEs using VXLAN tunnels. b.) If received by non-DF: same behavior as the DF for forwarding BUM traffic from the CE. With VXLAN, non-DF, does local flooding and forwards BUM traffic received from a CE to other local CEs (same VNI). This is known as local-bias.	
		2. For BUM traffic from the core (from PEs): a.) If received by DF: Since VXLAN tunnels are used in the data plane, it is possible to have visibility into the source IP addresses of BUM frames received from associated VTEPs/PEs. Receiving PE (here DF), will then look for Type 4 routes from this source address. If any ESI is shared, then BUM traffic is not forwarded. Since the DF knows that the source of BUM traffic (for a given VNI) is locally attached to the same ES, it must have forwarded traffic to all CEs (in that VNI). Else traffic is forwarded. b.) If received by non-DF: Does not traffic BUM traffic from remote PEs to multi-homed CEs – it's not the DF for a given ES.	

Route Types and Extended Communities	Format	Functionality and Usage	Benefit
Type 5 - IP Prefix Route	Route Distinguisher (8B) = VRF RD (PE lo0 IP: #)	Optional route that is used to advertise IP prefixes to allow for inter-subnet connectivity.	L3 routing integration
	Ethernet Segment ID (10B) = 0	MAC/IPs encoded in Type 2 routes are not only used to populate MAC-VRF and overlay ARP tables, but also IP-VRF tables with the encoded host routes (/32 for IPv4 routes). In some cases, when the L2 domain across data centers does not need to be extended, EVPN may advertise IP Prefixes and therefore provide aggregation in the IP-VRF tables, as opposed to programming individual host routes. This allows for smaller RIB/FIB scale requirements. L3VPNs have addressed tenant IP prefix reachability problem statement however EVPN has the advantage of being a unified control plane providing both L2 and L3 reachability. In addition to MPLS, it can use data plane technologies like VXLAN (Ethernet over IP), which are used commonly with data center deployments. Unlike Type 2 routes, the EVPN Type 5 IP prefix route decouples the host MAC address from its IP address and thus provides a clean advertisement of an IP prefix for each bridge domain.	
	Ethernet Tag ID (4B) = 0		
	IP Prefix Length (1B) = 32 or 128 (value in bits)		
	IP Prefix (4 or 16B) = IP prefix		
	GW IP Address (4 or 16B) = 0		
	MPLS Label (4 or 16B) = L3 VNI		
	Route Target (8B) = VRF RT		
	BGP encapsulation (8B) = VXLAN encapsulation		
	Router MAC (8B) = Chassis MAC		
	Different implementation models exist for Type 5 routes. NLRI Format shown is for the Pure model.		

Building a Data Center - Routing at the Leaf Layer (Spine Devices Servicing Each POD)

This case study demonstrates manual service chaining of East-West traffic flows for different tenants/VRFs within a data center through a service block (Intra-DC Intra-POD Inter-tenant - routing at the leaf with service block connected to the spine devices).

Testbed

As illustrated in Figures A.8 and A.9, QFX10002-36Q devices are used to build the leaf and spine layers. QFX5100-48S-6Q devices are used to build the VC (two QFX5100s) and the TOR (a QFX 5100). End hosts have been simulated using IXIA tester ports. The service block is represented by the service nodes (spine devices) and an SRX-5600 device. The Junos OS software version loaded on all the devices remains the same as from previous case studies.

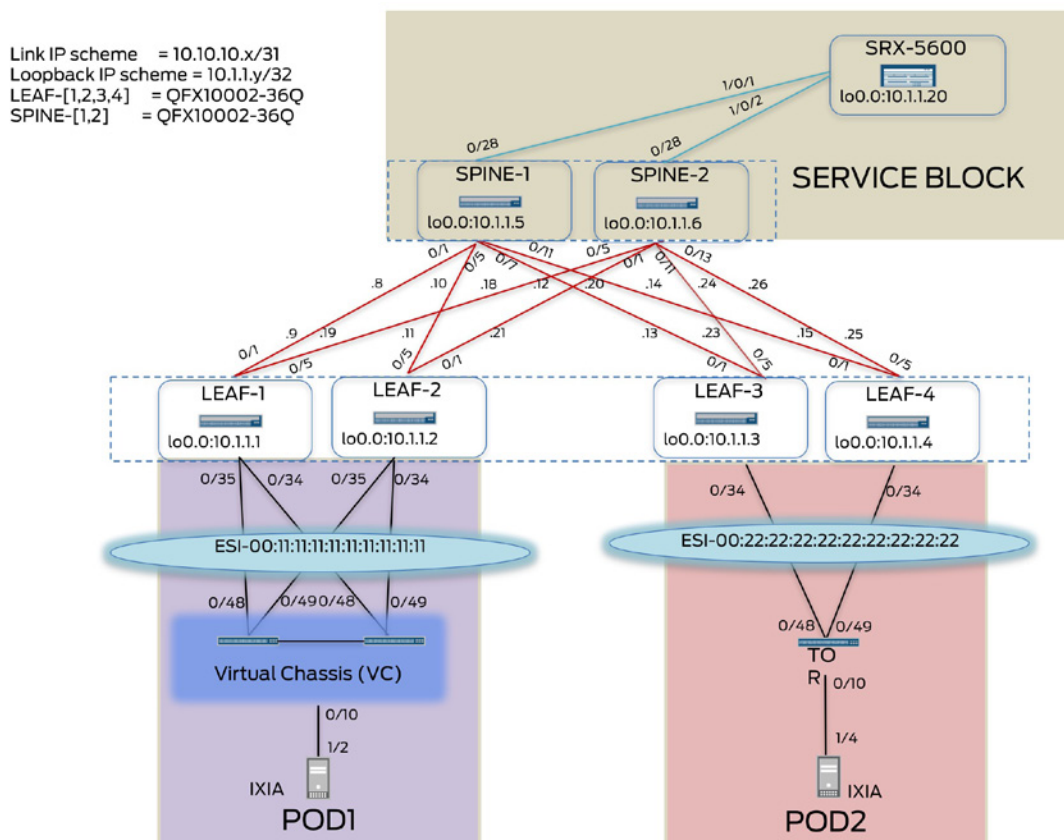


Figure A.8 Appendix Case Study – Physical Topology

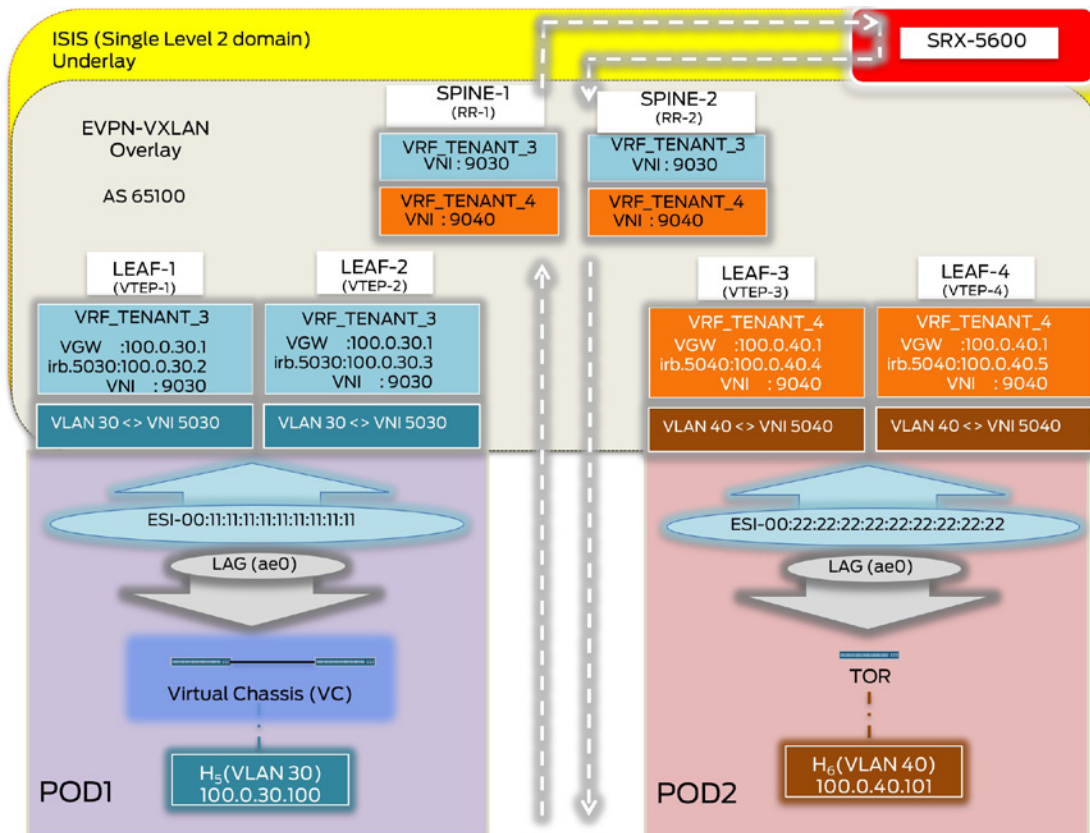


Figure A.9 Logical Topology

Design Highlights

For large data centers with thousands of servers and TOR (or Access) switches, some of them may not have the capability of maintaining or enforcing policies for inter-subnet switching. Even though policies among multiple subnets belonging to same tenant can be simpler, hosts belonging to one tenant can also send traffic to peers belonging to different tenants or security zones. In such scenarios, a L3GW may not only need to enforce policies for communication among subnets belonging to a single tenant, but also it may need to know how to handle traffic destined towards peers in different tenants.

Centralized and decentralized approaches are used simultaneously: leaf nodes switch inter-subnet traffic belonging to one tenant or one security zone locally; whereas inter-subnet traffic, belonging to two different tenants or security zones, is backhauled to centralized gateways known as service nodes (spine nodes). This traffic is further subjected to firewall processing for policy enforcement before being forwarded to the destination end-hosts.

Leaf nodes function as L2/L3 gateways (consolidated L2 and L3 gateway functionality in a single device).

VTEPs are located on each leaf node.

Spine nodes that are EVPN-aware, only participate in the exchange of Type 5 routes.

LEAF-1 and LEAF-2 host tenant 3 only (VRF_TENANT_3) while LEAF-3 and LEAF-4 host tenant 4 only (VRF_TENANT_4). Spine nodes host summary route information for both tenants. Pure EVPN Type 5 routes are used to advertise summary prefixes for different VRFs.

Manual service chaining through the SRX is achieved by leaking (redistributing) inter-vrf IP routes with physical next-hop set to the SRX (EVPN unaware).

VLAN aware EVPN service.

IP underlay is provided by a single ISIS Layer 2 domain.

The MTU has been set on all physical interfaces to account for VXLAN encapsulation.

CEs (VC and TOR) are multi-homed to the redundant pair of leaf nodes (all-active mode).

Configuration

A detailed walk-through is done here for the configuration of LEAF-1, LEAF-4, and SPINE-1 (as needed). All other devices are configured similarly. Configuration components for underlay and access remain unchanged from the previous use cases and will not be repeated here.

Overlay

Below are the configuration components required for overlay configuration.

MP-IBGP Configuration

Each spine node acts as an overlay RR (cluster-id 1.1.1.1) with all leaf nodes acting as clients, thus preventing the need for full-mesh IBGP sessions between all leaf nodes.

```
jnpr@SPINE-1> show configuration protocols bgp group IBGP-EVPN-RR
type internal;
local-address 10.1.1.5;
family evpn {
    signaling;
}
cluster 1.1.1.1; <<< Service nodes (SPINE-1 and SPINE-2) acting as RRs with all leaf devices as clients (NLRI = evpn)
local-as 65100;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 10.1.1.1;
neighbor 10.1.1.2;
neighbor 10.1.1.3;
neighbor 10.1.1.4;

jnpr@LEAF-1> show configuration protocols bgp group IBGP-EVPN
type internal;
local-address 10.1.1.1; <<< EVPN Type 5 used for IP prefix exchange (family inet-vpn not needed)
family evpn {
    signaling
}
local-as 65100;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 10.1.1.5;
neighbor 10.1.1.6;

jnpr@LEAF-4> show configuration protocols bgp group IBGP-EVPN
type internal;
local-address 10.1.1.4;
family evpn {
```

```

    signaling;
}
local-as 65100;
bfd-liveness-detection {
    minimum-interval 350;
    multiplier 3;
    session-mode automatic;
}
multipath;
neighbor 10.1.1.5;
neighbor 10.1.1.6;

```

EVI Configuration

Configuration remains the same as that from previous case studies in the book and only relevant changes are outlined here:

- *VLAN-VNI mapping:* On LEAF-1 and LEAF-2, VNI 5030 (L2 VNI) maps to VLAN 30 while on LEAF-3 and LEAF-4 VNI 5040 (L2 VNI) maps to VLAN 20. No configuration under this hierarchy (no L2 VNI) exists on the spine nodes.
- *Virtual-Switch configuration:* The VXLAN tunnel source address set to lo0.0 and global RT (target:1:100) for EVPN Type 1 routes, is configured on all leaf nodes. Manual RTs have been defined for VNI 5030 (target:1:5030 on LEAF-1 and LEAF-2) and VNI 5040 (target:1:5040 on LEAF-3 and LEAF-4). Import policy on leaf devices have been modified to accept routes for only local VNIs, therefore, LEAF-1 and LEAF-2 accept routes only for VNI 5030 not VNI 5040 and vice versa for LEAF-3 and LEAF-4. No configuration under this hierarchy exists on the spine nodes.
- *EVPN protocol configuration:* This configuration from the previous use case has been modified to include advertisement of routes for local VNIs. The vni-options statement on LEAF-1 and LEAF-2 includes vrf-target export target:1:5030 while on LEAF-3 and LEAF-4 this contains vrf-target export target:1:5040. No configuration under this hierarchy exists on the spine nodes.
- *IRB interface configuration:* IRB interface configuration has been modified to include L3 interfaces for VLANs 30 (irb.5030 on LEAF-1 and LEAF-2) and 40 (irb.5040 on LEAF-3 and LEAF-4). No configuration under this hierarchy exists on the spine nodes.
- *IP-VRF configuration:* A pure EVPN Type 5 model has been used to exchange summary routes between leaf and spine/service nodes, to allow Inter-VRF communication. All necessary routing information is provided by Type 5 routes and no Type 2 routes are exchanged (VRF-to-VRF model). IP-VRF for tenant 3 (on LEAF-1 and LEAF-2) and for tenant 4 (on LEAF-3 and LEAF-4) contains the IRB interfaces for the VLANs 30 and 40 respectively. IP-VRF for tenants 3 and 4 configured on the spine nodes does not contain any IRB interfaces, since these are only involved in Type 5 route exchange and have no local VLANs.

```

jnpr@LEAF-1> show configuration routing-instances VRF_TENANT_3
instance-type vrf;
interface irb.5030; <<< Used to advertise summary prefix, alternately an export policy can be used with
Type 5 configuration
interface lo0.30;
route-distinguisher 10.1.1.30:30;
vrf-target target:30:30;
protocols {
    evpn {
        ip-prefix-routes {
            advertise direct-nexthop; <<< Globally unique VNI used to identify L3 VRF (VRF_TENANT_3)
            encapsulation vxlan;

```

```

        vni 9030;
    }
}

jnpr@LEAF-4> show configuration routing-instances VRF_TENANT_4
instance-type vrf;
interface irb.5040; <<< Pure Type 5 (Type 5 route carries all attributes for NH - No Type 2 needed)
interface lo0.40;
route-distinguisher 10.1.4.40:40;
vrf-target target:40:40;
protocols {
    evpn {
        ip-prefix-routes {
            advertise direct-nexthop;
            encapsulation vxlan;
            vni 9040; <<< Globally unique VNI used to identify L3 VRF (VRF_TENANT_4)
        }
    }
}

```

Manual Service Chaining

For manual service chaining, the requirement is for all Inter-VRF traffic to go through the firewall. To demonstrate Inter-VRF communication between VRF_TENANT_3 and VRF_TENANT_4 on the service block spine nodes, each VRF establishes EBGp peering with the SRX. Routing policies are then used to export learned Type 5 summary prefixes from EVPN peers to the firewall, which are then learned by the desired VRF, with the physical next hop set to the SRX. This allows for data path traffic to be hair pinned through the firewall, for the desired firewall processing.

```

jnpr@SPINE-1> show configuration routing-instances | except #
VRF_TENANT_3 {
    instance-type vrf;
    interface xe-0/0/28:0.30;
    interface lo0.30;
    route-distinguisher 10.1.5.30:30;
    vrf-target target:30:30;
    protocols {
        bgp {
            group EBGp-to-SRX {
                type external;
                export ebgp-spine1-srx-vrf3; <<< EBGp peering with SRX (Redistribute EVPN T5 summary into
EBGP for NH = SRX when learned by the other VRF to hair pin data traffic via SRX)
                peer-as 65099;
                local-as 65103 loops 1;
                neighbor 20.1.1.1;
            }
        }
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop;
                encapsulation vxlan;
                vni 9030; <<< Shares VRF_TENANT_3 routing info with LEAF-1 and LEAF-2
            }
        }
    }
}
VRF_TENANT_4 {
    instance-type vrf;
    interface xe-0/0/28:0.40;
    interface lo0.40;
    route-distinguisher 10.1.5.40:40;
    vrf-target target:40:40;
    protocols {
        bgp {
            group EBGp-to-SRX {
                type external;
                export ebgp-spine1-srx-vrf4;
            }
        }
    }
}

```

```

        peer-as 65099;
        local-as 65103 loops 1;
        neighbor 20.1.1.3;
    }
}
evpn {
    ip-prefix-routes {
        advertise direct-nexthop;
        encapsulation vxlan;
        vni 9040; <<< Shares VRF_TENANT_4 routing info with LEAF-3 and LEAF-4
    }
}
jnpr@SRX-5600> show configuration protocols bgp | except #
group EBGp-to-Spine {
    type external;
    local-as 65099;
    multipath multiple-as;
    neighbor 20.1.1.0 {
        description Spine1-VRF3;
        export reject-203;
        peer-as 65103;
        as-override;
    }
    neighbor 20.1.1.2 { <<< EBGp peer sessions established over relevant logical interfaces terminating in
each VRF on the spine nodes
        description Spine1-VRF4;
        export reject-204;
        peer-as 65103;
        as-override;
    }
    neighbor 20.1.1.4 {
        description Spine2-VRF3;
        export reject-103;
        peer-as 65203;
        as-override;
    }
    neighbor 20.1.1.6 {
        description Spine2-VRF4;
        export reject-104;
        peer-as 65203;
        as-override;
    }
}

```

Verification

Leaf devices generate a EVPN Type 5 route advertising the summary prefix towards spine devices that share the same IP-VRF. LEAF-1 and LEAF-2 advertise 100.0.30/24, while LEAF-3 and LEAF-4 advertise 100.0.40/24, both received by SPINE-1 and SPINE-2.

```

jnpr@SPINE-1> show route receive-protocol bgp 10.1.1.1 table VRF_TENANT_3.evpn.0 detail
VRF_TENANT_3.evpn.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
* 5:10.1.1.30:30::0::100.0.30.0::24/304 (1 entry, 1 announced)
  Import Accepted
  Route Distinguisher: 10.1.1.30:30
  Route Label: 9030 <<< No overlay NH (Protocol NH used for resolution by remote node)
  Overlay gateway address: 0.0.0.0
  Nexthop: 10.1.1.1
  Localpref: 100
  AS path: I
  Communities: target:30:30 encapsulation0:0:0:0:vxlan router-mac:80:ac:ac:2f:00:d4

jnpr@SPINE-1> show route receive-protocol bgp 10.1.1.4 table VRF_TENANT_4.evpn.0 detail
VRF_TENANT_4.evpn.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden) <<< Chassis MAC advertised
using router-mac extended community used for inner DMAC rewrite
* 5:10.1.4.40:40::0::100.0.40.0::24/304 (1 entry, 1 announced)
  Import Accepted
  Route Distinguisher: 10.1.4.40:40
  Route Label: 9040

```

```

Overlay gateway address: 0.0.0.0
Nexthop: 10.1.1.4
Localpref: 100
AS path: I
Communities: target:40:40 encapsulation0:0:0:0:vxlan router-mac:80:ac:ac:97:15:a2

```

```
jnpr@LEAF-1> show evpn ip-prefix-database l3-context VRF_TENANT_3 direction exported extensive
```

```
L3 context: VRF_TENANT_3
```

```
IPv4->EVPN Exported Prefixes
```

```
Prefix: 100.0.30.0/24 <<< Summary routes exported as EVPN Type 5 routes by leaf nodes
```

```

EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9030
Router MAC: 80:ac:ac:2f:00:d4

```

```
jnpr@SPINE-1> show evpn ip-prefix-database l3-context VRF_TENANT_3 direction imported extensive
```

```
L3 context: VRF_TENANT_3
```

```
EVPN->IPv4 Imported Prefixes
```

```
Prefix: 100.0.30.0/24, Ethernet tag: 0 <<< Summary route (both tenants) advertised by respective leaf nodes imported as EVPN Type 5 routes by SPINE-1 (and SPINE-2).
```

```

IP route status: Created
Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 10.1.1.30:30
    VNI: 9030
    Router MAC: 80:ac:ac:2f:00:d4
    BGP nexthop address: 10.1.1.1
    Status: Active
  Route Distinguisher: 10.1.2.30:30
    VNI: 9030
    Router MAC: 80:ac:ac:2e:ca:cc
    BGP nexthop address: 10.1.1.2
    Status: Active

```

SPINE-1 and SPINE-2 redistribute the Type 5 summary route (100.0.30/24) to the SRX via BGP (routing policies). The SRX advertises this summary route back to SPINE-1 and SPINE-2 into VRF_TENANT_4.inet.0. This is further advertised over the EVPN session from the spine nodes and installed in VRF_TENANT_4.inet.0 at LEAF-3 and LEAF-4. Similarly, 100.0.40/24 is leaked into VRF_TENANT_3.inet.0 on relevant leaf and spine nodes.

```
jnpr@SRX-5600> show route 100.0.30/24
```

```

100.0.30.0/24      *[BGP/170] 00:05:11, localpref 100, from 20.1.1.0
                  AS path: 65103 I, validation-state: unverified
                  to 20.1.1.0 via xe-1/0/1.30 <<< Summary route for Tenant-3 received by SRX from
both SPINE-1 and SPINE-2
                  > to 20.1.1.4 via xe-1/0/2.30
                  [BGP/170] 00:00:48, localpref 100
                  AS path: 65203 I, validation-state: unverified
                  > to 20.1.1.4 via xe-1/0/2.30

```

```
jnpr@SPINE-1> show route 100.0.30/24 table VRF_TENANT_4.inet.0
```

```

100.0.30.0/24      *[BGP/170] 00:15:07, localpref 100
                  AS path: 65099 65099 I, validation-state: unverified
                  > to 20.1.1.3 via xe-0/0/28:0.40 <<< NH for 100.0.30/24 on spine nodes VRF_
TENANT_4 points to SRX

```

```
jnpr@SPINE-2> show route 100.0.30/24 table VRF_TENANT_4.inet.0
```

```

100.0.30.0/24      *[BGP/170] 00:11:20, localpref 100
                  AS path: 65099 65103 I, validation-state: unverified
                  > to 20.1.1.7 via xe-0/0/28:0.40

```

```
jnpr@LEAF-4> show route 100.0.30/24 table VRF_TENANT_4.inet.0 <<< Tenant-3 summary route is now available on NVEs handling Tenant-4 making Inter-VRF communication possible
```

```
100.0.30.0/24      *[EVPN/170] 00:14:15
```



```

        to 10.10.10.14 via et-0/0/1.0
    > to 10.10.10.24 via et-0/0/5.0

jnpr@LEAF-3> show route 100.0.30/24 table VRF_TENANT_4.inet.0
100.0.30.0/24      *[EVPN/170] 00:13:50
                  to 10.10.10.12 via et-0/0/1.0
                  > to 10.10.10.22 via et-0/0/5.0

jnpr@LEAF-3> show route 100.0.30/24 table VRF_TENANT_4.inet.0 extensive
*EVPN   Preference: 170
        Next hop type: Indirect, Next hop index: 0
        Address: 0xa5e7b90
Next hop type: Router, Next hop index: 1756
Next hop: 10.10.10.12 via et-0/0/1.0
Next hop type: Router, Next hop index: 1755
Next hop: 10.10.10.22 via et-0/0/5.0, selected
Session Id: 0x0
Protocol next hop: 10.1.1.5
Composite next hop: 0xc27e708 1725 INH Session ID: 0x0
VXLAN tunnel rewrite:
  MTU: 0, Flags: 0x0
  Encap table ID: 0, Decap table ID: 8
  Encap VNI: 9040, Decap VNI: 9040
  Source VTEP: 10.1.1.3, Destination VTEP: 10.1.1.5
  SMAC: 80:ac:ac:91:10:00, DMAC: 80:ac:ac:28:11:a0
Indirect next hop: 0xa64dd40 524287 INH Session ID: 0x0
Protocol next hop: 10.1.1.6
Composite next hop: 0xc27e898 1812 INH Session ID: 0x0
VXLAN tunnel rewrite:
  MTU: 0, Flags: 0x0
  Encap table ID: 0, Decap table ID: 8
  Encap VNI: 9040, Decap VNI: 9040
  Source VTEP: 10.1.1.3, Destination VTEP: 10.1.1.6
  SMAC: 80:ac:ac:91:10:00, DMAC: 80:ac:ac:7f:ba:7c

```

Traffic Flows

In Figures A.10 and A.11, Inter-VRF inter-subnet traffic flows between VRF_TENANT_3 and VRF_TENANT_4 are analogous to those explained for the symmetric IRB model, with the exception of the hair pinning of traffic through the firewall.

IXIA Statistics – Intra DC Inter-VRF Inter-Subnet (VNI 5030 <=> VNI 5040)

(H5 <=> H6 1000 flows @ 1000 pps)

Traffic Item	Tx Frames	Rx Frames ▲	Frames Delta	Loss %	Packet Loss Duration (ms)	Tx Frame Rate	Rx Frame Rate
H5 > H6	24,513	24,513	0	0.000	0.000	1,000.000	1,000.000
H5 < H6	33,871	33,871	0	0.000	0.000	1,000.000	1,000.000

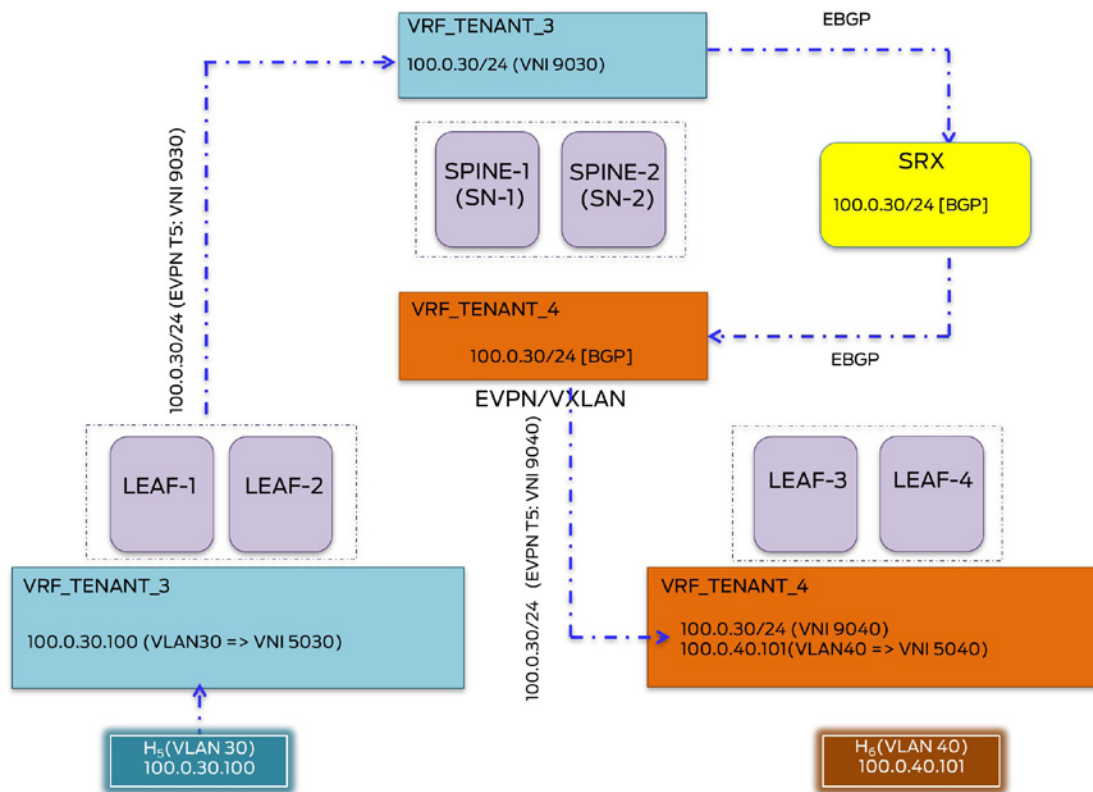


Figure A.10 Manual Service Chaining – EVPN Type 5 Route Exchange (H5 > H6)

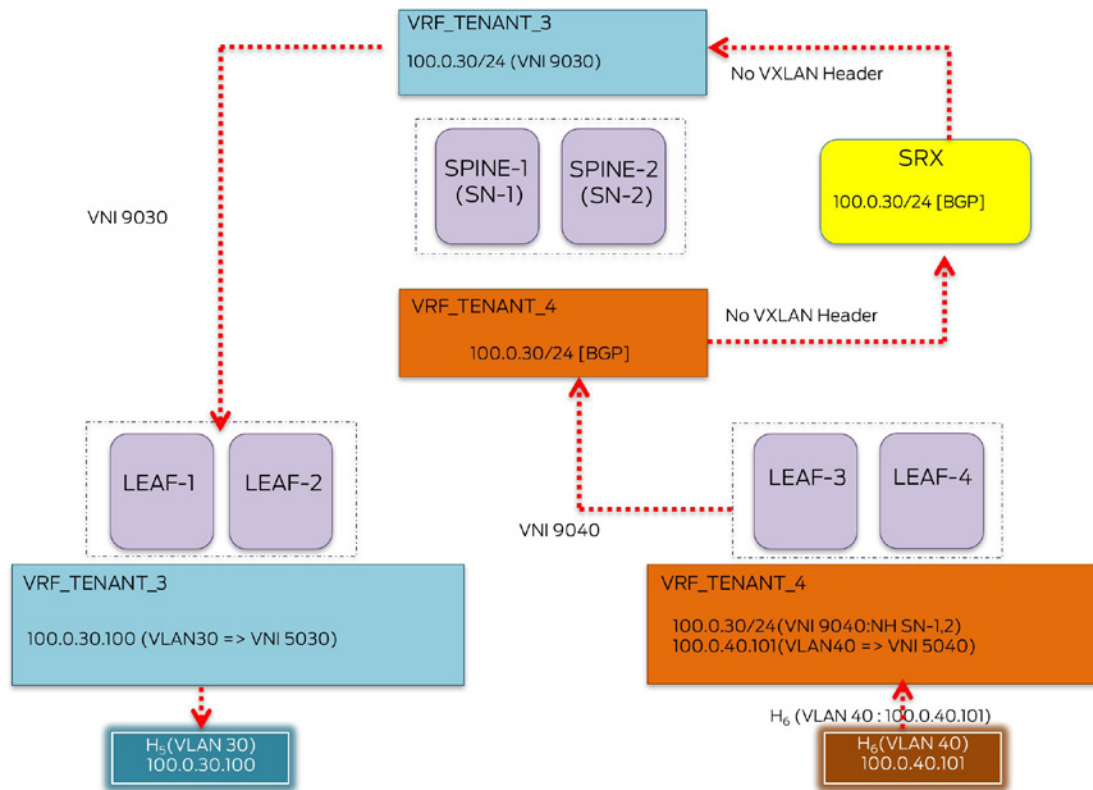


Figure A.11 Manual Service Chaining – Inter VRF Inter-subnet Traffic Forwarding (H₆ > H₅)