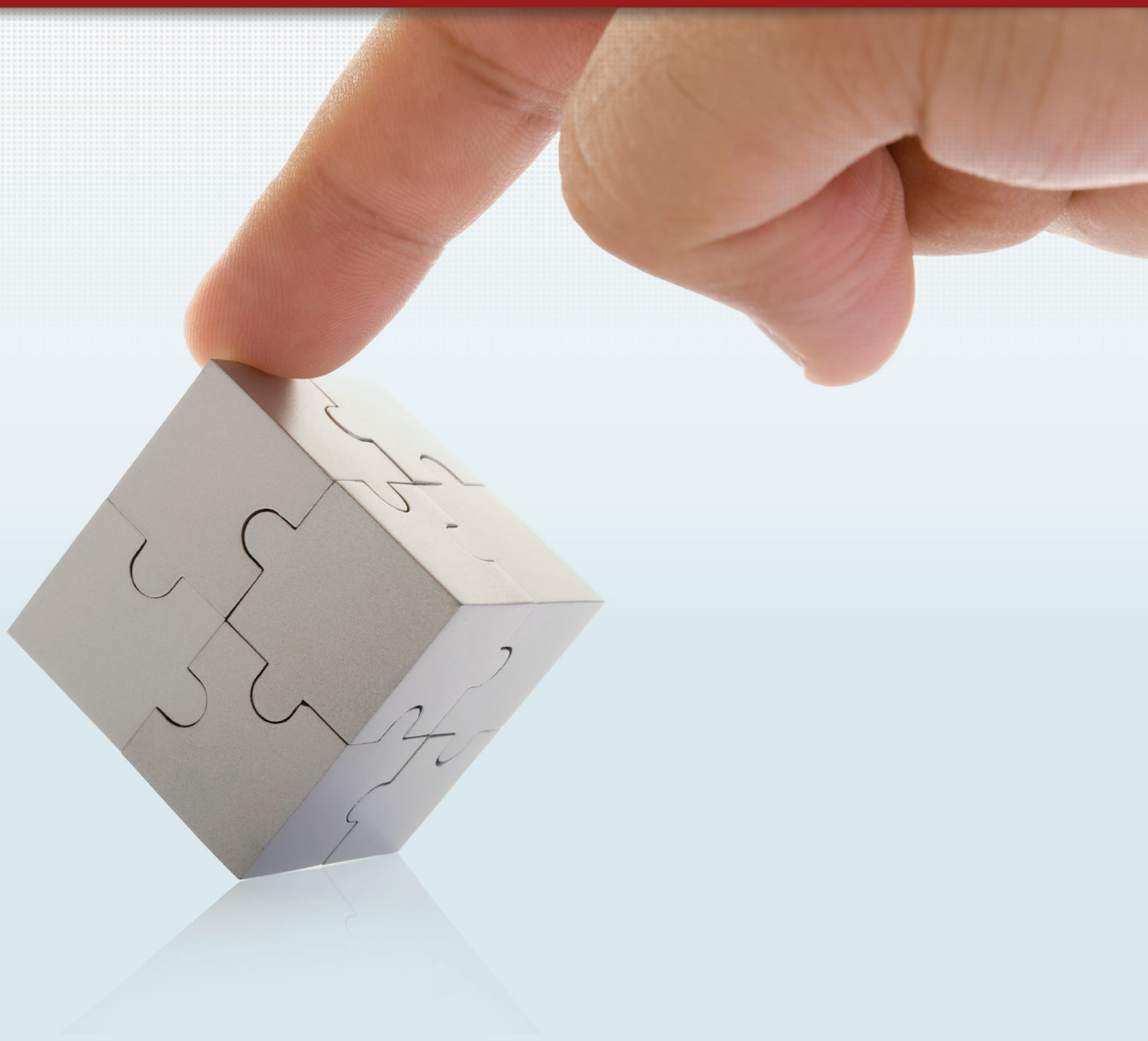
ipexpert

IPexpert's

Multiprotocol Label Switching (MPLS)

Operation and Troubleshooting



Authored By: Terry Vinson CCIE #35347 (R&S), CCNP
Technical Editor: Carl Yost Jr CCIE# 30486 (R&S), Jason Gooley CCNP
Editor: Tiffany Pagan

Before We Begin

This product is part of the IPexpert suite of materials that provide CCIE candidates and network engineers with a comprehensive training program. For information about the full solution, contact an IPexpert Training Advisor today.

Telephone: +1.810.326.1444
Email: sales@ipexpert.com

Congratulations! You now possess one of the ULTIMATE CCIE™ Lab preparation and network operation resources available today! Senior engineers, technical instructors, and authors boasting decades of internetworking experience produced this resource.

In order to enjoy technical support from IPexpert and your CCIE community, be sure to visit the following Internet resources:

http://blog.ipexpert.com
http://onlinestudylist.com

IPexpert is proud to lead the industry with multiple support options at your disposal free of charge. Our online communities have attracted a membership of over 20,000 of your peers from around the world! At blog.ipexpert.com, you can keep up to date with everything IPexpert does and read the latest in technical articles from word-renowned IPexpert instructors. At OnlineStudyList.com, you may subscribe to multiple "SPAM-free," moderated CCIE-focused email lists.

Feedback

Do you have a suggestion or other feedback regarding this book or other IPexpert products? At IPexpert, we look to you – our valued clients – for the real world, frontline evaluation that we believe is necessary so that we may always improve. Please send an email with your thoughts to feedback@ipexpert.com or call 1.866.225.8064 (international callers dial +1.810.326.1444).

In addition, for those using this book as CCIE™ preparation, when you pass the CCIE™ Lab exam, we want to hear about it! Email your CCIE™ number to success@ipexpert.com and let us know how IPexpert helped you succeed. We would like to send you a gift of thanks and congratulations.

Additional CCIE™ Preparation Material

IPexpert, Inc. is committed to developing the most effective Cisco CCIE™ R&S, Security, Voice and Wireless Lab certification preparation tools available. Our team of certified networking professionals develops the most up-to-date and comprehensive materials for networking certification, including self-paced workbooks, online Cisco hardware rental, classroom training, online (distance learning) instructor-led training, audio products, and video training materials. Unlike other certification-training providers, we employ the most experienced and accomplished teams of experts to create, maintain, and constantly update our products. At IPexpert, we are focus on making your CCIE™ Lab preparation more effective.

Issues with this Book

This book is carefully edited to ensure the accuracy of all content. Should you find any error whatsoever, please email a page reference and detailed comment to compsolv@me.com. Your email will be responded to promptly.

IPEXPERT END-USER LICENSE AGREEMENT

END USER LICENSE FOR ONE (1) PERSON ONLY

IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS,
DO NOT OPEN OR USE THE TRAINING MATERIALS.

This is a legally binding agreement between you and IPEXPERT, the “Licensor,” from whom you have licensed the IPEXPERT training materials (the “Training Materials”). By using the Training Materials, you agree to be bound by the terms of this License, except to the extent these terms have been modified by a written agreement (the “Governing Agreement”) signed by you (or the party that has licensed the Training Materials for your use) and an executive officer of Licensor. If you do not agree to the License terms, the Licensor is unwilling to license the Training Materials to you. In this event, you may not use the Training Materials, and you should promptly contact the Licensor for return instructions.

The Training Materials shall be used by only ONE (1) INDIVIDUAL who shall be the sole individual authorized to use the Training Materials throughout the term of this License.

Copyright and Proprietary Rights

The Training Materials are the property of IPEXPERT, Inc. ("IPEXPERT") and are protected by United States and International copyright laws. All copyright, trademark, and other proprietary rights in the Training Materials and in the Training Materials, text, graphics, design elements, audio, and all other materials originated by IPEXPERT at its site, in its workbooks, scenarios and courses (the "IPEXPERT Information") are reserved to IPEXPERT.

The Training Materials cannot be used by or transferred to any other person. You may not rent, lease, loan, barter, sell or time-share the Training Materials or accompanying documentation. You may not reverse engineer, decompile, or disassemble the Training Materials. You may not modify, or create derivative works based upon the Training Materials in whole or in part. You may not reproduce, store, upload, post, transmit, download or distribute in any form or by any means, electronic, mechanical, recording or otherwise any part of the Training Materials and IPEXPERT Information other than printing out or downloading portions of the text and images for your own personal, non-commercial use without the prior written permission of IPEXPERT.

You shall observe copyright and other restrictions imposed by IPEXPERT. You may not use the Training Materials or IPEXPERT Information in any manner that infringes the rights of any person or entity.

Exclusions of Warranties

THE TRAINING MATERIALS AND DOCUMENTATION ARE PROVIDED "AS IS." LICENSOR HEREBY DISCLAIMS ALL OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW THE LIMITATION OF INCIDENTAL DAMAGES OR LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU. This agreement gives you specific legal rights, and you may have other rights that vary from state to state.

Choice of Law and Jurisdiction

This Agreement shall be governed by and construed in accordance with the laws of the State of Michigan, without reference to any conflict of law principles. You agree that any litigation or other proceeding between you and Licensor in connection with the Training Materials shall be brought in the Michigan state or courts located in Port Huron, Michigan, and you consent to the jurisdiction of such courts to decide the matter. The parties agree that the United Nations Convention on Contracts for the International Sale of Goods shall not apply to this License. If any provision of this Agreement is held invalid, the remainder of this License shall continue in full force and effect.

Limitation of Claims and Liability

ANY ACTION ON ANY CLAIM AGAINST IPEXPERT MUST BE BROUGHT BY THE USER WITHIN ONE (1) YEAR FOLLOWING THE DATE THE CLAIM FIRST ACCRUED, OR SHALL BE DEEMED WAIVED. IN NO EVENT WILL THE LICENSOR'S LIABILITY UNDER, ARISING OUT OF, OR RELATING TO THIS AGREEMENT EXCEED THE AMOUNT PAID TO LICENSOR FOR THE TRAINING MATERIALS. LICENSOR SHALL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, REGARDLESS OF WHETHER LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. WITHOUT LIMITING THE FOREGOING, LICENSOR WILL NOT BE LIABLE FOR LOST PROFITS, LOSS OF DATA, OR COSTS OF COVER.

Entire Agreement

This is the entire agreement between the parties and may not be modified except in writing signed by both parties.

U.S. Government - Restricted Rights

The Training Materials and accompanying documentation are "commercial computer Training Materials" and "commercial computer Training Materials documentation," respectively, pursuant to DFAR Section 227.7202 and FAR Section 12.212, as applicable. Any use, modification, reproduction release, performance, display, or disclosure of the Training Materials and accompanying documentation by the U.S. Government shall be governed solely by the terms of this Agreement and shall be prohibited except to the extent expressly permitted by the terms of this Agreement.

IF YOU DO NOT AGREE WITH THE ABOVE TERMS AND CONDITIONS, DO NOT OPEN OR USE THE TRAINING MATERIALS AND CONTACT LICENSOR FOR INSTRUCTIONS ON RETURN OF THE TRAINING MATERIAL

Contents

Chapter 1: Overview of MPLS Operation and Troubleshooting

Chapter 1: Introduction to MPLS Operation and Troubleshooting introduces the team of authors, consultants, and editors that completed this book and describes the book's purpose. This chapter also provides suggestions for the usage of this written work. This introductory chapter also covers a basic overview of MPLS operation and troubleshooting concerns and the basic protocols that affect its operation.

About the Author

Terry Vinson, CCIE No. 35457 (R&S), Terry Vinson is a highly experienced training consultant, specializing in documentation development, validation, verification and communications. For the last 10 years, Terry has worked in the private sector as a Senior Technology Consultant and Trainer for several consulting firms in Washington DC, Northern and Central Virginia. In this capacity, he has provided services to Major Metropolitan Health Systems, the Mexican Embassy, and the Executive Office of the President of the United States of America (EOP).

About the Technical Editors

Carl Yost Jr., CCIE No. 30486 (R&S), currently works as a Network Engineer/Director of I.T. for a health care company in Buffalo NY. He has worked in numerous roles in I.T. since 1998. Carl is currently preparing for the CCIE in Security while living with his wife and children in Western New York. When not surrounded by Cisco devices, Carl truly enjoys working with Redhat Linux.

Jason Gooley, CCNP, Jason is a highly motivated network engineer with over 17 years of experience in the communications industry. Based in Chicago, Jason currently manages the network for the nation's most famous next day carpet company. Jason is currently in the process of pursuing his CCIE certification for Routing and Switching while also expanding his knowledge in Unified Communications and Security.

About the Technical Consultants

Scott Morris, CCIEx4, CCDE, JNCIEx2, CISSP, Scott has been one of the most well-known figures in the IT industry for over 25 years. He has fulfilled a number of important roles within both the public and private sectors. As a Certified Cisco Systems Instructor (CCSI) and Juniper Networks Certified Instructor (JNCI), Scott has provided world-renowned CCIE training since 2002. He has delivered courses to a wide variety of audiences including internal training at Cisco Systems.

Vikram Malhi, CCIE #13890 (Voice), CCVP, Cisco IP Telephony Support Specialist, Cisco IP Telephony Operations Specialist, Cisco IP Telephony Design Specialist, Cisco Wireless LAN Design Specialist, with over 14 years of IP Telephony training and consulting experience and a wealth of technical certifications, Vik Malhi has proven that he is one of the top Cisco CCIE Voice Instructors and Consultants in the world! Vik was the first engineer to install CM 3.0 in Europe, has years of AVVID consulting and implementation experience and has been teaching and developing CCIE Voice Lab courses and self-study learning materials for over 4 years. Vik is responsible for updating, supporting and teaching IPexpert's Voice-related products, services and courses. Over the past 4 years Vik has been the cornerstone of IPexpert's world-renowned CCIE Voice Lab product development and training and has assisted more CCIE Voice engineers in passing the lab than any other Instructor, worldwide!

Marko Milivojevic, CCIE #18427 (R&S SP), CCNP, CCDP, CCIP, Marko, a dual CCIE who has recently passed the CCIE R&S 4.0 lab, has spent 12 years designing and supporting some of Europe's largest service provider networks. He is accredited with designing the largest multi-service internet infrastructure in Iceland. He has been working with IPexpert over the past few years developing several self-study products, and will now be more-heavily involved in product development, product support and classroom training (throughout Europe, Australia and Asia) on full time basis.

About the Editor

Tiffany Pagan began her career in editing in 1997. Throughout her career, she has worked with several private individuals and companies such as Moffitt Cancer Center and Tampa General Hospital. Tiffany is currently working on writing her own series of short stories as well as working as an editor and personal assistant. Tiffany resides in Tampa, Florida with her husband and three beautiful children.

Who Should Read this Book?

This text has two primary audiences. The first audience is for those CCIE candidates that are searching for the most comprehensive and error-free materials available for the operation and troubleshooting of key technologies presented in the various tracks of the CCIE written and practical lab exams. These students should possess a home rack of equipment for CCIE-level command-line practice, they should possess an equipment emulator, or they should rent equipment from a company like www.proctorlabs.com. The authors and technical editors exhaustively tested all of the demonstrations found throughout the text and the important end of chapter Trouble Ticket challenges against all practice rack options described earlier. Where issues arise with popular equipment emulators, the text makes note. This book is the most remarkably thorough and technically accurate book written on the subject of MPLS to date.

The book's second audience is those readers that must support MPLS in their actual network environments. This book serves as an amazing guide and reference for real-world problem solving within production networks that deploy these specific technologies. In fact, while many courses and texts purport to have certification success as a by-product of a thorough investigation of all protocols, this book actually succeeds in this approach.

How to Use this Book

This book breaks specific MPLS technologies down on a chapter-by-chapter basis for a complete and thorough review of this broad set of topics. Each chapter begins with a review of the selected technology. Following this, the text provides an intense examination of the operation of the protocols, including key aspects of troubleshooting for the specific technology. After this, the chapter presents some of the most common issues that can result with a particular technology, and most importantly, details the simple troubleshooting tools and steps that succeed for remediation.

Each chapter concludes with sample troubleshooting scenarios that provide a full walkthrough of a well-designed approach for troubleshooting each major issue. The text provides reference guides for the most popular and powerful **show** and **debug** commands for a specific technology.

Some chapters also contain sample Trouble Tickets on specific technologies. Readers may download initial configurations for these sample Trouble Tickets, or install them in a simple Graphical User Interface (GUI) on www.proctorlabs.com. These sample Trouble Tickets allow students to build confidence and expertise by actually troubleshooting issues in the MPLS domain presented in the chapter.

Students are encouraged to follow along on a rack of equipment for every section of every chapter. This really enhances and strengthens the learning process.

An Introduction to MPLS

The sensible place to start when working with a given technology is to begin with a basic if not broad explanation of what it is. For the purpose of our discussion we will define MPLS as an extremely scalable transport protocol that places no significance on the protocol (payload) it is transporting. MPLS forwards packets based solely on the label that has been assigned to that packet. This means that MPLS does not need to look at the packet itself, and as such MPLS affords us the opportunity to build an end-to-end delivery schema across virtually any transport medium regardless of the protocol being run. We have to step back and think about this capability in a broader picture to really understand what this means.

MPLS allows us to eliminate the traditional dependence on the data link layer of the OSI model. Protocols that exist at that layer like ATM, frame-relay, or even Ethernet are really no longer important. Using MPLS we can now easily transport Frame Packets across a routed FastEthernet network just as simply as we did via traditional serial links, as a point of fact most if not all frame-relay services currently running in the world, what few are left, are probably relying on MPLS in the “cloud” to operate. The magnitude of this capability can best be grasped when we consider that with MPLS multiple types of traffic can now be transported without consideration or requirement for multiple Layer-2 network types.

MPLS works by making modifications to the traditional header format used by an ip packet. Where traditionally there existed only a data link layer header and a network layer header MPLS inserts a new header between them. This is why MPLS is colloquially referred to as a “layer 2.5 protocol”. This header that now exists between Layer 2 and Layer 3 affords the transparent payload carrying service for either circuit based networks and packet-switched networks that we have been discussing.

This idea is not a new one. The same capabilities though not as flexible exit in Asynchronous Transfer Mode (ATM) protocol, but MPLS has expanded and grown such that it can be employed to rely on the strengths of ATM while accommodating and making up for that protocols weaknesses or actually replace on it own. ATM is outside the scope of our discussion and we have mentioned it merely as a historical reference, but the important thing to note is that ATM supports connection-oriented services for variable-length frames. MPLS also supports variable-length frames with connection-oriented delivery, but it does so with considerably less operational overhead. This means that MPLS is being embraced by more and more service providers daily and as such it has become very commonplace in networking environments.

In this section we will begin with an in depth look at Cisco Express Forwarding (CEF) and how it enables the operation of MPLS. Once we understand what functionality CEF brings to MPLS we will look closely at the concepts of load balancing and virtual routing and forwarding (VRF) instances. Lastly we will look at possibly the most valuable component we need to understand in order to accurately and quickly troubleshoot MPLS: the formation of its Control and Forwarding Planes. There will be a lot of new

terminology, as well as show and debug commands introduced in this chapter which will need to be committed to memory in order to better understand upcoming chapters.

Cisco Express Forwarding (CEF)

It is necessary for us to understand the relation between MPLS and CEF. We personally feel that this single association will make considerable impact on a student's understanding of how first to configure MPLS as well as how to methodically isolate any faults that may be encountered in its operation. CEF switching is an adaptation made on Cisco IOS devices that allows very rapid forwarding of packets. The process was designed to involve four independent lookups to find forwarding information for a given ip address. Remember that the routing information base (RIB) of a router is not organized in any logical fashion. CEF is method used to optimize the recursive lookup process, and it is composed of two primary structures.

First we have the Forwarding Information Base (FIB). The FIB is will contain a set of entries for each address found in the ip routing table. Any changes to the routing table of the device will be directly reflected in the FIB. This maintains a one-to-one relationship between the two tables. Next hop information is copied from the RIB and placed in the FIB. This means that the FIB will have a record of all known prefixes and their next hops thus completely eliminating the need for maintaining a route cache like that found in traditional process and fast switching technologies.

Second, we have the adjacency table which is the database of the layer 2 address of all nodes that are one hop away from the local device via any link layer connection. This information is important because CEF will prepend this layer 2 information into outbound packets. This process significantly increases the efficiency of the protocol by eliminating the need for an address resolution protocol request for each table lookup. The adjacency table brings with it a number of new terms that we will need to understand, and those terms specifically are associated with particular adjacency states:

- **NULL**
  - Handles packets destined to a NULL interface, packets so address will be dropped normally by default.
- **GLEAN**
  - Destination is attached via a broadcast network but the MAC is unknown. These packets will always generate an ARP request in an effort to learn the missing MAC information.
- **PUNT**
  - If CEF is not supported or special handling instructions for a destination path, these packets are normally switched via the next-slower switching method (typically fast switching).
- **DROP**
  - Cannot be CEF switched at all. Packets are dropped BUT the prefix is checked
- **DISCARD**
  - These packets will always be discarded.
- **CACHE**
  - This type of entry contains the correct outbound interface and the correct MAC address for its FIB entry. The MAC address is the MAC address if the destination's subnet that is directly connected to the router, or it is the MAC address of the router that the packet

needs to be sent to if the destination's subnet is not directly connected to the router currently processing the packet.

- **RECEIVE**
  - This type of entry handles packets whose final destinations include the router itself. This includes packets whose IP addresses are assigned to the router itself, broadcast packets, and multicasts that have set up the router itself as one of the destinations.

The important thing to remember when dealing with CEF is that route cache building is not triggered by the first packet to arrive, but instead it is done for all entries in the routing table. Thus all changes that take place in the RIB are automatically reflected in the FIB.

We are discussing CEF at this point because it is a requirement for MPLS to function. CEF can be enabled globally by using the ***ip cef [distributed]*** or under an interface with the ***ip route-cache cef*** commands. Keep in mind that interface level configurations will override global configuration settings we regard to CEF. Additionally, we need to recognize what determines whether or not a packet will be CEF switched. Cisco IOS will switch a packet using CEF **ONLY** if CEF is enabled on the inbound interface. This means that the setting for any given outbound interface is irrelevant in the switching method determination.

We need to explore these new components and their adjacency states via the command line interface. It is very important that we come away from this discussion with a concrete understand of everything we have mentioned in order to move efficiently into the inner workings and operations of MPLS. To see these components and to observer the associated behaviors we will begin with the network topology outlined in Figure 1-1.

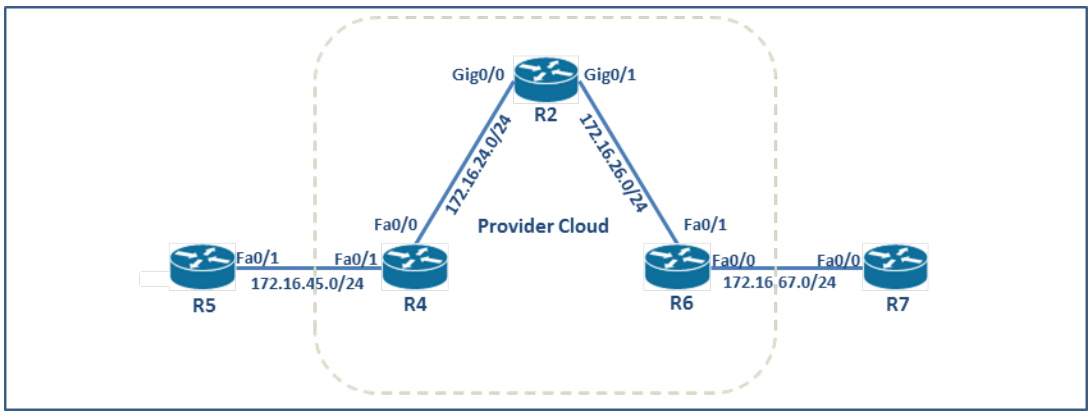


Figure 1-1: Provider Cloud

In this topology we are running OSPF area 0 on R4, R2 and R6. At this time R5 and R7 are not part of our discussion. We will begin by observing the output on our "cloud devices" and verifying that we have full reachability.

```
R4#ping 192.1.2.2 source loopback0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.2.2, timeout is 2 seconds:
Packet sent with a source address of 192.1.4.4
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

```
R4#ping 192.1.6.6 source loopback0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
Packet sent with a source address of 192.1.4.4
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

This tells us that our topology is configure and our OSPF process is running as we would expect. Now we will begin to look closer at CEF. First we will look at the output of the **show ip cef** command as it is executed on R2:

```
R2#show ip cef
Prefix              Next Hop            Interface
0.0.0.0/0           no route
0.0.0.0/8           drop
0.0.0.0/32          receive
127.0.0.0/8         drop
172.16.24.0/24      attached            GigabitEthernet0/0
172.16.24.0/32      receive             GigabitEthernet0/0
172.16.24.2/32      receive             GigabitEthernet0/0
172.16.24.4/32      attached            GigabitEthernet0/0
172.16.24.255/32    receive             GigabitEthernet0/0
172.16.26.0/24      attached            GigabitEthernet0/1
172.16.26.0/32      receive             GigabitEthernet0/1
172.16.26.2/32      receive             GigabitEthernet0/1
172.16.26.6/32      attached            GigabitEthernet0/1
172.16.26.255/32    receive             GigabitEthernet0/1
172.16.45.0/24      172.16.24.4         GigabitEthernet0/0
172.16.67.0/24      172.16.26.6         GigabitEthernet0/1
192.1.2.0/24        attached            Loopback0
192.1.2.0/32        receive             Loopback0
192.1.2.2/32        receive             Loopback0
192.1.2.255/32      receive             Loopback0
192.1.4.4/32        172.16.24.4         GigabitEthernet0/0
192.1.6.6/32        172.16.26.6         GigabitEthernet0/1
224.0.0.0/4         drop
224.0.0.0/24        receive
240.0.0.0/4         drop
255.255.255.255/32  receive
```

At this time be sure to observe the “Next Hop” column and note the values provided there. Compare these with the bulleted adjacency states provided previously. This begs the question, “what would the output look like if CEF was disabled?”

First we will disable CEF.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#no ip cef
R2(config)#exit
```

Then run the **show ip cef** command:


```
R2#show ip cef
%IPv4 CEF not running
R2#
```

The router very clearly notifies us that IPv4 CEF is not running. Now that we have seen this we will re-enable the process and then take a closer look at the adjacency table.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip cef
R2(config)#exit

Now we need to look at the adjacency table

R2#show adjacency
Protocol Interface                 Address
IP       GigabitEthernet0/0        172.16.24.4(10)
IP       GigabitEthernet0/1        172.16.26.6(10)
```

This version of the command only provides the briefest of details. To see the Layer 2 header information we mentioned previously it will be necessary to add the **detail** keyword to the end of the command.

```
R2#show adjacency detail
Protocol Interface                 Address
IP       GigabitEthernet0/0        172.16.24.4(10)
                                   10 packets, 1140 bytes
                                   epoch 0
                                   sourced in sev-epoch 0
                                   Encap length 14
                                   000AB819C92000179527BA000800
                                   ARP
IP       GigabitEthernet0/1        172.16.26.6(10)
                                   10 packets, 1140 bytes
                                   epoch 0
                                   sourced in sev-epoch 0
                                   Encap length 14
                                   000AB82DCB3100179527BA010800
                                   ARP
```

Now we can clearly see the information we have been discussing. The entry of “ARP” means that the entries where learned via address resolution protocol. The value in the parenthesis indicates the number of time the FIB has pointed to this adjacency entry. In the line above the ARP field in the list we see the layer 2 header information. The very first 12 characters are the mac address of the next hop, and the next 12 characters are the MAC address of the source interface. Finally , the last four characters are the standardized Ethertype values used by all vendors.

Please be aware of the fact that the mac address in this layer two header information must correspond with the address learned by the ARP process. If this is not the case there will be an inconsistent state that will prevent CEF from working properly, and this will directly impact MPLS operation. We can see that in both instances on R2 this condition is not taking place.


```
R2#show arp 172.16.24.4
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.24.4            31   000a.b819.c920  ARPA   GigabitEthernet0/0

R2#show arp 172.16.26.6
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.26.6            41   000a.b82d.cb31  ARPA   GigabitEthernet0/1
```

We clearly see that the values under the "Hardware Addr" field match the first twelve digits in the Layer 2 header information in the previous show command.

Load Balancing

Though not as important as understanding the value of CEF to the MPLS process, CEF load balancing will ultimately prove itself important in instances where multiple Label Switched Paths are going to be available. The default behavior for CEF load balancing is per-destination, that is to say load balancing will be done on a per flow basis. Keep in mind that flow is typically and best described as traffic associated with a specific source and destination address and port pair. This source address, port, destination address and port is commonly referred to as a quadrant. In ***Chapter 2: MPLS Labels*** we will look at how load balancing by destination will result in labels being assigned to certain next hops, and how all prefixes using that next hop will follow the same label switched path.

To verify the status of load balancing on a given device it is necessary to use the **show ip cef exact-route <src> <dst>** command:

```
R4#show ip cef exact-route 192.1.4.4 192.1.6.6
192.1.4.4 -> 192.1.6.6 => IP adj out of FastEthernet0/0, addr 172.16.24.2
```

What if we want to verify the default behavior of per-destination load balancing?

```
R2#show cef state
CEF Status:
 RP instance
 common CEF enabled
IPv4 CEF Status:
 CEF enabled/running
 dCEF disabled/not running
 CEF switching enabled/running
 universal per-destination load sharing algorithm, id BDB883A2
IPv6 CEF Status:
 CEF disabled/not running
 dCEF disabled/not running
 universal per-destination load sharing algorithm, id BDB883A2
```

In the book **Multicast Operation and Troubleshooting** we introduced the concept of a hashing algorithm. This hash was used to load balance between more than one Rendezvous Point. The hashing computation was an approximate process, meaning that there was a best effort to equally distribute between the multiple RPs. CEF load balancing is very similar to this process. In a nutshell, the load sharing algorithm employed by CEF supports up to sixteen hash "buckets" a bucket is a logical construct that supports the notion of a single flow. The fact that CEF only supports 16 buckets means that it can

only support 16 simultaneous flows, any more than that will result in a hash collision. In the event of a hash collision some buckets will contain multiple flows. Many people use the term load sharing to describe this process because of this best-effort-approach.

Virtual Routing and Forwarding (VRF)

So far we have discussed the enabling details for MPLS and even though CEF is required for the successful operation of the protocol, VRFs are the first component that is directly related to MPLS and Layer 3 VPN across MPLS networks. We discussed CEF, and how CEF support the efficient transfer of ip packets. Now we take the concept one step further.

Virtual routing and forwarding is a technology that allows multiple instances of a routing table to exist in a router and work simultaneously. This increases overall network performance by allowing network paths to be segmented without using multiple devices. This segregation of network traffic increases network security and can eliminate the need for encryption and authentication in certain circumstances. Internet service providers (ISPs) more often than not will leverage VRF to create separate virtual private networks (VPNs) for customers; thus the technology is also referred to as VPN routing and forwarding. We will cover this concept in substantial detail in ***Chapter 4: MPLS VPN***. Until then we will speak of VRFs in generic terms.

VRF instances create a concept that can best be described as a logical router, but while a logical router may include many routing tables, a VRF instance uses only a single routing table. In addition, a VRF requires a forwarding table that defines the next hop for each data packet, this equates to a list of devices that may be called upon to forward discreet packets, and a set of rules and routing protocols that govern how the packet will be forwarded. These tables prevent traffic from being forwarded outside a specific VRF path and also serves to block traffic that should remain outside the VRF path.

These VRF paths rapidly became the backbone of most virtual systems; linking the servers and storage necessary to provide services into a single highly reliable fabric. The issue is that virtual networks -- and therefore the virtual systems they compose – are often limited to one facility. The concept of virtual routing and forwarding is the primary solution that enables these resources and control mechanisms to be spread across multiple locations.

This means that VRF-capable routers can and are used to subdivide a virtual network by dividing a router into multiple independent virtual devices were each virtual router supports a single virtual network. Networks that support standard routing protocols such as OSPF or BGP. These routing protocols operate on each virtual router independently of the routing processes running on the other virtual routers configured on the same physical device. Each virtual router maintains a separate set of routing and forwarding tables thus eliminating the need for all of the virtual routers to support the same set of routing protocols.

This separation of processes means that functions such as Network Address Translation (NAT) and firewall must operate independently for each virtual network. NAT and firewall functions in VRF-equipped routers operate within each virtual router instance. As a direct result, each virtual network can

have its own firewall policies and maintain a separate IP address space. Additionally, there will be single CEF adjacency table for each VRF instance.

Creating a VRF Instance

For the purpose of our discussion we will introduce the concept of a VRF and we will create one via the most streamlined process possible. Keep in mind that there are other parts involved in this process, but none of them are absolutely necessary at this particular juncture. So later expect to revisit this concept with regard to Layer 3 VPNs:

- **Step One** – Create Virtual Router
  - Create a virtual router called VPN_A
  - Specify a route descriptor
- **Step Two** – Assign L3 Interface to VRF
  - Enter interface configuration mode
  - Associate the interface with the Virtual Router
  - Assign an ip address
- **Step Three** – Show the status of the virtual router
  - Show the routes

We will walk through this process on a single router and discuss the aspects of this configuration in detail. We will begin on R4. Later in our discussion R4 will one of a couple of routers that will be referred to as provider edge (PE) devices. A provider edge route will typically be where VRFs are configured in the services provider controlled cloud. VRF will not typically exist anywhere other than devices that are directly attached to customers. This concept of the provider edge is an important one and we will revisit it in future discussions quite often.

Step one tells us that we need to create a virtual router. In this case we will name it VPN_A per the instructions. This is done by starting in the global configuration mode on R4.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#ip vr
R4(config)#ip vrf ?
  WORD  VPN Routing/Forwarding instance name

R4(config)#ip vrf VPN_A
R4(config-vrf)#
```

Now that we have the vrf instance created we see that we are in the vrf configuration context. We will take this opportunity to explore here:

```
R4(config-vrf)#?
VPN Routing/Forwarding instance configuration commands:
  bgp           Commands pertaining to BGP
  context       Associate SNMP context with this vrf
  default       Set a command to its defaults
  description   VRF specific description
```



```
exit           Exit from VRF configuration mode
export         VRF export
import         VRF import
maximum        Set a limit
mdt            Backbone Multicast Distribution Tree
no             Negate a command or set its defaults
rd             Specify Route Distinguisher
route-target   Specify Target VPN Extended Communities
vpn            Configure VPN ID as specified in rfc2685
```

We see that we have a limited number of options and by the time we finish this book we will have covered most if not all of these commands, but right now the command we are interested in is the **rd** command. We need this command to configure the second task under step one. The route distinguisher is part of the basic functionality of MPLS based VPNs. To make this part of the configuration clear and understandable we will briefly discuss this concept here, but we will cover it great detail in ***Chapter 4: MPLS VPNs***. Right now we will simple acknowledge the fact that MPLS VPNs use BGP to communicate between PE routers. This allows the customer routes to be exchanged between these devices, and is made possible by extensions made to BGP that allow the transport of no IPv4 address packets. This constitutes the fundamentals of Multi-Protocol BGP (MBGP) that will be discussed in **Chapter 5: MP-iBGP**. For the purpose of this discussion we are only going to look at the route distinguisher, the most well-known extension in MBPG.

What does the RD do? Well the RD makes any prefix value on any router unique in the cloud. The route distinguisher is added as a prefix value to a traditional IPv4 address. This combination of the 32 bit IPv4 address and the 64 bit route distinguisher creates a new address entity. This new 96 bit long address is called a VPNv4 or VPN-IPv4 address. When a router learns of an IPv4 prefix via an interface that is participating in vrf forwarding then the RD for that given vrf is prefixed as we stated earlier, and this address is now translated into the VPNv4 prefix. This means that if the same IPv4 address exists in multiple client networks the use of different RD allows MBGP treat them as unique based on the new VPNv4 address.

So this means that an RD is just a 64 bit number; it does not provide any information. It is only used to translate an IPv4 prefix into VPNv4 prefix, making same IPv4 prefix a completely different VPNv4 prefix, allowing BGP to distribute these VPNv4 prefixes. But traditionally, there is some logic that goes into the customary creation of an RD. For the purposes of our discussion an RD will be typically made up of two fields:

- **Administrator Field** – should be set to the Autonomous System Number (ASN- only public ASNs should be used) that is assigned by the appropriate authority.
- **Assigned Number Field** – contains a number from a numbering space that is administered by the enterprise to which the ASN is assigned. This number is arbitrary in nature but should fit into some type of intuitive numbering schema.

Again for the purposes our discussion we will assume that our ASN is 65100 and we will use the assigned number field of 100 for customers in the vrf VPN_A.


```
R4(config-vrf)#rd 65100:100
R4(config-vrf)#exit
R4(config)#
```

We can see the results of this configuration via the **show ip vrf** command:

```
R4#show ip vrf
  Name                              Default RD          Interfaces
  VPN_A                             65100:100
```

We can clearly see the RD has been assigned to the vrf VPN_A, but note that there are no interfaces participating in the vrf instance. Step two is where we assign an interface to this vrf:

```
R4(config)#interface FastEthernet0/1
R4(config-if)#ip vrf forwarding VPN_A
% Interface FastEthernet0/1 IP address 172.16.45.4 removed due to enabling VRF VPN_A
R4(config-if)#
```

Observe that once the ip vrf forwarding command is entered the existing IP address is removed from interface FastEthernet0/1. This is a result of the need for the ip address to be associated with the RD. Once the address is reapplied we will look at the output of the show ip vrf command again.

```
R4(config-if)#ip address 172.16.45.4 255.255.255.0
R4(config-if)#exit
R4(config)#end
R4#show ip vrf
  Name                              Default RD          Interfaces
  VPN_A                             65100:100           Fa0/1
```

We see that the FastEthernet0/1 interface is now participating in the vrf VPN_A. This means that the new VPNv4 address that will be used by MBGP will be 65100:100:172.16.45.4. Before we move on we need to take a closer look at exactly what has been done to the R4's routing table. In accordance with step 3 we will run the **show ip route** command:

```
R4#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.24.0 is directly connected, FastEthernet0/0
C    192.1.4.0/24 is directly connected, Loopback0
```

Oddly enough we do not see the prefix 172.16.45.0/24. This is because now we have partitioned R4 into believing that it is now two routers. The output of the show ip route command above lists the routes

and prefixes found in the “global routing table”, to see the prefixes in the “VPN_A” routing table we will need to instruct the router to show that output, with the command **show ip route vrf VPN_A**:

```
R4#show ip route vrf VPN_A

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.45.0 is directly connected, FastEthernet0/1
```

The route is visible in the routing table for VPN_A, but what about reachability? Can we ping that interface from R4?

```
R4#ping 172.16.45.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.45.4, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

The address is not reachable. We can use the **debug ip packet** command to find out why.

```
R4#debug ip packet
IP packet debugging is on
R4#
R4#ping 172.16.45.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.45.4, timeout is 2 seconds:

IP: s=0.0.0.0 (local), d=172.16.45.4, len 100, unroutable.
IP: s=0.0.0.0 (local), d=172.16.45.4, len 100, unroutable.
IP: s=0.0.0.0 (local), d=172.16.45.4, len 100, unroutable.
IP: s=0.0.0.0 (local), d=172.16.45.4, len 100, unroutable.
IP: s=0.0.0.0 (local), d=172.16.45.4, len 100, unroutable.
Success rate is 0 percent (0/5)
R4#
R4#u all
All possible debugging has been turned off
```

We see that this ping is unsuccessful based on the destination being unroutable. The router is using this output to tell us that the destination prefix is not found in the global routing table. You have to remember that when we do not specify a given vrf Cisco IOS will assume we want to use the global routing table. So it correct this we will specify the proper vrf in the ping command:


```
R4#ping vrf VPN_A 172.16.45.4

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.45.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

A very useful command to see all the routing tables running on given device is **show ip route vrf ***.

```
R4#show ip route vrf *
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.24.0 is directly connected, FastEthernet0/0
C    192.1.4.0/24 is directly connected, Loopback0

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.45.0 is directly connected, FastEthernet0/1
```

We need to also realize that CEF is just as important to us now as it was at the beginning of this discussion. Now that we have an active vrf we will find that not only have we partitioned our RIB but we have also partitioned our cef adjacency table and FIB.

```
R4#show ip cef 172.16.45.4
0.0.0.0/0
  no route
R4#
```

There is no entry for 172.16.45.4 in the “global cef table” instead it is in the table associated with the VPN_A vrf:

```
R4#show ip cef vrf VPN_A 172.16.45.4
```



```
172.16.45.4/32
  receive for FastEthernet0/1
```

This clearly illustrates that CEF is built independently for the global routing and for each VRF.

The creation of the virtual routing and forwarding instances is the first step in establishing the overall MPLS architecture at the command line interface. This becomes the first component we have authoritative control over that defines how an MPLS backbone will build label forwarding tables and actually forward labeled traffic. The VRF instances we define on each device directly affects the following MPLS components:

Forward Equivalence Class

A forwarding equivalency class (FEC) is how a group of IP packets that share specific label will be forwarded. However, it should be pointed out that the a more accurate term than packet would be IP prefixes or routes due to the fact that these elements can and will more often than not share a particular label. Thus they will be treated equivalent in forwarding. This is not to say that a given FEC cannot reflect treatment for a specific prefix verses a group of prefixes. In fact a FEC can be as generic or as granular as we as administrators need it to be.

Control Plane

With MPLS, routers determine how to forward a give FEC or labeled back in the identical fashion employed traditionally to forward IP packets via ip routing. The decision is based on the incoming label of a particular packet. This process involves a consultation of the forwarding table to determine the interface that will be used to forward the labeled packet. Then the actual forwarding process will take place. In this discussion routing is the movement of packets (labeled or otherwise) from one network to another, where forwarding is the actual process of migrating a packet (labeled or otherwise) between interfaces on a given device.

The most basic concept that drives the inner workings of MPLS is the dynamic creation of the label forwarding information base (LFIB) from router to router. In a similar fashion as that used by our IGP routing protocols, information exchange takes place to between MPLS speakers to create this table. This process is best described as the formation of the MPLS control plane, and defines the process whereby labels are bound to network prefixes found in the FIB. This process requires the distribution of label information between devices. This process will be addressed in depth in the following chapter where we discuss MPLS labels, but at this time we need to understand that MPLS speaking device will dynamically exchange label information such that they can create their own discreet LFIB. The information that is exchanged by this process is the local label assigned by the router itself and the outgoing label that will be used to switch the traffic to a neighboring device.

To summarize, up to this point the router has assigned a label to each prefix found in the RIB. The processes refers to these prefixes as FECs (Forwarding Equivalency Classes), and all prefixes that share the same label will be treated equivalent in forwarding. This information is then advertised to any of the routers MPLS peers. The resulting database of FECs, interfaces and assigned labels is referred to as the Label Information Base (LIB).

Forwarding Plane

Now that we have created the LIB on all MPLS speaking devices in the topology we have everything we need to forward labeled packets. But the process does not stop here. Labels that are actually in use on a given device (due to being physically located or adjacent to our local MPLS router) are selected from the LIB and added to a new database called the Label Forwarding Information Base (LFIB). The LFIB is the database used by the local device to make forwarding decisions.

To reiterate this process:

The LFIB is the database of information that is used to forward labeled packets. The LFIB contains the best local and remote labels taken from the label information base (LIB) that was constructed in the control plane phase of MPLS. This database manages a wide range of MPLS operations to include labeled packet forwarding decisions, drop decisions (label switched packets that arrive on an interface that are not in the LFIB are dropped), and "best source" selection.

The purpose of the MPLS Data Forwarding plane is to create a labeled switch path through the network. This is a virtual path across which traffic will be forwarded from one end of the network to the other. This is the main premise of MPLS label switching process in terms of label distribution and the building of the label forwarding tables. To be able to understand even the most basic aspects of the operation and troubleshooting of MPLS the concepts presented in this chapter must be understood fully. To that end the following diagram details the control and forwarding plane processes in a top down fashion in the hope that a high level view will make things easier to understand.

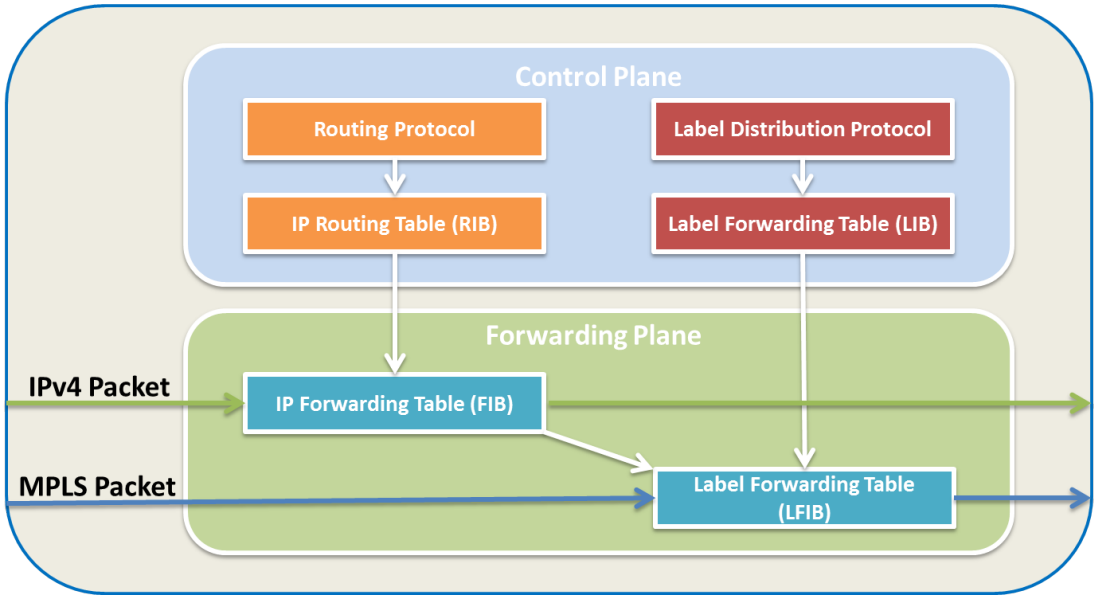


Figure 1-2: MPLS Control and Forwarding Plane

Observe that if an IPv4 Packet arrives the packet is forwarded based on the results of the typical FIB and RIB best path determination process used by the underlying routing protocol. However, in the event that an MPLS (a labeled) packet arrives the forwarding process is determined based on the results of the a process of best path selection process between the LIB and LFIB. The process is similar but still slightly

different from the normal routing process. Notice the concept of the Label Distribution Protocol at the top right of the figure. In ***Chapter 3: MPLS Label Distribution*** we will discuss the actual label distribution process that takes place between the MPLS peers we described cursorily in this chapter. But first we need a better understand of what a label is and how it helps us. This concept will be discussed at length in ***Chapter 2: MPLS Labels***.

Chapter 1 Challenge: Troubleshooting Tools

1-1. Based on the exhibit provided what show command was executed on R4?

```
R4#??????????
  Name                              Default RD          Interfaces
  VPN_A                             65100:100           Fa0/1
```

1-2. Based on the exhibit provided what show command was executed on R4?

```
R4#??????????
VRF VPN_A; default RD 65100:100; default VPNID <not set>
  Interfaces:
    Fa0/1
VRF Table ID = 1
  No Export VPN route-target communities
  No Import VPN route-target communities
  No import route-map
  No export route-map
  VRF label distribution protocol: not configured
  VRF label allocation mode: per-prefix
```

1-3. Based on the exhibit provided what show command was executed on R4?

```
R4#???????????
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.24.0 is directly connected, FastEthernet0/0
C    192.1.4.0/24 is directly connected, Loopback0

Routing Table: VPN_A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.45.0 is directly connected, FastEthernet0/1
```

1-4. To ping the ip address 172.16.45.5 from R4 what would the syntax of the command look like?

_____

1-5. The RD value is used to define what VPN a give prefix belongs to. (True or False)

Chapter 1 Challenge: Multiple Choice Review Answers

1-1. show ip vrf

1-2. show ip vrf detail VPN_A

1-3. show ip vrf VPN_A route

1-4. show ip route vrf

1-5. False – the RD is used to ensure that a give prefix is unique in the MPLS cloud. This has nothing to do with what VPN the prefix belongs to.

Chapter 2: MPLS Labels

In this chapter we examine what MPLS labels look like, how they are represented, what operational characteristics they define, and how they are managed during the MPLS switching process.

Fundamental Concepts

To make certain that we have a clear grasp on the concepts and basic fundamentals of MPLS we will review things one more time, and add some additional information to our understanding. The driving motivation behind the creation of MPLS was the desire to improve the forwarding speed of routers, however this improvement goal quickly became a secondary objective as new and enhanced capabilities have been integrated into MPLS over time. The single greatest capability that MPLS brings us is Traffic Engineering (TE) but this topic is currently not part of the Cisco's routing and switching lab exam. But suffice it to say that traffic engineering gives us the ability to mandate the path that traffic will take through the network. So that begs the question what does MPLS offer us in the R&S lab exam. The answer is Virtual Private Network support. VPNs, specifically layer 3 VPNs a key application where MPLS is superior to any currently available IP technology.

MPLS was designed to operate independent of Layer 2, and was initially created to provide a more effective means of deploying IP networks across service provider's traditional ATM-based backbones. This is accomplished thanks to the concept of labels, and has found wide scale adoption in networks outside the service provider cloud.

The most basic fundamental of MPLS is the creation and assignment of a short fixed-length label that performs like a summarized representation of an IP packet's header. A perfect analogy that we can compare this too is how the US postal service employs ZIP codes to represent house, street and city in a postal address. Just like the mail carrier label switch router will make use of this label to make forwarding decisions regarding labeled packets. Traditional IP packets have a field in their 'header' that contains the address to which the packet is to be routed this information is then processed at every router in a packet's path on a hop-by-hop basis. But in MPLS, the IP packets are encapsulated with these labels, that we have been discussing, by the first MPLS device they encounter as they enter the network. These devices are referred to as Label Edge Routers (LER) or more commonly Provider Edge (PE) routers.

The job of the PE router is to analyze the contents of the standard IP header and select an appropriate label with which to encapsulate the packet. MPLS's popularity and wide scale adoption has come from how it, in contrast to conventional IP routing, can base it's routing decisions on more than just the destination address carried in the IP header. But the single most significant difference between MPLS and tradition routing is that as a packet is delivered via the label switched network each subsequent node makes forwarding decisions based on the MPLS label, and not the IP header. Ultimately an MPLS labeled packet will leave the LSP via a PE router at a distant end of the provider or customers MPLS backbone. At this juncture that PE device will remove any labels that have been added to the IP packet.

In ***Chapter One: Introduction to MPLS*** we discussed the fundamental aspects of the mechanisms behind the creation of the control plane and the forwarding plane. We also discussed the comparative similarities between the traditional RIB and FIB and the MPLS concepts of the LIB and LFIB, and how that these new databases contained information about the local and remotely learned labels that would be used to forward labeled packets. Now to complete our basic understanding, we will look closely at the labels themselves. At this juncture we are not interested in the mechanisms that managed their

propagation and exchange; that will be covered at length in ***Chapter 3: Label Distribution***. In this chapter we are concerned with what labels look like, how are the represented, what operational characteristics do they define, and how they are managed during the MPLS switching process.

Forward Equivalency Class

In the previous chapter we discussed how labels identify prefixes that will be switched along the same path and how MPLS calls these prefixes forward equivalency classes or FEC's. The important thing to note at this point is that FECs, meaning prefixes that share the same label are all going to be treated in the same fashion. It is easy to remember the term FEC if we keep in mind that all prefixes defined by a FEC will be forwarded equally, thus the name forward equivalency. We add the term "class" because MPLS performs classification as labeled packets enter the router. So when viewed in this way the term FEC makes perfect sense.

Label Payloads

MPLS labels do not contain payload information, this is because intermediate devices known as Label Switch Routers (LSR) or more commonly Provider routers (P) have no need to know or even interest in the nature of the packet payload in order to do their jobs. It needs to be made clear that the PE edge devices do not need to have the payload information either because where the packet will exit the MPLS backbone is where the local binding will have need initially created, and the FEC is already ware of the initial payload type. As a result of this behavior of MPLS labels, we obtain the "payload agnostic" capability that is so highly sought after by large multi-protocol environments.

Label Numbers

MPLS Labels are represented by numbers. In fact the total number of labels that a Cisco IOS device can support varies from platform to platform but for the purposes of our discussion we will focus on the capabilities of the Cisco 2811 router running IOS version 12.4(24)T2. To determine the number of labels supported by one of these devices we can execute the **show mpls label range** command:

```
R1#show mpls label range
Downstream Generic label region: Min/Max label: 16/100000
```

We immediately see some interesting information as a result of this output. Note that the minimum assigned label number is 16, and for this device the maximum is 100,000. That begs the question, "why aren't labels 0-15 used?" The answer is that they are in fact used but they are reserved by the MPLS process itself for special operations. We will begin by looking closely at the special functions assigned to each of the labels in the range that have been defined. Currently only labels 0-3 have been adopted for use, leaving the remaining labels in the range for future proof expansion.

The current labels that have been defined inside the reserved range manage one of two aspects of MPLS operations. These aspects can best be describe as how label switched packets will be treated end-to-end across the LSP, and then how to tell if a packet requires hardware or software switching. We will first being by looking at the Alert Label. Alert labels will always have a label number 1.

Router Alert Label (Label 1)

This label can appear anywhere in the label stack except at the bottom. When the Router Alert label is the top label, it serves to notify that particular device that the packet needs to be managed separately. Specifically this means that the packet will not be forwarded in hardware. Instead the packet will be forwarded at the software level. At this time the Router Alert will be removed and a lookup based on the next label in the label stack will be performed. The forwarding information will be taken from the LFIB to decide where to switch the packet. At this time, a label operation is performed, and the Label 1 is pushed back on top of the label stack, and the packet is forwarded.

Note: We will cover label operations in detail later in this chapter, but at this time we do need to understand some of the basic terms. We have predominately three label operations to consider:

- **PUSH** – Adding a label to an incoming packet (Imposition)
- **POP** – Remove the label from an outgoing packet (Disposition)
- **SWAP** – Replace the label of an incoming packet

Right now it is only important to understand the terms. We will provide a detailed explanation of each process and where that process takes place along the LSP of the back-bone later in this chapter.

The concept of a NULL label defines how a given labeled packet will be treated as it exits the MPLS domain. There are two types of NULL labels: implicit and explicit. Both implicit and explicit null labels are generated by the last hop router and are advertised to its neighbors as part of the MPLS control plane formation process we discussed in ***Chapter One: MPLS Overview***.

Implicit null labels are assigned by default during this process, meaning the next to last router should only send IP packets to the last router in the backbone. This means the next to last router pops the label. There is however, one disadvantage in the implicit null approach. The problem is that if the network is configured for QoS based on information found in the assigned label, then QoS is lost between the second to last and last hop router. This means that by default any MPLS driven QoS mechanism will not extend end-to-end. Remember that QoS is most commonly designed hop-by-hop but it needs to be deployed end-to-end to function correctly.

The label used to signal this behavior is:

Implicit NULL Label (Label 3)

An egress PE (the last router in the LSP) will assign the implicit NULL label to any directly connected and/or summarized routes, thus notifying the upstream LSR to perform a pop operation. This upstream LSR or the second to last LSR in the LSP is called the ***penultimate hop*** and because the label pop operation is taking place on that specific device we use the term **Penultimate Hop Popping (PHP)** to refer to this behavior. As we mentioned previously this PHP process is the default mode on Cisco IOS devices running MPLS.

We have discussed the disadvantages associated with PHP, but what about the advantages. Well without an Implicit-null label the Egress PE route would receive packets with only one label on top of the

label stack. It would then have to do two lookups. First, it would have to look up the label in the LFIB, just to figure out that the label needs to be removed; then it would have to perform an IP lookup. These are lookups are actually unnecessary and a waste of time from a forwarding point of view. So by relying on PHP and using the implicit-null label a double lookup is prevented. The egress PE signals the penultimate router to use implicit NULL by not sending a regular label by sending it a label with a value of 3. The result is that the egress PE receives an IP packet and only needs to perform a single IP lookup to be able to forward the packet. This enhances the performance on the egress PE.

This leaves us with the concept of an Explicit Null. In this case, we can make use of Explicit null which means that penultimate hop router does not pop the label. It sends the packet to the last hop LSR with a label value of 0 but keeps any QoS markings in the label intact. This way QoS treatment is preserved between the penultimate LSR and last hop LSR. Explicit null should be configured manually on the last hop router with the **mpls ldp explicit-null** command:

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#mpls ldp explicit-null ?
  for  Access-list specifying controls on destination prefixes
  to   Access-list specifying controls on LDP peers
  <cr>

R1(config)#mpls ldp explicit-null
R1(config)#
```

We see based on the output of the context sensitive help that we have options but at this time we will simply press return. Now the R1 will send a its neighbors a label of:

Explicit NULL Label (Label 0 or Label 2)

The use of implicit NULL solved the double look up problem but it had one downside: The packet is forwarded with one label less than it was received by the PHP LSR or unlabeled if it was received with only one label in the stack. We mention that the label does not just contain the label value, the label also holds any QoS bits. When a label is removed, the this information is lost. Because this data is exclusively used for quality of service (QoS), the QoS part of the packet is lost when the top label is removed. In some cases, we may need to keep this QoS information and have it delivered to the egress LSR. Implicit NULL cannot be used in that case, so Explicit-null is the solution to overcome the downside of implicit-null.

Because the egress LSR signals the IPv4 explicit NULL label (value 0 for IPv4 or 2 for IPv6) to the PHP router. The egress PE then receives labeled packets with a label of value 0 as the top label. The LSR cannot forward the packet by looking up the value 0 in the LFIB because it can be assigned to multiple FECs. The PE will remove the explicit NULL label, then another lookup will occur, but the advantage is that the router can receive the QoS information of the received packet by looking at any quality of service information inside the explicit NULL label.

To ensure that we understand the information we have discussed up to the point regarding MPLS reserved labels please refer to the follow summarization:

Reserved Label Numbers at a Glance

0 – IPv4 Explicit NULL

- PHP LSR does not pop the label. It is sent to the egress PE which only looks at the QoS information in the label and pops the label without looking it up in the LFIB
- ONLY an IPv4 lookup is made.

1 – Router Alert v4/v6

- Router POPs the label, examines the packet, performs LFIB lookup, and pushes one label.
- CAN NOT be set at the bottom layer.

2 – Ipv6 Explicit NULL

3 – IPv4 Implicit NULL

- Advertised to the Penultimate LSR to POP the label and send the packet untagged.
    - used for connected and aggregate prefixes
- PHP - no need for egress PE to perform two lookups (label and IP).
- ONLY ONE LABEL is POP'd off at the PHP.

Label Mode Types

When it comes time to discuss MPLS Label Mode type and operations we have two choices to deal with, but one of the types namely Cell Mode is outside of the scope of the CCIE R&S. That leaves us with Frame Mode MPLS. In Frame Mode MPLS an ingress PE router receives a normal packet and then performs the following actions:

- **Step 1** – A routing lookup is performed to determine the outgoing interface
- **Step 2** – A label is assigned and inserted between the layer two frame header and the layer 3 packet header **if** the outgoing interface is enabled for MPLS and a "next-hop label" for the destination exists in the LIB.
- **Step 3** – The labeled packet is then forwarded out of the designated interface.

Frame mode MPLS is used with protocols that utilized frame based L2 headers, as mentioned in step 2 of the outlined operational process, the label will be inserted between the L2 frame header and the L3 packet header, this is the reason that MPLS is often time referred as a Layer 2.5 protocol. Additionally, many people call the label being inserted a "shim header". On additional process takes place that need to be mentioned. At the time the MPLS label is inserted a modification his made to the L2 frame header itself. This modification serves to indicate that the packet is now a labeled packet. This modification comes in the form of rewriting the protocol identifier field. The following are common protocol identifiers used in this process:

- Eth 0x8847 - IPv4 unicast
- Eth 0x8848 - IPv4 Multicast
- PPP 0x0281
- HDLC 0x8847
- FR 0x80
- IEEE SNAP with Eth 0x8847

Now that we have discussed the additions and the modification that are made to the ip packet as it arrives we need to look closer at the actual significance of these packets, what he mean some of the governances regulating there assignment.

Label Assignment

We still have not reached the point where we are interested in how labels are assigned. That again will be covered in the **Chapter 3: Label Distribution**. Right now was are only interested in what the labels represent and any rules we may need to know about how they are created. The first and possible the most important thing to remember at this stage of our discussions is that labels are only locally significant. That means that the same label can be used multiple times throughout our MPLS topology. So as we describe a label or its characteristics it is necessary to keep in mind that a label may represent a give FEC on router B and a completely different FEC on router C. In fact it should be mentioned that each LSR binds given FEC to a given label in a process that is whole independent of any other device in the domain. This means that a different label will be assigned to each FEC in the table. This behavior is universal for all routing protocols and prefixes with the exception of BGP. In the case of BGP a single label will be assigned to all prefixes that share the same BGP next hop.

Label Switched Path

Assignment of these local labels is performed by the label switch routers. Label generation/distribution protocols, which we will discuss in detail later, are responsible for label creation within an LSR. The following series of figures are designed to elaborate further on how label exchange helps create the label switch path.

We will look closer at the label assignment process.

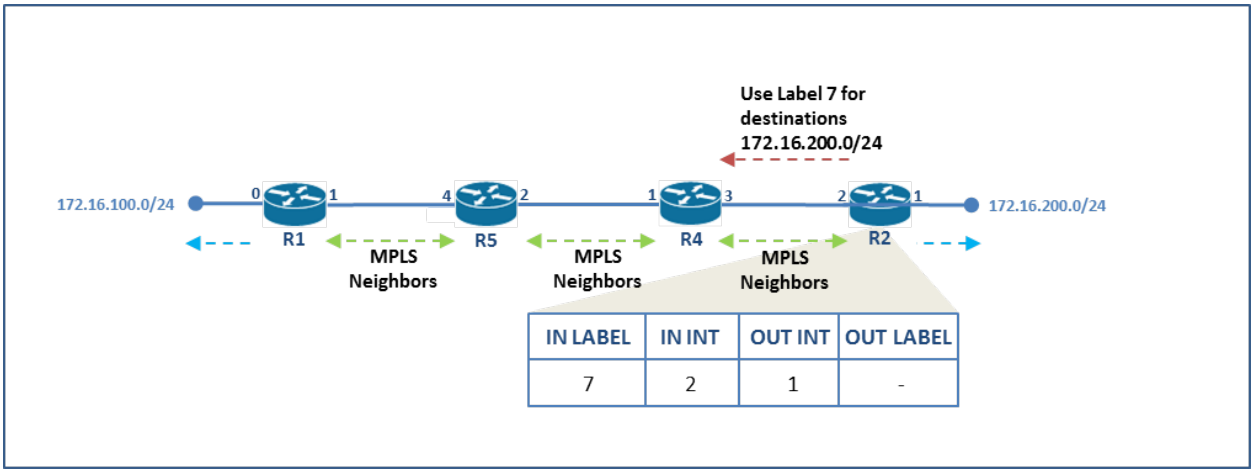


Figure 2-1: LSP creation phase 1

Initially the PE R2 will create a binding between the FEC and the label value. In this instance the label value will be 7 and it will be bound to the FEC for the prefix 172.16.200.0/24. It must be understood that this label we are describing is purely arbitrary. But through various mechanisms the devices in the MPLS domain will exchange label information. The ultimate outcome of this process means that along the label switched path all adjacent routers will have a common view of FEC to label bindings. To illustrate this process we will follow this logic through the LSP formation for the FEC 172.16.200.0/24.

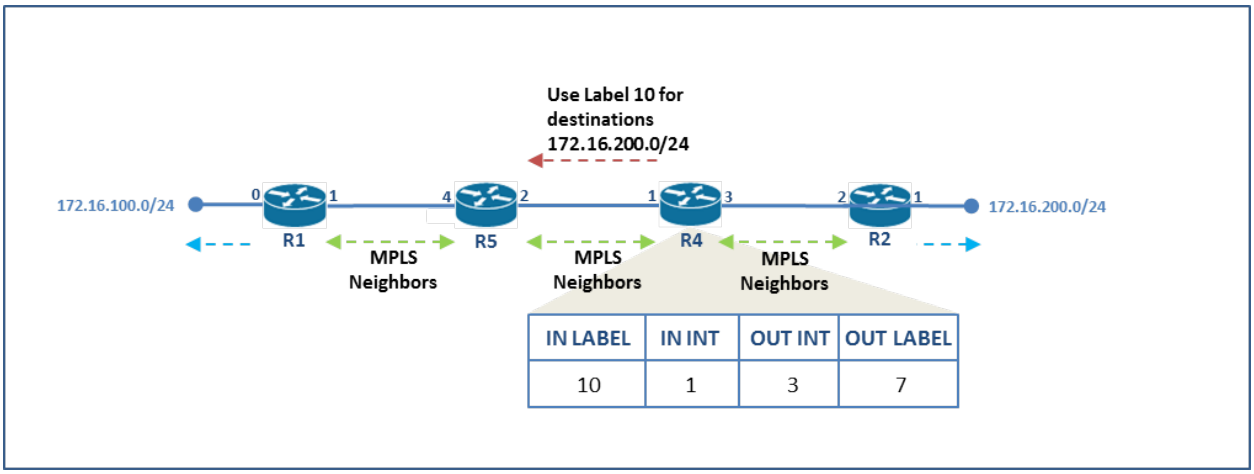


Figure 2-2: LSP creation phase 2

In Step 2 of this process the LSR R4 receives notification that R2 (PE) will expect to receive packets with label 7 for the FEC 172.16.200.0/24. R4 will assign the FEC 172.16.200.0/24 its own label. For the purposes of this discussion R4 will assign label 10 and then communicate that label information to its upstream neighbor R5.

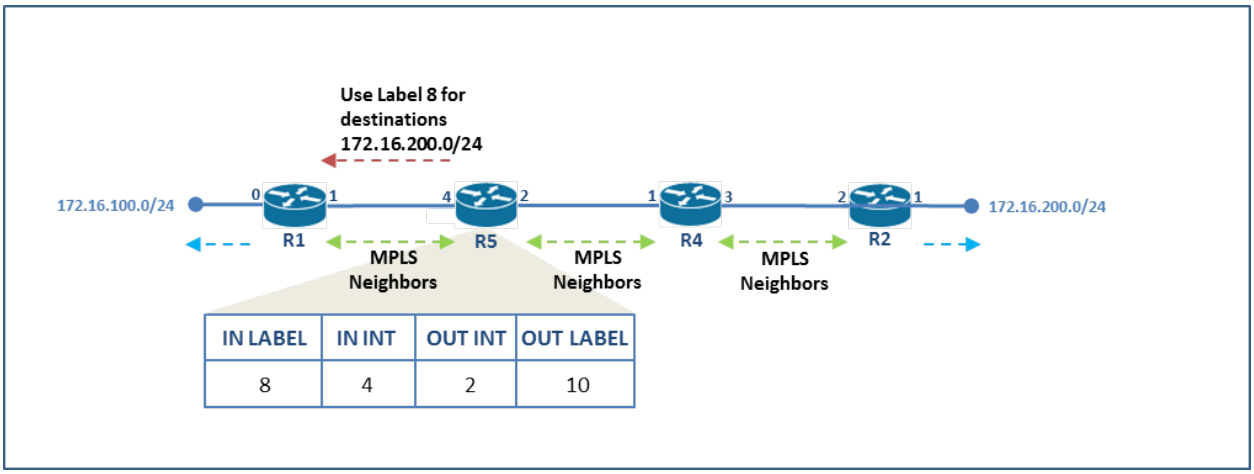


Figure 2-3: LSP creation phase 3

In Step 3 of this process the LSR R5 receives notification that R4 will expect to receive packets with label 10 for the FEC 172.16.200.0/24. R5 will assign the FEC 172.16.200.0/24 its own label. For the purposes of this discussion R5 will assign label 8 and then communicate that label information to its upstream neighbor R1.

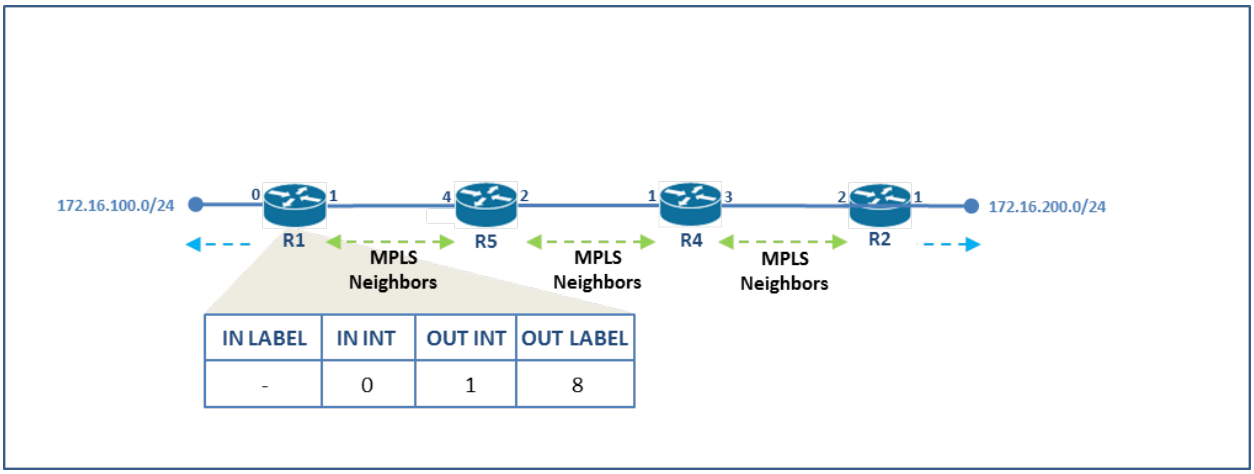


Figure 2-4: LSP creation phase 4

In Step 4 of this process the PE R1 receives notification that R5 will expect to receive packets with label 8 for the FEC 172.16.200.0/24. R1 will assign no label for the FEC 172.16.200.0/24 because it has no upstream MPLS neighbor. We have just observed the creation of the MPLS control plane, which in turn will drive the actual forwarding plane creation that will allow traffic sourced from R1 destined to the network 172.16.200.0/24 on R2 to be label switched rather than ip packet switched. We will look at this process now.

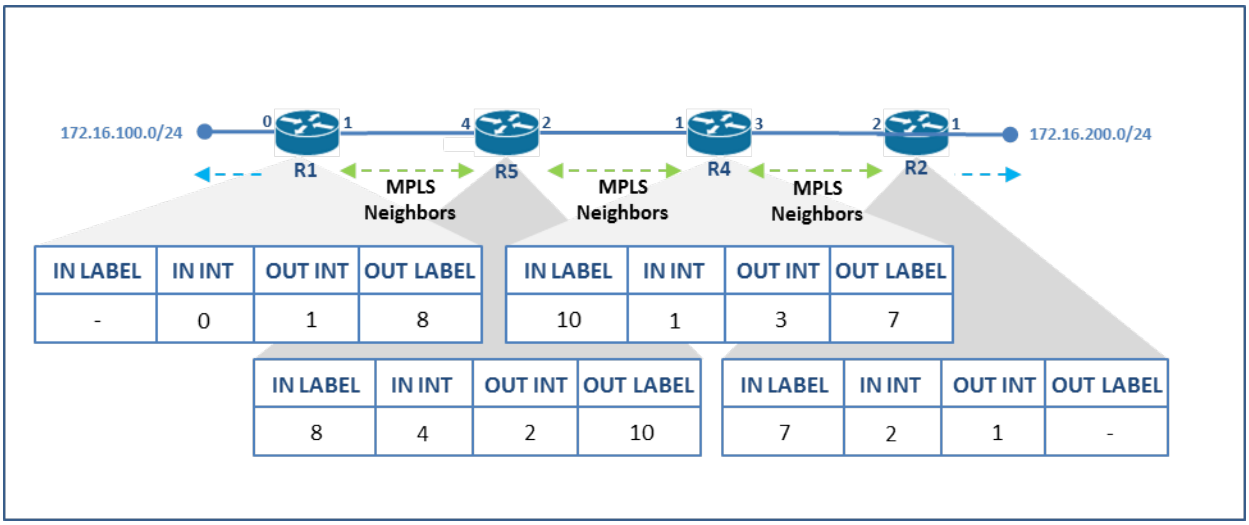


Figure 2-5: The Forwarding Plane

Now that each device in the label switch path has created its own LFIB by virtue of binding labels to FEC we now have a predetermined path that label switched packets will take to reach the network 172.16.200.0/24. We also can extrapolate the labels that will be used during this forwarding process on a hop-by-hop basis. To clarify this process we will visualize an IP packet arriving on R1's interface 0 that is destined for the target off R2's interface 1. We absolutely have to understand that this packet will arrive with no label applied. This can be seen in the table on R1 as "IN LABEL" value "-". The lookup process starts on R1, and reveals that the FEC 172.16.200.0/24 exists in the LIB and that entry has a next-hop label of 8 assigned. This satisfies the conditions necessary for R1 to forward the packet via

MPLS. This means that a label of 8 is now inserted into the originating ip packet. This is a perfect description of the PUSH label operation.

The packet will leave R1 for the next hop router R5. As the now labeled packet arrives on R5, the LIB look up process happens again and R5 seeing the Label of 8 knows that it must forward that packet out interface 2 with a label of 10 toward R4. This describes the SWAP label operation process. The same process will take place on R4 and the ultimate outcome will be that R4 will label switch the packet with an outbound label of 7. Again this is a SWAP operation.

Ultimately in our topology R2 will receive the packet. The packet will have an inbound label of 7 but the corresponding outbound interface is not running MPLS nor so obviously there will be no next-hop label. This means that the packet will be forwarded in the traditional fashion to the network located out R2's interface 1. To do so R2 must remove the inbound label, and perform a FIB look up. This is a perfect example of a POP label operation. This description is to illustrate the process and was not created to take the explicit or implicit NULL operation into consideration. **Figure 2-6: The Label Switch Path Operation** details the process we have just described.

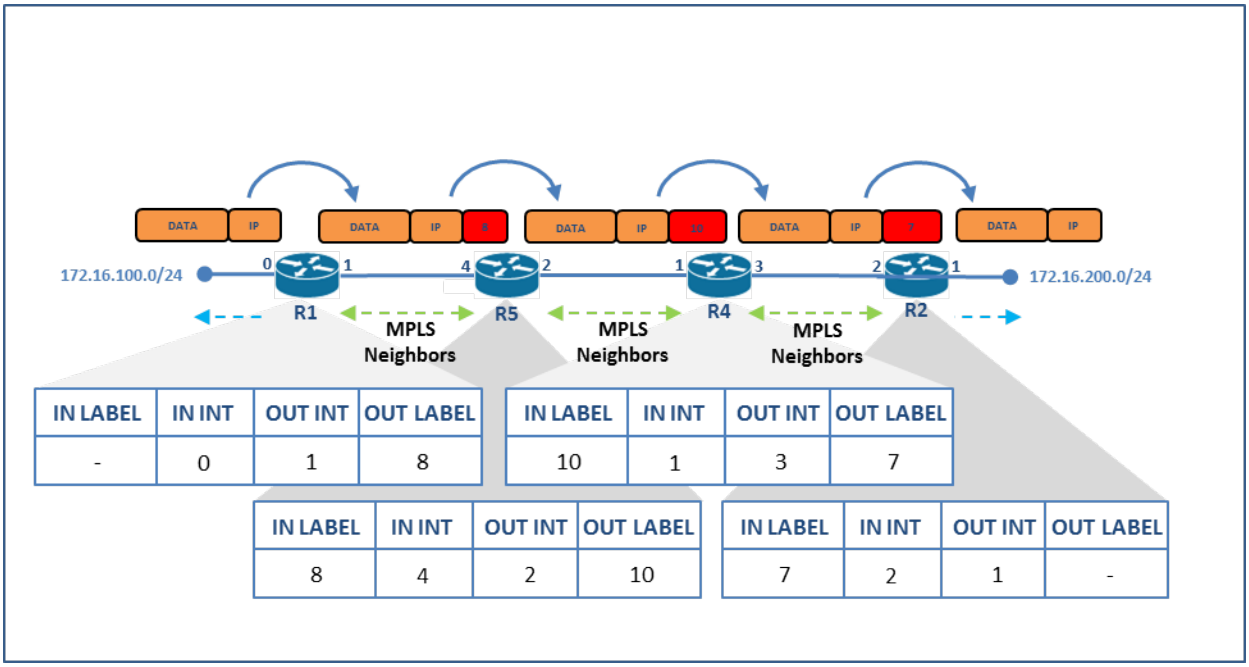


Figure 2-6: The Label Switch Path Operation

It is vital that we understand that label switch paths are unidirectional in nature. Also we need to recognize that the forwarding plane will work in the opposite direct of the control plane. Think of it as the device that first creates a FEC to Label binding has the primary task of notifying is MPLS peers of what label it has bound to a particular FEC. Each of those neighbors in turn notify theirs upstream peers of what label they have bound to that FEC. So it would be safe to say that control plane formation flows in an upstream fashion, where data forwarding is performed along the downstream path.

Distribution Modes

We have observed how a label is assigned to a destination prefix found in the FIB, and how it is distributed to upstream neighbors in the MPLS domain and how label binding of a specific prefix to a local label and a next-hop label (received from downstream LSR) is then stored in the LFIB and LIB structures. No matter what protocol is used for label distribution there are two primary methods used in MPLS to govern the actual process used to "solicit a neighbor's label bindings:

Downstream on Demand (DOD) – This mode of label distribution allows an LSR to explicitly request from its downstream next-hop router a label mapping to a particular destination prefix and is known as downstream on demand label distribution. DOD is used with ATM and is therefore outside our scope with regard to our discussion. But it is still noteworthy.

Unsolicited downstream – This mode of label distribution allows an LSR to distribute bindings to upstream LSRs that have not explicitly requested them and is referred to as unsolicited downstream label distribution. This mode of label binding distribution is used in frame mode MPLS and as such will govern all label distribution models we will discuss in **Chapter 3: Label Distribution**.

Retention Modes

We have spent a lot of time discussing how labels are created and learned from downstream peers, but that does leave one element that governs this process uncovered. How long does an LSR maintain a list of label bindings for FECs that it learns from peers that are not the next-hop for the particular FEC? In Cisco IOS we have two modes of operation regarding this question that we are going to discuss and each will offer us a different answer to our question. The first is conservative retention mode.

In conservative retention (CLR) mode bindings are removed from the LIB after the best next-hop is chosen and placed in the LFIB. This means that only the "best" binding will be stored in the LIB. The biggest benefit to this approach is the reduced necessity for memory, but the negative impact is long convergence times resulting from routing changes that will affect the resolution of the next-hop for a given destination. The reason convergence is an issue in this mode of retention is the fact that a new label must be obtained from the new next hop before labeled packets can be forwarded. Conservative mode retention is a default retention mode selected on ATM interfaces.

The remaining retention mode is Liberal Label Retention (LLR) mode. In Downstream Unsolicited distribution mode, label mapping advertisements for all routes may be received from all MPLS peers. LLR means that, every label mapping received from any peer LSR will be retained regardless of whether the LSR is the next hop for the advertised mapping or not. The main advantage of the LLR mode is that recovery reaction time to routing changes for the next-hops can be quick because alternate labels already exist in the LIB. The main disadvantage of the liberal mode is that unneeded label mappings are distributed and maintained. Even though LLR modes can be used in both ATM and Frame mode, it is the default retention mode on any non-ATM interface.

MPLS Label Format

We have discussed the MPLS label only in terms of its creation and binding to a give FEC. Now it is time to look at the actual format of the label its self. An MPLS header is 32-bits long and contains the following fields:

- **The Label Field** – 20-bits used to carry the actual value of the MPLS label.
- **The Class of Service (CoS) Field** – 3-bits use to affect the queuing and discard algorithms applied to the packet as it is transmitted through the network. Since the CoS field has 3 bits which means that 8 distinct service classes can be maintained.
- **The Label Stack (S) Field** – 1-bit used to support a hierarchical label stack. Although MPLS supports a stack, the processing of a labeled packet is always based on the top label, without regard for the possibility that some of other labels may have been above it in the past, or that some number of other labels may be below it at present. An unlabeled packet can be thought of as a packet whose label stack is empty (i.e., whose label stack has depth 0). If a packet's label stack is of depth *n*, we will refer to the label at the bottom of the stack as the level 1 label, to the label above it (if such exists) as the level 2 label, and to the label at the top of the stack as the level *n* label. The label stack is used for routing packets through LSP Tunnels. In Chapter 4:Layer 3 VPNs we will discuss this process in depth.
- **The TTL (time-to-live) field** – 8-bits that provide conventional IP TTL functionality

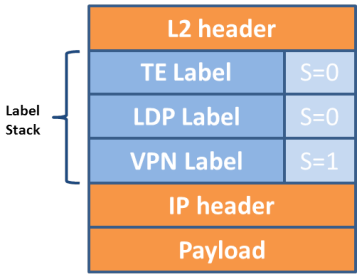
Label Stack

In the previous section we covered the 1 bit in the MPLS header that designates a hierarchical label stack. That means, for the purpose of our discussions, we will look closer at how labels can be stacked in the IP header. It is necessary to point out that one of the labels that can be stacked is referred to as the TE label and it will not be part of our studies at this juncture, but it is an ideal example of how multiple labels can be stacked. To illustrate this process we will look at the IP header and note how multiple labels can been added between the Frame header and the IP header.

Bottom of Stack Bit

- 1 - bottom label, next is IP header
- 0 - more labels follow

TE - Identified TE tunnel end-point, used by P, and PE routers
LDP - Used by P routers to label-switch packets between LSR's
VPN - label identifies VRF, used by the PE Egress LSR does not do an IP lookup for the VPN label, because the LFIB already points to the proper NH along with the Interface and L2 rewrite data.



Observe how the multiple labels in blue each have a value of 0 or 1 in each label that represents the bottom of stack bit value. This means that labels will be removed in a top down fashion as the labeled packet is switched across the MPLS domain. Note that the last label has the “bottom of stack” bit set to one which means that the label will be removed, and the IP packet will be routed via traditional treatment.

MPLS TTL

By default **mpls ip propagation-ttl** is enabled in global configuration mode. This enables the ability to trace the hops of the mpls routers with labels by using the traceroute command. This is because the MPLS TTL field is copied from the IP TTL field when the initial label is assigned, and on each subsequent MPLS LSR hop that TTL will be decremented.

To "hide" the MPLS hops it is possible to disable this behavior by executing the **no mpls ip propagation-ttl** command on every LSR in the service provider cloud. Disabling MPLS propagation TTL will change the MPLS TTL field to a value of 255, and on every MPLS LSR hop the IP TTL value will be kept intact. The IP TTL will only be decremented when the egress PE sends it out to the destination host as an unlabeled packet. Additionally, Cisco IOS will overwrite the IP TTL if the MPLS TTL is higher as part of its loop prevention mechanism. Keep in mind that the egress PE will not copy the label's TTL into the IP TTL. This process will in effect hide the core topology of the service provider. It is important to mention that one hop will be shown with the cumulative delay.

If the TTL reaches the value of zero on a LSR an ICMP time exceeded message will be sent forward along the LSP to the downstream LSR. This time exceeded message can only be used by IPv4 and IPv6 packets. It is important to realize that this message will be sent with a TTL of its own of 255.

MPLS MTU

This value specifies whether a labeled packet being forwarded through an interface will be fragmented. MPLS MTU can be higher than the actual interface's MTU, but the router will issue a warning notifying us that this condition could result in data corruption, high CPU utilization and packet drops.

In the case that the interface MTU is less than 1524 bytes, you can set the maximum MPLS MTU to 24 bytes more than the interface MTU. However, if the interface MTU is equal to or greater than 1524 bytes, then you can set the maximum MPLS MTU equal to the interface MTU.

The default behavior is for the MPLS MTU to automatically match that of the interface MTU when the router first initializes. If the interface MTU is changed, the MPLS MTU will also change to match that value automatically. However, the reverse is not true.

The MPLS MTU can be changed via interface configuration mode using the **mpls mtu** command.

```
R1(config-if)# mpls mtu 1492
```

Additionally Cisco recommends when this command is used, the value used on the interface should be less than or equal to interface MTU.

MRU

Maximum Received Unit (MRU) is a Cisco-proprietary parameter which implicitly means MTU per FEC.

This value is used to inform a neighboring LSR of the largest packet (in bytes) that can be forwarded for a given FEC or destination prefix without fragmentation taking place. The MRU will changes per prefix

depending on the label operation performed on the prefix. To view the exact value of the MRU for an FEC, use the **show mpls forwarding-table <*prefix*> detail** command.

Chapter 2 Challenge: Multiple Choice Questions

Chapter 2 Challenge: Multiple Choice Review Answers

Chapter 3: MPLS Label Distribution

LDP is the protocol that dynamically creates and governs the label switched path in our network. So everything we have discussed up to this point regarding label generation, assignment, and exchange is maintained by this single protocol.

Introduction to Label Distribution

Thus far in this book we have discussed the values and characteristics of labels, we have dealt with their meaning and how they affect the creation of both the control and forwarding planes used in MPLS. Throughout this entire process we have loosely referred to the exchange of labels between label switching devices. Now it is time to discuss the process and mechanism behind this exchange or advertising process.

There are two methods at our disposal to exchange labels and their associated bindings between devices that are peered. The key word here is "peered". Whether we are using the Cisco proprietary Tag Switching or the industry standard Label Distribution Protocol (LDP) as the mechanism of choice to communicate label information, it all still boils down simply to peering relationships and the rules governing those peers. For the purpose of our discussion we will deal specifically with the LDP from this point forward in our discussions.

When MPLS is enabled on an integrated services router (ISR) LDP is the default label distribution protocol. It is this protocol that dynamically creates and governs the label switched path in our network. So everything we have discussed up to this point regarding label generation, assignment, and exchange is maintained by this single protocol. Additional, to these local processes LDP also is responsible for the formation of the peering relationships we mentioned earlier. These relationships facilitate the exchange of our local label bindings between peers.

At this time we feel that we need to define and innumerate the major operational roles this protocol will be responsible for in terms our discussions in this chapter. Primarily we can outline three such functions:

- Dynamic discovery of adjacent LDP Peers
- Peering establishment
- Regulation of peer-to-peer communication and label exchange

We will initiate this introduction to the concepts associated with these four major operational function in their own individual sections.

Dynamic Discovery of Adjacent LDP Peers

To learn more about the operation and detailed functions associated with LDP you can read RFC 3036, where the process of peer discovery and adjacency formation is discussed in depth. However, for the purposes of our discussions and the application of this knowledge toward the goal of fault isolation and remediation we will make specific references to only a portion of the information in the RFC here in this chapter.

For the purposes of our discussion we are going to look at LDP using the same context as that used to understand a link state protocol like say OSPF. We know that OSPF or even EIGRP will first attempt to form a peering relationship with neighboring devices with whom it shares a network segment. We also know that once this process produces adjacencies between devices these adjacencies are constantly

probed to determine if they are still up and operational via things like HELLO packets. The good news here is that LDP performs much the same way. There are rules governing each phase of these processes that relate specifically to LDP but we will be careful to point them out as we progress through this discussion.

We have effectively defined three operational phases in this section, first a neighbor discovery phase, then a session establishment phase, and lastly a session maintenance phase. We will first look closely at the first of these three phases to better understand the process that is taking place.

Neighbor Discovery

When it comes to neighbor discovery there are to methods that LDP can rely on using traffic destined to the well-known UDP port 646. This process describes the discovery HELLO mechanism used by LDP to find neighbors. However, there are actually two mechanisms used to do this. Method one is referred to as the basic neighbor discovery mechanism. With basic neighbor discovery the LSR uses multicast hellos to communicate with directly connected neighbors. However, there is a mechanism used to allow peering between non-directly connected devices using a unicast discovery HELLO. This non-directly connected peering is known as extended neighbor discovery. It must be pointed out that whether basic or extended neighbor discovery mechanisms are in play the discovery communication in LDP is always going to be via UDP 646.

We can see this process clearly if we configure R1 to run mpls in the topology as seen in **Figure 3-1**.

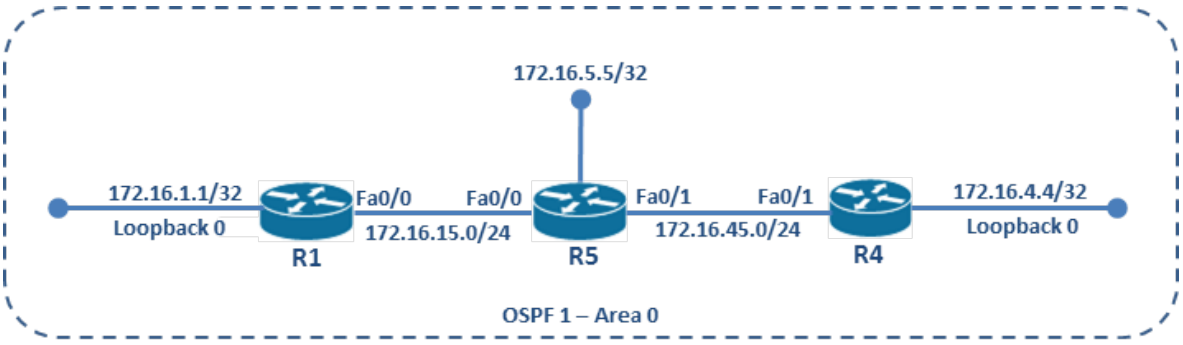


Figure 3-1: Basic MPLS topology

In this demonstration we will configure R1 such that we will capture all ip packets inbound our outbound that use UDP port 646. We will accomplish this via an access-list to filter our **debug ip packet** command.

```
R1(config)#access-list 100 permit udp any any eq 646
R1(config)#access-list 100 permit udp any eq 646 any
R1(config)#exit
R1#debug ip packet detail 100
IP packet debugging is on (detailed) for access list 100
```

Now we will enable mpls on R1 and observe the results as they transpire.

```
R1(config)#interface FastEthernet0/0
R1(config-if)#mpls ip
```

This will immediately result in output associated with the debug operation:

```
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending
broad/multicast
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending full packet
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending
broad/multicast
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending full packet
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending
broad/multicast
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending full packet
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending
broad/multicast
    UDP src=646, dst=646
IP: s=172.16.15.1 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending full packet
    UDP src=646, dst=646
```

Observe how R1 is now sending ip packets destined to the link local multicast address 224.0.0.2 both sourced and destined to the UDP port 646. First the router sends what it calls a "broad/multicast" packet then it subsequently sends a "full" packet. The point here is that LDP is attempting to find a neighbor that is running MPLS LDP. Can we see these packets arrive on R5 in our topology?

We can see what happens on R5 by using the same filter and the same debug:

```
R5>en
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#access-list 100 permit udp any any eq 646
R5(config)#access-list 100 permit udp any eq 646 any
R5(config)#end
R5#debug ip packet detail 100
IP packet debugging is on (detailed) for access list 100
R5#
```

As packets arrive on R5 we see some interesting behavior:

```
IP: s=172.16.15.1 (FastEthernet0/0), d=224.0.0.2, len 62, rcvd 0
    UDP src=646, dst=646
FIBipv4-packet-proc: route packet from FastEthernet0/0 src 172.16.15.1 dst 224.0.0.2
FIBfwd-proc: Default:224.0.0.0/24 receive entry
FIBipv4-packet-proc: packet routing failed
pak 47D578D4 consumed in input feature , packet consumed, MCI Check(64), rtype 0,
forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
```


Observe that the packet sourced from R1 (172.16.15.1) arrives and R5 does not know what to do with it. Note the "packet routing failed" message. What would happen if we activated MPLS on R5's FastEthernet0/0 interface?

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface FastEthernet0/0
R5(config-if)#mpls ip
R5(config-if)#end
```

We see now that R5 sends its own packets out.

```
IP: s=172.16.15.5 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending
broad/multicast
    UDP src=646, dst=646
IP: s=172.16.15.5 (local), d=224.0.0.2 (FastEthernet0/0), len 62, sending full packet
    UDP src=646, dst=646
```

Observe that on R1 we see the following message:

```
R1(config)#
*Jun 19 16:38:50.534: %LDP-5-NBRCHG: LDP Neighbor 192.1.5.5:0 (1) is UP
```

According to this output we now have a neighbor adjacency between R1 and R5, we can verify this via the use of the **show mpls ldp neighbor** command:

```
R1#sh mpls ldp neighbor
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.21288 - 192.1.1.1.646
        State: Oper; Msgs sent/rcvd: 10/11; Downstream
        Up time: 00:04:24
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.5
        Addresses bound to peer LDP Ident:
          172.16.15.5     192.1.5.5
```

We have just observed the natural behavior of LDP. The moment two neighbors exchange LDP Discovery Hello messages these devices will establish an LDP session. The bad part of this demonstration is that this mechanism happens to fast for us to truly be able to observe the two step process used to establish LDP sessions.

Peering establishment

Up to this point we have discussed the exchange of UDP messages using multicast, but now we need to look at the next phase of LDP communication. TCP is used to establish peering sessions between LDP neighbors. In the demonstration above we clearly observed this process. But we can take a closer look at this fact via the **show tcp brief** command on each of our peers:

```
R1#show tcp brief
TCB       Local Address               Foreign Address             (state)
498D80D8  192.1.1.1.646               192.1.5.5.21288             ESTAB
```



```
R5#show tcp brief
TCB        Local Address               Foreign Address             (state)
47C08A40   192.1.5.5.21288             192.1.1.1.646               ESTAB
```

This output clearly demonstrates that we have a TCP peering session that is in the established state. We will use this output to look at the first part of this two-step process. We will call this initial phase the transport connection phase, and during this operation one of two actions can transpire. If the two peers have never previously established a TCP session then they will do so immediately. During this process the two devices must decide with of them will be the active device in the relationship. The active device will use a TCP port from the ephemeral (high random) range and is more commonly referred to as the client. Where the server will be the device that will be listening for connections on the well-known TCP port of 646. In this situation the device with the highest ip address will play the role of the client and the lower will be the server. We can see this by again looking at the output of the **show tcp brief**:

```
R1#show tcp brief
TCB        Local Address               Foreign Address             (state)
498D80D8   192.1.1.1.646               192.1.5.5.21288             ESTAB
```

We see that R1 is the server in this peering, by virtue of the local TCP port being 646, this again is mandated because of the lower IP address. This begs the question, “what happens when a previous peering exists?” The answer is if the two LSRs already have a TCP session between them, possibly one over another interface, a new TCP session will not be created. The key element to this phase of the process is the creation of the transport connection. Which immediately brings us to the final phase of this process.

Session Establishment

Immediately after the establishment of the transport connection our devices will begin to negotiate session parameters. LDP messages will be used to negotiate a number of variables that extend to the LDP protocol version, the label exchange method, and the operational timer values. Immediately after this negotiation process the LDP TCP peering session will be fully established.

If some element fails in this process, possibly due to incompatible configurations, the routers will exchange special messages called Error Negotiation Messages. These Error messages are then used to signal that the devices should attempt to renegotiate the session. The issue with this particular connection dynamic is the possibility of constantly repeating a series of renegotiation attempts. One tool created to attempt to address this “looping” process of failed negotiations involves the deployment of an exponential initiation and backoff values. By default Cisco IOS will use 15 seconds for the LDP initial backoff value, and 120 seconds for the LDP maximum LDP backoff value.

We can clearly see the values assigned to these timers via the **show mpls ldp parameters** command.

```
R1#show mpls ldp parameters
Protocol version: 1
Session hold time: 180 sec; keep alive interval: 60 sec
Discovery hello: holdtime: 15 sec; interval: 5 sec
Discovery targeted hello: holdtime: 90 sec; interval: 10 sec
Downstream on Demand max hop count: 255
```

```
Downstream on Demand Path Vector Limit: 255
LDP for targeted sessions
LDP initial/maximum backoff: 15/120 sec
LDP loop detection: off
```

It is important to note that we see values outlining other timers. Specifically we have address the meaning and purpose of the LDP initial/maximum backoff value but please note there are other timers also specified in this output. These values come into play after the TCP session has been established and are a vital part in the care and maintenance of the operation connection between peers. This brings us to our third and final primary function of LDP.

Regulation of Peer-to-Peer Communication and Label Exchange

We have observed the process used to dynamically discover LDP neighbors. We actually monitored the establishment of the TCP session used to manage the LDP communication process. Now we are going to look critically at the actual process employed by Cisco IOS to maintain LDP sessions. We have seen how the label distribution protocol works to allow neighbor discovery and session establishment, and it should come as no surprise that the maintenance mechanisms build into LDP also works to maintain and verify the LDP process at both of these levels as well. Hello Adjacency messages are used to maintain the neighbor peering states, while LDP session maintenance is accomplished by sending and receiving keep alive messages between peers to maintain the connection at a session level.

We have looked at the processes involved in LDP and we have mentioned that these process are directly governed at various levels in the protocol architecture by timers. Before we look at the benefits and advantages of manipulating these timers we should take a closer look at what they do and how they do it.

Timers

The timers that we are describing regulate the transmission and anticipated arrival of both discovery messages and keep-alives. Based on these timer values, LDP will isolate a failed session, reset an existing session or bring up a new session. These values allow us to manipulate these processes and criteria from the command line.

As an example LDP relies on the regular arrive of HELLO Adjacency messages to determine a peers intent to use a defined label space. The label space in question is identified within the hello. All label switch routers will therefore maintain a hold timer value for each HELLO Adjacency with gets reset as a new HELLO arrives from a give peer. If this time expires before a new HELLO Adjacency arrives then it is assumed that the peer no longer want to participate in the session. This means that the peer will not take part in label switching, the existing label space for that peer will be considered invalid and a notification message will be sent to terminate the LDP session. This also will result in the actual TCP session being closed as well.

Timers that affect LDP neighbor discover include:

- **(G) mpls ldp discovery hello interval <sec>**
  - Default every 5 seconds
- **(G) mpls ldp discovery hello holdtime <sec>**

- Default every 15 seconds

We have discussed the timers that work with discovery but now we need to move to the timers that are used to maintain the actual LDP sessions themselves. It must be noted that these timers only come into play once an LDP session has been established. It may be easiest to think of these timers as those that affect session integrity or what is referred to colloquially as session "housekeeping".

Where HELLO Adjacencies where used previously we are now looking at keep-alive messages. These messages are typically sent every 60 seconds by default. The keep-alive timer for each peer session resets whenever it receives any packet on that session. If the keep-alive timer expires, LDP treats that session as if the TCP connection is down.

Timers that affect LDP Session maintenance include:

- **(G) mpls ldp holdtime <sec>**
  - Keep-alive time is reset every time LDP packets or keep-alive (60 sec) are received.
  - Default is 180 sec
  - Keep-alive values are automatically adjusted to 1/3 of the configured holdtime

Command that we will use to determine the assigned values for these timers on a local basis is:

```
R1#show mpls ldp parameters
Protocol version: 1
Session hold time: 180 sec; keep alive interval: 60 sec
Discovery hello: holdtime: 15 sec; interval: 5 sec
Discovery targeted hello: holdtime: 90 sec; interval: 10 sec
Downstream on Demand max hop count: 255
Downstream on Demand Path Vector Limit: 255
LDP for targeted sessions
LDP initial/maximum backoff: 15/120 sec
LDP loop detection: off
```

But to see what values have been negotiated during the peering process describe in the previous section we will rely on **show mpls ldp neighbor detail**:

```
R1#show mpls ldp neighbor detail
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.21288 - 192.1.1.1.646
        Password: not required, none, in use
        State: Oper; Msgs sent/rcvd: 261/265; Downstream; Last TIB rev sent 6
        Up time: 03:45:12; UID: 2; Peer Id 0;
        LDP discovery sources:
          FastEthernet0/0; Src IP addr: 172.16.15.5
            holdtime: 15000 ms, hello interval: 5000 ms
        Addresses bound to peer LDP Ident:
          172.16.15.5     192.1.5.5
        Peer holdtime: 180000 ms; KA interval: 60000 ms; Peer state: estab
```


Be mindful of the fact that the neighbor detail output will provide values in measurements of milliseconds. Furthermore, keep in mind that should the need arise to change timers those values affecting keep-alive will require a session reset to take effect. We can observe this in output provided below:

```
R1(config)#mpls ldp holdtime 100
%Previously established sessions may not use the new holdtime.
R1(config)#
```

This entire discussion has brought us to the point where we can now cover the actual exchange and advertisement of labels that is handled via LDP. We will begin by taking a comprehensive look at the standard behavior of LDP, and then at the tools and mechanisms we have at our disposal to manipulate this underlying processes involved.

Label Distribution and Control

Thus far we have discussed the creation of the peering session between LSRs. It must be mentioned that for each such session the LSRs must agree on the label distribution method that will be employed between them. We discussed the options that we have for distribution methods in ***Chapter 2: MPLS Labels***.

Given that we are concerned with the performance of MPLS on the ISRs we will be working with in the CCIE lab exam, we will focus specifically on Unsolicited Downstream (UD) distribution mode with Independent LSP Control mode. Both of these configurations are default operational parameters frame mode MPLS.

In Independent LSP mode our devices will automatically begin advertising label mappings to their neighbors. This process may take place at any time due the UD distribution mode we discussed earlier, meaning that no signaling is necessary for requests between peers. Each LSR will create bindings for prefixes as soon as they appear in the RIB. This means that they are either connected or being received via an IGP. These LSR's may now advertise a label mapping for a given FEC to neighbors as soon as the label binding is made to a FEC. This order operation in non-deterministic in nature due to the fact that an upstream label can be advertised before a corresponding downstream label is received.

When labels are passed along the LSP we see two primary categories of devices. A internal LSR known as a provider router and an egress device known as a label edge router (LSR) or more commonly a PE. The determining factor as two which role a give LSR assumes in regard to a give FEC is based on one of the following conditions:

- The FEC refers to the LSR itself (including one of its directly attached interfaces).
- The next hop router for the FEC is outside of the Label Switching Network. Meaning no label will exist in the LIB for the next-hop address.
- The prefix is reachable by crossing a routing domain boundary, such as another area for OSPF, or possibly even another autonomous system.

In Unsolicited Downstream label distribution mode, label mapping advertisements for all routes may be received from all LDP peers. Based on the default retention mode used by frame mode MPLS, every label mapping received from a LDP peer is retained regardless of whether the LSR is the next hop for the advertised mapping. The immediate benefit to this mode of operation is that the response time to routing changes is very fast because labels will already exist in the LFIB. A disadvantage is that any unused label mappings will be distributed to peers where the will be maintained in the LFIB due to the default liberal retention mode we discussed in ***Chapter 2: MPLS Labels***.

Using this model that we just discussed any given LSR will maintain all learned labels in the LIB, and in that database there will be an entry for each prefix that has an associated collection of pairs. These pairs are comprised of the LDP identifier and the label value itself. As a result one such pair will exist for each peer advertising a label for any given prefix.

Should a situation occur where a next hop value changes for a given FEC, the local LSR must consult the LIB to obtain the new label to use for packet forwarding. In order to obtain this label from the LIB the router must be able to map the next hop address for the prefix in question to a reachable LDP Identifier.

To observe this process we will return to the network displayed in Figure 3-1 where we will look first at the normal operation of ipv4 routing on R1. For the purpose of this discuss we will advertise multiple loopback addresses into the OSPF domain on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface loopback 1
R5(config-if)#ip address 1.1.1.1 255.255.255.255
R5(config-if)#interface loopback 2
R5(config-if)#ip address 2.2.2.2 255.255.255.255
R5(config-if)#interface loopback 3
R5(config-if)#ip address 3.3.3.3 255.255.255.255
R5(config-if)#interface loopback 4
R5(config-if)#ip address 4.4.4.4 255.255.255.255
R5(config-if)#interface loopback 5
R5(config-if)#ip address 5.5.5.5 255.255.255.255
```

Now we will redistribute the connected interfaces into the OSPF process.

```
R5(config-if)#router ospf 1
R5(config-router)#redistribute connected subnets
```

This will result in these routes appearing as type E2 prefixes on R1 and R4:

```
R1#show ip route ospf | inc E2
O E2    1.1.1.1 [110/20] via 172.16.15.5, 00:03:14, FastEthernet0/0
O E2    2.2.2.2 [110/20] via 172.16.15.5, 00:03:14, FastEthernet0/0
O E2    3.3.3.3 [110/20] via 172.16.15.5, 00:03:14, FastEthernet0/0
O E2    4.4.4.4 [110/20] via 172.16.15.5, 00:03:14, FastEthernet0/0
O E2    5.5.5.5 [110/20] via 172.16.15.5, 00:03:14, FastEthernet0/0
R1#
```


The same results should appear on R4:

```
R4#show ip route ospf | inc E2
O E2    1.1.1.1 [110/20] via 172.16.45.5, 00:01:14, FastEthernet0/1
O E2    2.2.2.2 [110/20] via 172.16.45.5, 00:01:14, FastEthernet0/1
O E2    3.3.3.3 [110/20] via 172.16.45.5, 00:01:14, FastEthernet0/1
O E2    4.4.4.4 [110/20] via 172.16.45.5, 00:01:14, FastEthernet0/1
O E2    5.5.5.5 [110/20] via 172.16.45.5, 00:01:14, FastEthernet0/1
R4#
```

Now we will enable MPLS on R1, R5 and R4. We will do this at the interface level and observe the results on a hop-by-hop basis. This process will allow us to progress through our configuration and see what a working deployment should look like. We will use the knowledge later in this chapter to optimize our fault isolation and remediation skills.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip cef
R1(config)#mpls ldp router-id lo 0 force
R1(config)#interface FastEthernet0/0
R1(config-if)#mpls ip
R1(config-if)#end

R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#ip cef
R5(config)#mpls ldp router-id lo 0 force
R5(config)#interface FastEthernet0/0
R5(config-if)#mpls ip
R5(config-if)#exit
R5(config)#interface FastEthernet 0/1
R5(config-if)#mpls ip
R5(config-if)#end
R5#

R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#ip cef
R4(config)#mpls ldp router-id lo 0 force
R4(config)#interface FastEthernet0/1
R4(config-if)#mpls ip
R4(config-if)#end
```

Based on our understanding of how LDP works we can assume that we now have peering relationships using TCP port 646 between R1, R5 and R4. This can be observed using the **show mpls ldp neighbors** command on a hop by hop basis:

```
R1#show mpls ldp neighbor
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.35524 - 192.1.1.1.646
        State: Oper; Msgs sent/rcvd: 22/22; Downstream
        Up time: 00:04:45
```



```
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.5
        Addresses bound to peer LDP Ident:
          172.16.15.5     172.16.45.5     192.1.5.5       1.1.1.1
          2.2.2.2         3.3.3.3         4.4.4.4         5.5.5.5
```

We need to take the time to look at this output from R1. We see that R1 has peered with R5. We know this based on the Peer LDP Identifier of 192.1.5.5:0. The Identifier number is composed of the 32 bit address used by R5 to establish the transport session and the value of "0" after the colon is a two byte number used to represent the "label space". This value of 0 states that the label space is platform wide and based on the devices and protocols that are considered in scope for the CCIE R&S we will not see any other values here but "0".

Next we see the line for "State" note that we are sending and receiving messages and by repeating the show command we can see that these values increment over time:

```
R1#show mpls ldp neighbor
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.35524 - 192.1.1.1.646
        State: Oper; Msgs sent/rcvd: 36/36; Downstream
        Up time: 00:11:40
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.5
        Addresses bound to peer LDP Ident:
          172.16.15.5     172.16.45.5     192.1.5.5       1.1.1.1
          2.2.2.2         3.3.3.3         4.4.4.4         5.5.5.5
R1#
```

The value has increased from 22 to 36. Note that there is a one to one relationship between these values. We send 36 and received 36. Should these numbers be different we would know there is an issue related to either the link connecting these devices or the configuration of the devices themselves.

Lastly, we need to look at the "Address bound to peer LDP Identifier". This line and the subsequent lines of output tell us exactly what ip label-bindings are being communicated to us from R5. This are the FECs that have Labels assigned to them on R5. Note that we see the loopback addresses we created on R5 clearly in this output. Before we go to R5 we need to take a look at the labels we are learning from R5. To do this we need to look at the output of the show mpls forwarding-table on R1:

```
R1#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     Pop Label     1.1.1.1/32       0             Fa0/0      172.16.15.5
17     Pop Label     2.2.2.2/32       0             Fa0/0      172.16.15.5
18     Pop Label     3.3.3.3/32       0             Fa0/0      172.16.15.5
19     Pop Label     4.4.4.4/32       0             Fa0/0      172.16.15.5
20     Pop Label     5.5.5.5/32       0             Fa0/0      172.16.15.5
21     Pop Label     172.16.45.0/24   0             Fa0/0      172.16.15.5
22     52            192.1.4.4/32     0             Fa0/0      172.16.15.5
23     Pop Label     192.1.5.5/32     0             Fa0/0      172.16.15.5
```

Note that we see the labels that are being advertised to us and the labels that R1 is assigning to each FEC. The point is that each prefix representing the loopbacks we created are shown to have a value of "Pop Label" rather than an actual label number. What does this mean?

In **Chapter 2: MPLS Labels**, we discussed the concept of the implicit NULL and the explicit NULL. What we are seeing is the direct result of the implicit NULL concept as it applies to penultimate hop popping. We will return to this situation before we conclude this chapter, but now we are going to look at the labels relating to the prefix 192.1.4.4. We see that we have assigned a local label of 22 to this FEC. We learned this prefix from R5 and we can see that R5 is using the label value of 52 to represent it. So we will send any LSP destined for 192.1.4.4 out interface FastEthernet0/0 toward the next hop of 172.16.15.6 with an outgoing label of 52. We can see this clearly by using a traceroute to 192.1.4.4:

```
R1#traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.15.5 [MPLS: Label 52 Exp 0] 4 msec 0 msec 0 msec
  2 172.16.45.4 0 msec *  0 msec
```

We are send this packet via MPLS using the label 52, notice that it arrive on R4 as a standard ipv4 routed packet. Again this will be a result of the PHP process. It is time to turn out attention to R5 and look specifically at the output of the **show mpls ldp neighbor** and **show mpls forwarding-table** commands.

```
 R5#show mpls ldp nei
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.5.5:0
        TCP connection: 192.1.4.4.646 - 192.1.5.5.23715
        State: Oper; Msgs sent/rcvd: 1414/1401; Downstream
        Up time: 20:01:56
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 172.16.45.4
        Addresses bound to peer LDP Ident:
          172.16.45.4     192.1.4.4
    Peer LDP Ident: 192.1.1.1:0; Local LDP Ident 192.1.5.5:0
        TCP connection: 192.1.1.1.646 - 192.1.5.5.35524
        State: Oper; Msgs sent/rcvd: 16/16; Downstream
        Up time: 00:01:53
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.1
        Addresses bound to peer LDP Ident:
          192.1.1.1       172.16.15.1
R5#
```

R5 has formed LDP neighbors with R1 and R4 as we expect. We also see that each of these peers are advertising labels for two prefixes each. To look closer at these labels (both remotely learned and local orginated) we will use the **show mpls forwarding-table** command.

```
R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
```



```
51      No Label      192.1.1.1/32      33636         Fa0/0      172.16.15.1
52      Pop Label     192.1.4.4/32      1620          Fa0/1      172.16.45.4
R5#
```

Observe that only the learned prefixes are present in the LIB. But again our focus is on the label behavior. We see that any packet destined to the prefix 192.1.1.1 will be sent without a label to R1, and that any packet destined to the prefix 192.1.4.4 will have any label removed before it is sent out FastEthernet0/1. We can see this with the traceroute utility.

```
R5#traceroute 192.1.1.1

Type escape sequence to abort.
Tracing the route to 192.1.1.1

  1 172.16.15.1 4 msec *  0 msec

R5#traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.45.4 4 msec *  0 msec
R5#
```

Observe that all traceroute packets were delivered via normal ipv4 routing.

Penultimate Hop Popping (PHP)

At this time we are not going to deal with the specifics related to PHP that was already in covered in **Chapter 2: MPLS labels**. We are instead going to look at the process that is associated with the default implicit-null behavior as it relates to connected interfaces and there advertisement. First we will look specifically at what the implicit-null label advertisement that takes place between R5 and its LDP neighbors. We will do this by enabling the debug mpls

```
R1#debug mpls ldp bindings
LDP Label Information Base (LIB) changes debugging is on
R1#

R4#debug mpls ldp bindings
LDP Label Information Base (LIB) changes debugging is on
R4#
```

We will now clear the ldp neighbor configuration between R5 and its peers with the **clear mpls ldp neighbor *** command.

```
R5#clear mpls ldp neighbor *
R5#
```

Now we will look at the output on R1 and R4.

```
tagcon: announce labels for: 1.1.1.1/32; nh 172.16.15.5, Fa0/0, inlabel 16, outlabel
imp-null (from 192.1.5.5:0), get path labels
```

```
tib: 2.2.2.2/32:: learn binding 1 from 192.1.5.5:0
tib: a new binding to be added
tagcon: tibent(2.2.2.2/32): label imp-null from 192.1.5.5:0 added
tib: next hop for route 2.2.2.2/32(0, 172.16.15.5, Fa0/0) is mapped to peer
192.1.5.5:0
tib: invoke iprm label announcement for 2.2.2.2/32
tib: prefix recurs walk start: 2.2.2.2/32, tableid: 0
tib: get path labels: 2.2.2.2/32, tableid: 0, Fa0/0, nh 172.16.15.5
tib: found route info for 2.2.2.2/32(0, 172.16.15.5, Fa0/0), remote label Unknown
tib: update route info for 2.2.2.2/32(0, 172.16.15.5, Fa0/0), with remote label imp-
null from 192.1.5.5:0
tagcon: announce labels for: 2.2.2.2/32; nh 172.16.15.5, Fa0/0, inlabel 17, outlabel
imp-null (from 192.1.5.5:0), get path labels
tib: 3.3.3.3/32:: learn binding 1 from 192.1.5.5:0
tib: a new binding to be added
tagcon: tibent(3.3.3.3/32): label imp-null from 192.1.5.5:0 added
tib: next hop for route 3.3.3.3/32(0, 172.16.15.5, Fa0/0) is mapped to peer
192.1.5.5:0
tib: invoke iprm label announcement for 3.3.3.3/32
tib: prefix recurs walk start: 3.3.3.3/32, tableid: 0
tib: get path labels: 3.3.3.3/32, tableid: 0, Fa0/0, nh 172.16.15.5
tib: found route info for 3.3.3.3/32(0, 172.16.15.5, Fa0/0), remote label Unknown
tib: update route info for 3.3.3.3/32(0, 172.16.15.5, Fa0/0), with remote label imp-
null from 192.1.5.5:0
tagcon: announce labels for: 3.3.3.3/32; nh 172.16.15.5, Fa0/0, inlabel 18, outlabel
imp-null (from 192.1.5.5:0), get path labels
tib: 5.5.5.5/32:: learn binding 1 from 192.1.5.5:0
tib: a new binding to be added
tagcon: tibent(5.5.5.5/32): label imp-null from 192.1.5.5:0 added
tib: next hop for route 5.5.5.5/32(0, 172.16.15.5, Fa0/0) is mapped to peer
192.1.5.5:0
tib: invoke iprm label announcement for 5.5.5.5/32
tib: prefix recurs walk start: 5.5.5.5/32, tableid: 0
tib: get path labels: 5.5.5.5/32, tableid: 0, Fa0/0, nh 172.16.15.5
tib: found route info for 5.5.5.5/32(0, 172.16.15.5, Fa0/0), remote label Unknown
tib: update route info for 5.5.5.5/32(0, 172.16.15.5, Fa0/0), with remote label imp-
null from 192.1.5.5:0
tagcon: announce labels for: 5.5.5.5/32; nh 172.16.15.5, Fa0/0, inlabel 20, outlabel
imp-null (from 192.1.5.5:0), get path labels
tib: 4.4.4.4/32:: learn binding 1 from 192.1.5.5:0
tib: a new binding to be added
tagcon: tibent(4.4.4.4/32): label imp-null from 192.1.5.5:0 added
tib: next hop for route 4.4.4.4/32(0, 172.16.15.5, Fa0/0) is mapped to peer
192.1.5.5:0
tib: invoke iprm label announcement for 4.4.4.4/32
tib: prefix recurs walk start: 4.4.4.4/32, tableid: 0
tib: get path labels: 4.4.4.4/32, tableid: 0, Fa0/0, nh 172.16.15.5
tib: found route info for 4.4.4.4/32(0, 172.16.15.5, Fa0/0), remote label Unknown
tib: update route info for 4.4.4.4/32(0, 172.16.15.5, Fa0/0), with remote label imp-
null from 192.1.5.5:0
tagcon: announce labels for: 4.4.4.4/32; nh 172.16.15.5, Fa0/0, inlabel 19, outlabel
imp-null (from 192.1.5.5:0), get path labels
R1#
```

We can see that the "outlabel" for the each loopback address of R5 will use the "implicit-null" label. This is what causes R1 to pop the label before they send packets to these destination prefixes as we saw before. But we have the capacity to alter this default behavior by forcing the process to stop relying on the PHP behavior. This can be accomplished by telling R5 to use the explicit-null configuration.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#mpls ldp explicit-null
```

Observe that once this command is executed the label state and composition is changed on R1 and R5. For the purpose of our discussion we will focus on the events that take place on R1 regarding the prefix information and labels related to the FEC 5.5.5.5/32.

```
R1#
tagcon: tibent(5.5.5.5/32): label imp-null from 192.1.5.5:0 removed
tib: prefix recurs walk start: 5.5.5.5/32, tableid: 0
tib: get path labels: 5.5.5.5/32, tableid: 0, Fa0/0, nh 172.16.15.5
tib: found route info for 5.5.5.5/32(0, 172.16.15.5, Fa0/0), remote label Unknown
tib: update route info for 5.5.5.5/32(0, 172.16.15.5, Fa0/0), with remote label
Unknown
tagcon: announce labels for: 5.5.5.5/32; nh 172.16.15.5, Fa0/0, inlabel 20, outlabel
unknown (from 192.1.5.5:0), get path labels
tagcon: announce labels for: 5.5.5.5/32; nh 172.16.15.5, Fa0/0, inlabel 20, outlabel
exp-null (from 192.1.5.5:0), get path labels
```

First we see that the old implicit-null label learned from 192.1.5.5:0 is removed from the LIB, then there is a brief period where a series of recursive lookups take place until a label for the next hop can be found. Now observe that the outlabel has been changed to explicit-null. We can see the effects of this process by repeating the show mpls forwarding-table.

```
R1#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     explicit-n    1.1.1.1/32        0             Fa0/0       172.16.15.5
17     explicit-n    2.2.2.2/32        0             Fa0/0       172.16.15.5
18     explicit-n    3.3.3.3/32        0             Fa0/0       172.16.15.5
19     explicit-n    4.4.4.4/32        0             Fa0/0       172.16.15.5
20     explicit-n    5.5.5.5/32        0             Fa0/0       172.16.15.5
21     explicit-n    172.16.45.0/24    0             Fa0/0       172.16.15.5
22     52            192.1.4.4/32      0             Fa0/0       172.16.15.5
23     explicit-n    192.1.5.5/32      0             Fa0/0       172.16.15.5
R1#
```

Observe that the Pop Label entry we saw previously has now been replaced with the explicit-n label. This means that the PHP behavior has now been turned off. By turning off the PHP we now have end-to-end labels across the entire LSP. This as we discussed in ***Chapter 2: MPLS Labels*** is how we can leverage QoS throughout the entire cloud.

Label Filtering

Given the default behavior that we have been discussing regarding labels being advertised for every prefix found in the routing table, we may find it necessary to filter given labels from this exchange. We will first observe what happens if we simply stop this exchange on R5 with the **no mpls ldp advertise** command with the debugging process still running on R1.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#no mpls ldp advertise-labels
R5(config)#end
```

We will look specifically at the labels that are removed from the LIB on R1.

```
R1#show logg | inc removed
tagcon: tibent(1.1.1.1/32): label 0 from 192.1.5.5:0 removed
tagcon: tibent(2.2.2.2/32): label 0 from 192.1.5.5:0 removed
tagcon: tibent(3.3.3.3/32): label 0 from 192.1.5.5:0 removed
tagcon: tibent(4.4.4.4/32): label 0 from 192.1.5.5:0 removed
tagcon: tibent(5.5.5.5/32): label 0 from 192.1.5.5:0 removed
tagcon: tibent(172.16.15.0/24): label 0 from 192.1.5.5:0 removed
tagcon: tibent(172.16.45.0/24): label 0 from 192.1.5.5:0 removed
tagcon: tibent(192.1.1.1/32): label 51 from 192.1.5.5:0 removed
tagcon: tibent(192.1.4.4/32): label 52 from 192.1.5.5:0 removed
tagcon: tibent(192.1.5.5/32): label 0 from 192.1.5.5:0 removed
R1#
```

This means that we will have no labels learned for these remotely learned FEC from R5.

```
R1#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     No Label      1.1.1.1/32       0             Fa0/0      172.16.15.5
17     No Label      2.2.2.2/32       0             Fa0/0      172.16.15.5
18     No Label      3.3.3.3/32       0             Fa0/0      172.16.15.5
19     No Label      4.4.4.4/32       0             Fa0/0      172.16.15.5
20     No Label      5.5.5.5/32       0             Fa0/0      172.16.15.5
21     No Label      172.16.45.0/24   0             Fa0/0      172.16.15.5
22     No Label      192.1.4.4/32     0             Fa0/0      172.16.15.5
23     No Label      192.1.5.5/32     0             Fa0/0      172.16.15.5
```

This means that there are no labels being learned from R5 now, and this fact means that this extends to labels advertised from R4 to R5. We know based on this fact that label switching will not take place, because there is no label assigned to the prefix specified. This can be illustrated using the traceroute utility on R1.

```
R1#traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.15.5 4 msec 0 msec 0 msec
  2 172.16.45.4 4 msec *  0 msec
```

```
R#
```

Clearly this traffic is being IP routed rather than label switched.

But what would happen if we opted to selectively decide what prefixes we allow R5 to advertise. To illustrate this process we will allow R5 to send labels for 192.1.4.4 and 192.1.5.5. This will be accomplish using an access-list and a modification of the previous command we used to block all labels.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#access-list 1 permit 192.1.4.4
R1(config)#access-list 1 permit 192.1.5.5
R1(config)#!
R1(config)#mpls ldp advertise-labels for 1
R1(config)#end
R1#
```

Now we will see the result via the debug running on R1.

```
R1#show logging | inc binding
tib: 192.1.4.4/32:: learn binding 52 from 192.1.5.5:0
tib: a new binding to be added
tib: 192.1.5.5/32:: learn binding 0 from 192.1.5.5:0
tib: a new binding to be added
R1#
```

We can see the contents of the new LIB on R1 via the **show mpls forwarding-table**.

```
R1#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     No Label      1.1.1.1/32        0             Fa0/0      172.16.15.5
17     No Label      2.2.2.2/32        0             Fa0/0      172.16.15.5
18     No Label      3.3.3.3/32        0             Fa0/0      172.16.15.5
19     No Label      4.4.4.4/32        0             Fa0/0      172.16.15.5
20     No Label      5.5.5.5/32        0             Fa0/0      172.16.15.5
21     No Label      172.16.45.0/24    0             Fa0/0      172.16.15.5
22     52            192.1.4.4/32      0             Fa0/0      172.16.15.5
23     explicit-n    192.1.5.5/32      0             Fa0/0      172.16.15.5
R1#
```

Notice that the two routes we are advertising from R5 are now labeled. We can look at this label closely via the show mpls forwarding-table while specifying the FEC in question and the detail keyword.

```
R1#show mpls forwarding-table 192.1.4.4 detail
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
22     52            192.1.4.4/32      0             Fa0/0      172.16.15.5
        MAC/Encaps=14/18, MRU=1500, Label Stack{52}
        000AB819C6B0000AB86BA3F08847 00034000
        No output feature configured
```

Now that we have a label traffic will again be label switched rather than ip routed. This is best illustrated using the traceroute utility to test the 192.1.4.4 prefix.

```
R1#traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.15.5 [MPLS: Label 52 Exp 0] 4 msec 0 msec 0 msec
  2 172.16.45.4 0 msec *  0 msec
R1#
```

The tool that we just illustrated is called conditional outbound filtering, there is another tool at our disposal that will perform filtering inbound to an LSR. But before we look at that specific mechanism it is important that we make one final observation on R5. Note that before any outbound filter can be performed successfully, we first must tell the router to not send any label at all. We will us the show run command to demonstrate that both the initial **no mpls ldp advertise** command and the filtered command calling access-list 1 are both present in the configuration.

```
R5#show run | inc advert
no mpls ldp advertise-labels
mpls ldp advertise-labels for 1
R5#
```

Now we will employ inbound filtering on R1 for the prefix 192.1.4.4.

```
R1(config)#access-list 1 permit 192.1.5.5
R1(config)#mpls ldp neighbor 192.1.5.5 labels accept 1
R1(config)#end
```

We will look immediately to see if this label is removed from R1 LIB via the **show mpls forwarding-table** command.

```
R1#show mpls forwarding-table 192.1.4.4
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
22     No Label      192.1.4.4/32      0             Fa0/0      172.16.15.5
R1#
```

Now we can see that the traffic will be ip routed rather than label switched once again.

```
R1#traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.15.5 0 msec 4 msec 0 msec
  2 172.16.45.4 4 msec *  0 msec
R1#
```


Authentication

One additional component that we still need to discuss regarding LDP sessions is the mechanics of session authentication. LDP relies on MD5 authentication and must be configured between specific LDP peers. We will establish a protected session between R1 and R5 to demonstrate how this is accomplished and where to look for confirmation and verification.

The most important thing to remember is that authentication is accomplished via the global level mpls ldp neighbor command. We will illustrate this beginning on R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#mpls ldp neighbor 192.1.5.5 password 0 CISCO
R1(config)#
```

Once this configuration has been made on R1 the neighbor relationship between R1 and R5 will not drop immediately. We can verify this with the **show mpls ldp neighbor** command.

```
R1#sh mpls ldp neighbor
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.64611 - 192.1.1.1.646
        State: Oper; Msgs sent/rcvd: 108/100; Downstream
        Up time: 00:46:29
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.5
        Addresses bound to peer LDP Ident:
          172.16.15.5     172.16.45.5     192.1.5.5       1.1.1.1
          2.2.2.2         3.3.3.3         4.4.4.4         5.5.5.5
```

To force the use of these MD5 passwords we will need to apply the **mpls ldp password required** command.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#mpls ldp password required
```

Now we will see that the peering goes down.

```
R1#show mpls ldp neighbor

R1#
```

Now we need to make the configurations on R5 to restore the peering session.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#mpls ldp neighbor 192.1.1.1 password 0 CISCO
R5(config)#
```

We should now see that the peering session is restored.

```
R1#show mpls ldp neighbor
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
```

```
        TCP connection: 192.1.5.5.49746 - 192.1.1.1.646
        State: Oper; Msgs sent/rcvd: 14/6; Downstream
        Up time: 00:00:50
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.15.5
        Addresses bound to peer LDP Ident:
          172.16.15.5     172.16.45.5     192.1.5.5       1.1.1.1
          2.2.2.2         3.3.3.3         4.4.4.4         5.5.5.5
```

Interestingly enough there is no direct indication in this output that indicates that authentication has been enabled between these two peers. There however is a command that will tell us about the basic configuration of the authentication and peering status.

```
R1#show mpls ldp neighbor password current
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.1.1:0
        TCP connection: 192.1.5.5.49746 - 192.1.1.1.646
        Password: required, neighbor, in use
        State: Oper; Msgs sent/rcvd: 20/11
```

Note that we still do not see the password, but we now know to look for authentication. We will repeat this command by removing the authentication on R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#no mpls ldp neighbor 192.1.5.5 password 0 CISCO
```

With the configuration on R1 removed we need to verify that the commands still exist on R5, and then see what the status of the **show mpls ldp neighbor password** command is.

```
R5#show run | sec password
no service password-encryption
mpls ldp neighbor 192.1.1.1 password CISCO
```

The configuration is still in place, but problem is because there is no peering to R1 there will be no information provided by the specialized show command.

```
R5#show mpls ldp neighbor 192.1.1.1 password

R5#
```

However, be sure to either check that logging is being sent to the console or remember to check the logging output going to the buffer. There you will see information regarding the TCP MD5 problem.

```
R5#show logging
Syslog logging: enabled (0 messages dropped, 3 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering disabled)

No Active Message Discriminator.



No Inactive Message Discriminator.
```



```
    Console logging: disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging:  level debugging, 285717 messages logged, xml disabled,
                     filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled
    Persistent logging: disabled

No active filter modules.

ESM: 0 messages dropped

    Trap logging: level informational, 251 message lines logged

Log Buffer (4096 bytes):
 from 192.1.1.1(646) to 192.1.5.5(36842) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(12744) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(12744) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(12744) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(12744) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(65355) (RST)
%TCP-6-BADAUTH: Invalid MD5 digest from 192.1.1.1(646) to 192.1.5.5(65355) (RST)
<output omitted>
```

Note that this output is very helpful. The situation is clearly called out on R5. R1 is sending an invalid value to R5 causes the session to be reset (RST). We can look quickly at R1 to see additional output that is going to the buffer.

```
R1#show logging | inc PWD
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
%LDP-4-PWD: MD5 protection is required for peer 192.1.5.5:0, no password configured
<output omitted>
```

We can clearly see that the authentication is not configured on R1. We can easily remedy this by reapplying the command we removed, but as an added verification we will recheck the log on R1 once the peering is reestablished.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#mpls ldp neighbor 192.1.5.5 password 0 CISCO
R1(config)#end
R1#show logging | inc 192.1.5.5
%LDP-5-NBRCHG: LDP Neighbor 192.1.5.5:0 (1) is UP
tagcon: 192.1.5.5:0: 172.16.15.5 added to addr<->ldp ident map
```

```
tagcon: 192.1.5.5:0: 172.16.45.5 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 192.1.5.5 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 1.1.1.1 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 2.2.2.2 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 3.3.3.3 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 4.4.4.4 added to addr<->ldp ident map
tagcon: 192.1.5.5:0: 5.5.5.5 added to addr<->ldp ident map
```

Note that there is no mention of authentication in the output associated with the peering, we just see the link come up and the address to label identifier map sequence initiate.

As a final verification we can traceroute to 192.1.4.4 after we remove our inbound filter from R1 and the outbound filter from R5.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#no mpls ldp neighbor 192.1.5.5 labels accept 1
R1(config)#end

R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#mpls ldp advertise-labels
R5(config)#no mpls ldp advertise-labels for 1
R5(config)#
```

To insure that the filters are cleared and all labels can be exchanged we will clear our LDP neighbors on R5.

```
R5#clear mpls ldp neighbor *
R5#
```

Finally we will conduct the traceroute.

```
R1# traceroute 192.1.4.4

Type escape sequence to abort.
Tracing the route to 192.1.4.4

  1 172.16.15.5 [MPLS: Label 52 Exp 0] 4 msec 0 msec 4 msec
  2 172.16.45.4 0 msec *  0 msec
R1#
```

IGP Synchronization

We have discussed normal operations regarding how LDP allows LSRs to rapidly recover from changes associated with next-hop reachability. But what happens when a give LDP session is broken along a given link? This creates a problem where the ip routing protocol still has the prefix as a valid outbound route; this means that packets can still be forwarded out the link. The situation that is transpiring is due to the IGP behavior of installing the best path in the routing table for any prefix. In this situation, packets destined to prefixes where the next hop is reachable out of a link where the LDP session is broken are sent unlabeled. This means that Layer 3 VPNs may not operate as expected, because data packets may be discarded thus resulting in traffic black-holes.

IGP synchronization currently only extends to a special features capability located in OPSF. The ability is activated under the router process itself. Once the **mpls ldp sync** command has been applied the links advertised by OSPF that are associated with MPLS enabled interfaces will be advertised with a max cost until such time that the LDP session establishment process is completed. Additionally, hellos will not be sent on a given link when the LDP session is down, or until the configured synchronization timer expires. An OSPF peering adjacency will be formed if LDP detects that the link in question is the only link that can be used to reach a given neighbors LDP Router-id. MPLS LDP synchronization can be disabled at the interface level using the **no mpls ldp igp sync** command. The following parameters should be noted regarding the holddown timers we mentioned earlier:

- **(G) mpls ldp igp sync holddown <msec>**
    - If the hold time expires then the OSPF session in established
    - this will happen even if OSPF is not synched with LDP
    - However, the link is still announced with the max cost of 65536

We can observe how this configuration is setup by deploying the it under the routing process on R1 in our topology.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#mpls ldp sync
R1(config-router)#end
```

This configuration can best be verified using either of the follow show commands:

```
R1#show ip ospf mpls ldp interface
Loopback0
  Process ID 1, Area 0
  LDP is not configured through LDP autoconfig
  LDP-IGP Synchronization : Not required
  Holddown timer is disabled
  Interface is up
FastEthernet0/0
  Process ID 1, Area 0
  LDP is not configured through LDP autoconfig
  LDP-IGP Synchronization : Required
  Holddown timer is not configured
  Interface is up
```

or

```
R1#show mpls ldp igp sync
    FastEthernet0/0:
        LDP configured; LDP-IGP Synchronization enabled.
        Sync status: sync achieved; peer reachable.
        Sync delay time: 0 seconds (0 seconds left)
        IGP holddown time: infinite.
        Peer LDP Ident: 192.1.5.5:0
        IGP enabled: OSPF 1
```


Now we can break the LDP peering between R1 and R5 but first we need to look at the OSPF cost associated the router LSA for R1s 192.1.1.1 interface on R5.

```
R5#show ip ospf database router 192.1.1.1

            OSPF Router with ID (192.1.5.5) (Process ID 1)

                Router Link States (Area 0)

  LS age: 73
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 192.1.1.1
  Advertising Router: 192.1.1.1
  LS Seq Number: 80000045
  Checksum: 0x7A7E
  Length: 48
  Number of Links: 2

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 192.1.1.1
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 172.16.15.1
     (Link Data) Router Interface address: 172.16.15.1
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

R5#
```

Now we will break the LDP neighborship by removing the MD5 authentication from R5:

```
R5#conf t
R5(config)#no mpls ldp neighbor 192.1.1.1 password CISCO
R5(config)#do show mpls ldp nei
R5(config)#end
```

We see that the LDP session has ended, but is the OSPF adjacency still up?

```
R5#show ip ospf nei

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.1.4.4         1   FULL/BDR        00:00:37    172.16.45.4     FastEthernet0/1
192.1.1.1         1   FULL/DR         00:00:31    172.16.15.1     FastEthernet0/0
```

The answer is yes but what cost is going to be reported for the router LSA for 192.1.1.1 now?

```
R5#show ip ospf database router 192.1.1.1

            OSPF Router with ID (192.1.5.5) (Process ID 1)
```

```
                Router Link States (Area 0)

  LS age: 150
  Options: (No TOS-capability, DC)
  LS Type: Router Links
  Link State ID: 192.1.1.1
  Advertising Router: 192.1.1.1
  LS Seq Number: 80000046
  Checksum: 0x5A9E
  Length: 48
  Number of Links: 2

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 192.1.1.1
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metrics: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 172.16.15.1
     (Link Data) Router Interface address: 172.16.15.1
      Number of TOS metrics: 0
       TOS 0 Metrics: 65535

R5#
```

Auto Configuration

One other enhancement exits in OSPF that allows us to use the characteristics of the routing protocol itself to assign what interfaces are going to operate via mpls. Under the routing process just like in the previous section we have the ability to specify that all OSPF enabled interfaces or just those in a given area participate in MPLS. This is done using the **mpls ldp autoconfig** command. We will look at this process on R1 after a fresh restart where all previous configurations have been removed.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#mpls ldp autoconfig area 0
%LDP must be enabled globally by means of global 'mpls ip' command
 before configuring the router mode 'mpls ldp autoconfig ...' command
R1(config-router)#
```

Observe that the process itself notifies us that the global mpls ip command must be configured before we can use the **mpls ldp autoconfig** command. Now we will us the proper order of operations.

```
R1(config)#mpls ip
R1(config)#router ospf 1
R1(config-router)#mpls ldp autoconfig area 0
R1(config-router)#end
R1#
```

We can see what interfaces are running mpls by using the **show mpls interface** command.

```
R1#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/0        Yes (ldp)     No       No  No     Yes
```

Observe that the loopback interface is not participating in MPLS according to this output. This is because MPLS is not supported on loopback interfaces. We can see this clearly if we try to manually configure it under loopback 0 on R1.

```
R1(config-if)#mpls ip
% MPLS not supported on interface Loopback0
R1(config-if)#
```

Common Issues with Label Distribution

Label Distribution has a number of issues that can surface when deployed. The most common problems relate to the exchange of the essential control and forwarding plane information. The control plane establishment in LDP is very streamlined. For simplicity in troubleshooting common issues encountered during Label Distribution, we identify four categories of problems: LDP session startup Issues, No Label Allocation, or Allocated labels are not distributed between peers.

LDP Session Startup Issues

There are a relatively small number of issues that can result in a failure to see session establishment take place between LDP peers. These issues can most notably be isolated if we find the answers to the following questions:

- Do all expected neighbors appear?
  - ***show mpls ldp discovery***
- Is MPLS on all needed interfaces?
  - ***show mpls interface***
- Do we have all expected direct adjacencies?
  - ***traceroute*** (look for more than 1 hop)
- Are all neighbors using the same protocol (LDP/TDP)?
  - ***show mpls interface detail***
- Are there any ACLs or Filters blocking ports 711 or 646?
  - ***show ip int | inc line|list***
- Lastly, verify reachability between loopbacks
  - ***ping***

No Label Allocation

We have spent exhaustive amount of time up to this point to learn what processes LDP is responsible for. We know that LDP relies on the underlying IGP protocol to provide a list of prefixes that will have labels bound to them. Before we can hope advertise or exchange these labels we must know that they are being created in the first place. Issues evolving a failure to create a label locally can most notably be isolated by if we find the answers to the following questions:

- Are labels being assigned to local prefixes?
  - **show mpls forwarding-table**
- Ensure CEF is enabled globally or on interfaces?
  - **show cef interface**

Allocated Labels Are Not Distributed

Once we have determined that LDP creating labels as we would expect it to, the next operational role for the protocol is to communicate these label to ip bindings to all adjacent or non-adjacent LDP peers. For the purposes of the Quickfire Troubleshooting process there are only a handful of issues that can prevent the successful exchange of this information, and these causes can best be isolated by finding the answers to following questions:

- Does adjacent LSRs display received labels?
    - **show mpls ldp bindings**
- Is conditional label advertising configured?
    - **show run | inc advert**

Label Distribution Sample Troubleshooting Scenarios

This section provides a detailed look at how to best approach troubleshooting some of the common issues discussed in previous section. It includes coverage of a methodology for identification, isolation, and remediation of faults in the label distribution operational process. The intent here is to hone and develop troubleshooting skills tailored to first identify if a problem exists, and then how to begin isolating the cause of the fault in the most efficient manner possible. Figure 3-2 illustrates the topology used to explore this topic.

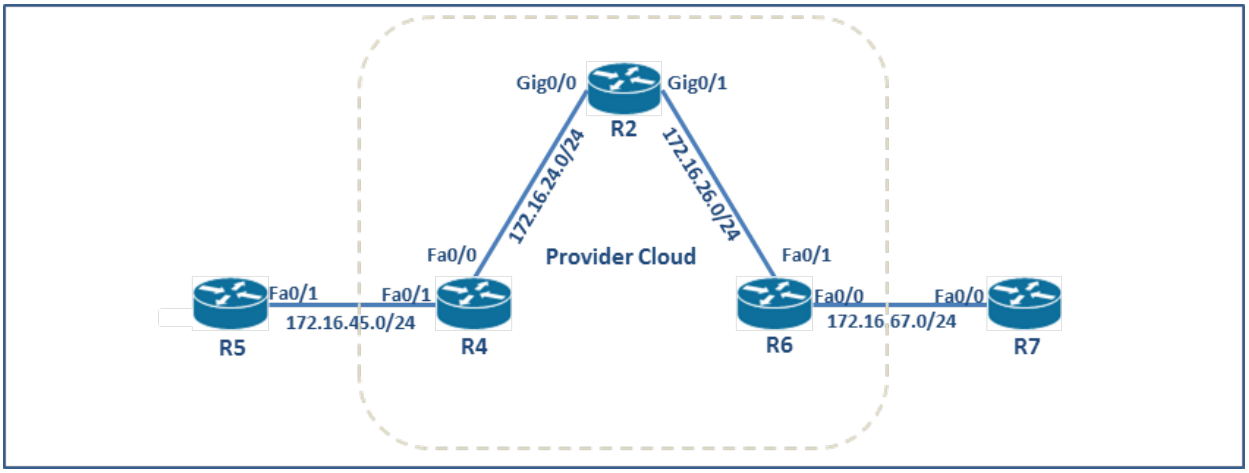


Figure 3-2: A Sample Label Distribution Topology

In the ***Common Issues with Label Distribution*** section, three primary types of problems were identified: LDP session startup Issues, no label allocation, or allocated labels are not distributed between peers. This section explores these three categories of failure, by directing our attention to the commands necessary to verify a problem, isolate it and remediate it.

LDP Session Startup Issues

Setting the stage: The LDP sessions between R4, R2 and R6 are not forming as expected as illustrated by the **show mpls ldp neighbor** command on executed on R2.

```
R2#show mpls ldp neighbor

R2#
```

Step One: Based on the current configuration we will need to verify the configuration based on a hop-by-hop analysis between R4 and R6.

On R4 we will first look at the output of the **show mpls ldp neighbor** command:

```
R4#show mpls ldp neighbor

R4#
```

We now need to check to see if MPLS is running on the appropriate interfaces.

Next we need to verify that MPLS is running on the appropriate interfaces. This is accomplished via the **show mpls ldp interfaces** command.

```
R4#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/0        Yes (tdp)     No       No  No     Yes
R4#
```

We see immediately that the interface is operational, but observe that we are running TDP rather than LDP. This is not an issue of we are running TDP on R2. We can verify that by using the same command on R2.

```
R2#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
GigabitEthernet0/0     Yes (ldp)     No       No  No     Yes
```

We see that R2 is running LDP on the GigabitEthernet0/0 interface. We will chose to change the label protocol on R4 to LDP.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#mpls label protocol ldp
R4(config)#end
R4#
%LDP-5-NBRCHG: LDP Neighbor 192.1.2.2:0 (1) is UP
```

Note that the peering comes up immediately once the change has been made.

Step Two: Verify that the link is up and operational by repeating the **show mpls ldp neighbor** command on R2.

```
R2#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.4.4.20934 - 172.16.24.2.646
        State: Oper; Msgs sent/rcvd: 15/15; Downstream
        Up time: 00:02:52
        LDP discovery sources:
          GigabitEthernet0/0, Src IP addr: 172.16.24.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4
R2#
```

We see that the link is up between R2 and R4, however we should also see a connection between the R2 and R6. We will use the **show mpls interfaces** command to verify that MPLS is running on GigabitEthernet0/1.

```
R2#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
GigabitEthernet0/0     Yes (ldp)     No       No  No     Yes
R2#
```


The command tells us that GigabitEthernet0/1 is not running LDP, we can see what is happening on GigabitEthernet0/1.

```
R2#show run interface GigabitEthernet0/1
Building configuration...

Current configuration : 116 bytes
!
interface GigabitEthernet0/1
 ip address 172.16.26.2 255.255.255.0
 duplex auto
 speed auto
 media-type rj45
end

R2#
```

We see that mpls is not running on this interface, and can make the changes necessary by adding the command **mpls ip**.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface GigabitEthernet0/1
R2(config-if)#mpls ip
R2(config-if)#end
R2#
```

We can verify that this enables MPLS on the GigabitEthernet0/1 has worked by using the **show mpls interface** command.

```
R2#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
GigabitEthernet0/0     Yes (ldp)     No       No  No     Yes
GigabitEthernet0/1     Yes (ldp)     No       No  No     Yes
R2#
```

However, even after waiting several minutes the LDP peering relations does not form between R2 and R6. We need to verify that R6 is running MPLS.

```
R6#show mpls interfaces
Interface              IP            Tunnel   BGP Static Operational
FastEthernet0/1        Yes (ldp)     No       No  No     Yes
R6#
```

MPLS is running LDP on interface FastEthernet0/1 facing R2. The current situation means that some configuration is stopping the peering relationship from being established. We will verify whether or not there is any information being exchanged between R2 and R6 at any level of the peering establishment: either transport connection establishment or session establishment. We will do this via the **show mpls ldp discovery detail** command.

```
R6#show mpls ldp discovery detail
 Local LDP Identifier:
```



```
    192.1.6.6:0
    Discovery Sources:
    Interfaces:
        FastEthernet0/1 (ldp): xmit/recv
            Enabled: Interface config
            Hello interval: 5000 ms; Transport IP addr: 192.1.6.6
            LDP Id: 192.1.2.2:0
              Src IP addr: 172.16.26.2; Transport IP addr: 192.1.2.2
              Hold time: 15 sec; Proposed local/peer: 15/15 sec
              Reachable via 192.1.2.2/32
              Password: not required, neighbor, in use
R6#
```

We see that information is being exchanged but no neighbor relationship is being formed. We need to take the next logical course of action to verify if any filters are stopping the formation of the peers. This is best accomplished with the **show ip int | inc line|list** command.

```
R6#show ip int | inc line|list
FastEthernet0/0 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
FastEthernet0/1 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
Serial0/1/0 is administratively down, line protocol is down
Serial0/2/0 is administratively down, line protocol is down
Serial0/2/1 is administratively down, line protocol is down
Loopback0 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
R6#
```

No filters are present on R6. What about R2?

```
R2#show ip int | inc line|list
GigabitEthernet0/0 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
GigabitEthernet0/1 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
Serial0/1/0 is administratively down, line protocol is down
Serial0/2/0 is administratively down, line protocol is down
Loopback0 is up, line protocol is up
  Outgoing access list is not set
  Inbound  access list is not set
R2#
```

This leaves us with a possible authentication issue. We have two methods to verify this first we can check the configuration for any password commands. This is best accomplished via the **show run** command.

```
R2#show run | inc password
```



```
no service password-encryption
R2#
```

R2 is not running any authentication. Now we will look at R6.

```
R6#show run | inc password
no service password-encryption
mpls ldp neighbor 192.1.2.2 password CISCO
R6#
```

We mentioned that we have two ways to test this issue. The **show run | inc password** is the most direct, but we can also rely on the console messages to provide information about misconfigured MPLS authentication. It is important to note that these messages are only going to appear where the authentication configuration has been made. We can see this on R6 using the **show logging** command.

```
R6#show log | inc MD5
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(62928)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(62928)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(62928)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(62928)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(30789)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(39385)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(39385)
%TCP-6-BADAUTH: No MD5 digest from 192.1.2.2(646) to 192.1.6.6(39385)
<output omitted>
```

To drive home the point that this information is not going to appear on the device where there is no authentication for MPLS configured we will us the same command.

```
R2#show log | inc MD5
R2#
```

Be mindful that this issue will and can be further obfuscated by the logging in the Troubleshooting section of the lab being made to the buffer rather than the console.

To correct this issue we will place a matching mpls authentication on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#mpls ldp neighbor 192.1.6.6 password 0 CISCO
R2(config)#end
%LDP-5-NBRCHG: LDP Neighbor 192.1.6.6:0 (2) is UP
R2#
```

We should see the peering come up almost immediately. This can also be verified with the **show mpls ldp neighbor** command.

```
R2#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.4.4.20934 - 172.16.24.2.646
        State: Oper; Msgs sent/rcvd: 59/60; Downstream
        Up time: 00:41:55
```



```
        LDP discovery sources:
          GigabitEthernet0/0, Src IP addr: 172.16.24.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4
    Peer LDP Ident: 192.1.6.6:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.6.6.19779 - 192.1.2.2.646
        State: Oper; Msgs sent/rcvd: 13/13; Downstream
        Up time: 00:01:28
        LDP discovery sources:
          GigabitEthernet0/1, Src IP addr: 172.16.26.6
        Addresses bound to peer LDP Ident:
          172.16.67.6     172.16.26.6     192.1.6.6
```

We now see that we have two peering relationships that are up and operational. In addition we see that these peers are exchanging information regarding IP to label bindings.

Step Three: Next we will validate the fact that label switching is taking place by using a traceroute from R4 to R6's loopback 0 interfaces.

```
R4#traceroute 192.1.6.6 source loopback 0

Type escape sequence to abort.
Tracing the route to 192.1.6.6

  1 172.16.24.2 [MPLS: Label 20 Exp 0] 0 msec 0 msec 4 msec
  2 172.16.26.6 0 msec *  0 msec
```

We see that the traffic is label switched. It is this process that we will examine closer in the upcoming section.

No Label Allocation

Setting the stage: When we attempt to traceroute from R4's loopback 0 interface to R7's loopback 0 interface the utility is unsuccessful.

```
R4#traceroute 192.1.7.7 source lo 0

Type escape sequence to abort.
Tracing the route to 192.1.7.7

  1  *  *  *
  2  *  *  *
  3  *  *  *
  4  *  *  *
  5  *  *  *
  6  *  *  *
  7  *  *  *
  8  *  *  *
  9  *  *  *
 10  *  *  *
 11  *  *  *
 12  *  *  *
 13  *  *  *
 14  *  *  *
```

```
 15  *  *  *
 16  *  *  *
 17  *  *  *
 18  *  *  *
 19  *  *  *
 20  *  *  *
 21  *  *  *
 22  *  *  *
 23  *  *  *
 24  *  *  *
 25  *  *  *
 26  *  *  *
 27  *  *  *
 28  *  *  *
 29  *  *  *
 30  *  *  *
R4#
```

Step One: In this situation we are going to verify from again on a hop-by-hop basis.

On R4 we will look for the presence of LDP peers with **show mpls ldp neighbor**:

```
R4#show mpls ldp neighbor
    Peer LDP Ident: 192.1.2.2:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 172.16.24.2.646 - 192.1.4.4.25455
        State: Oper; Msgs sent/rcvd: 65/65; Downstream
        Up time: 00:45:46
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.24.2
        Addresses bound to peer LDP Ident:
          172.16.24.2     172.16.26.2     192.1.2.2
R4#
```

We see that R4 is learning the following prefixes from R2: 172.16.24.2, 172.16.26.2 and 192.1.2.2 from R2. What labels are being assigned to these prefixes.

```
R4#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     Pop Label     172.16.26.0/24   0             Fa0/0       172.16.24.2
17     17            172.16.67.0/24   0             Fa0/0       172.16.24.2
18     No Label      192.1.2.2/32     0             Fa0/0       172.16.24.2
19     No Label      192.1.5.5/32     0             Fa0/1       172.16.45.5
20     20            192.1.6.6/32     0             Fa0/0       172.16.24.2
21     21            192.1.7.7/32     0             Fa0/0       172.16.24.2
R4#
```

Note that the prefix 172.16.24.0/24 is connected so there will be no labels assigned for it specifically. Additionally, we see that the two other prefixes have the labels 16 and 18 assigned.

Step Two: We need to check the status of R2 by using the same series of commands.

On R2 we will look for the presence of LDP peers with **show mpls ldp neighbor**:

```
R2#show mpls ldp neighbor
    Peer LDP Ident: 192.1.6.6:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.6.6.51014 - 192.1.2.2.646
        State: Oper; Msgs sent/rcvd: 72/72; Downstream
        Up time: 00:52:18
        LDP discovery sources:
          GigabitEthernet0/1, Src IP addr: 172.16.26.6
        Addresses bound to peer LDP Ident:
          172.16.67.6     172.16.26.6     192.1.6.6
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.4.4.25455 - 172.16.24.2.646
        State: Oper; Msgs sent/rcvd: 72/72; Downstream
        Up time: 00:52:16
        LDP discovery sources:
          GigabitEthernet0/0, Src IP addr: 172.16.24.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4
```

We see that R2 is learning the following prefixes from R4: 172.16.24.4, 172.16.45.4 and 192.1.4.4 from R4. What labels are being assigned to these prefixes.

```
R2#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
R2#
```

We see that no labels are being assigned at all to any prefix. At this juncture we know that the routing table has routes, as evidenced by the she output of the **show ip route** command.

```
R2#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 4 subnets
O       172.16.45.0 [110/2] via 172.16.24.4, 00:54:52, GigabitEthernet0/0
C       172.16.24.0 is directly connected, GigabitEthernet0/0
C       172.16.26.0 is directly connected, GigabitEthernet0/1
O       172.16.67.0 [110/2] via 172.16.26.6, 00:54:52, GigabitEthernet0/1
     192.1.4.0/32 is subnetted, 1 subnets
O       192.1.4.4 [110/2] via 172.16.24.4, 00:54:52, GigabitEthernet0/0
     192.1.5.0/32 is subnetted, 1 subnets
O       192.1.5.5 [110/3] via 172.16.24.4, 00:54:53, GigabitEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O       192.1.6.6 [110/2] via 172.16.26.6, 00:54:53, GigabitEthernet0/1
     192.1.7.0/32 is subnetted, 1 subnets
O       192.1.7.7 [110/3] via 172.16.26.6, 00:54:53, GigabitEthernet0/1
C    192.1.2.0/24 is directly connected, Loopback0
```


Something is stopping this the LDP process from allocating labels for each of these prefixes. We know that this process requires the information from the routing table (which we have just looked at) and the forwarding information base created by CEF. This is accomplished via the **show ip cef** command.

```
R2#show ip cef
%IPv4 CEF not running
R2#
```

Step Three: We have isolated the issue by determining that CEF is disabled. To correct this we will enable CEF on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip cef
R2(config)#
```

We can now see that R2 will immediately allocate labels.

```
R2#show ip cef
Prefix              Next Hop            Interface
0.0.0.0/0           no route
0.0.0.0/8           drop
0.0.0.0/32          receive
127.0.0.0/8         drop
172.16.24.0/24      attached            GigabitEthernet0/0
172.16.24.0/32      receive             GigabitEthernet0/0
172.16.24.2/32      receive             GigabitEthernet0/0
172.16.24.4/32      attached            GigabitEthernet0/0
172.16.24.255/32    receive             GigabitEthernet0/0
172.16.26.0/24      attached            GigabitEthernet0/1
172.16.26.0/32      receive             GigabitEthernet0/1
172.16.26.2/32      receive             GigabitEthernet0/1
172.16.26.6/32      attached            GigabitEthernet0/1
172.16.26.255/32    receive             GigabitEthernet0/1
172.16.45.0/24      172.16.24.4         GigabitEthernet0/0
172.16.67.0/24      172.16.26.6         GigabitEthernet0/1
192.1.2.0/24        attached            Loopback0
192.1.2.0/32        receive             Loopback0
192.1.2.2/32        receive             Loopback0
192.1.2.255/32      receive             Loopback0
192.1.4.4/32        172.16.24.4         GigabitEthernet0/0
192.1.5.5/32        172.16.24.4         GigabitEthernet0/0
Prefix              Next Hop            Interface
192.1.6.6/32        172.16.26.6         GigabitEthernet0/1
192.1.7.7/32        172.16.26.6         GigabitEthernet0/1
224.0.0.0/4         drop
224.0.0.0/24        receive
240.0.0.0/4         drop
255.255.255.255/32  receive
```

We now need to verify if the traceroute works on R4.


```
R4#traceroute 192.1.7.7

Type escape sequence to abort.
Tracing the route to 192.1.7.7

  1 172.16.24.2 [MPLS: Label 21 Exp 0] 0 msec 0 msec 0 msec
  2 172.16.26.6 [MPLS: Label 21 Exp 0] 4 msec 0 msec 4 msec
  3 172.16.67.7 0 msec *  0 msec
R4#
```

The issue has been remediated. Now we see that the traffic is being label switched. The main reason for a LSR to not allocate labels will normally be associated with IP CEF. Once we know that LDP is allocating labels we next need to verify that labels are being exchanged.

Allocated Labels Are Not Distributed

Setting the Stage: Traffic destined to 192.1.7.7 from R4 is not being label switched across the entire LSP.

```
R4#traceroute 192.1.7.7

Type escape sequence to abort.
Tracing the route to 192.1.7.7

  1 172.16.24.2 4 msec 0 msec 0 msec
  2 172.16.26.6 [MPLS: Label 21 Exp 0] 4 msec 0 msec 4 msec
  3 172.16.67.7 0 msec *  0 msec
```

Step One: In this situation we are going to verify again on a hop-by-hop basis.

On R4 we will look for the presence of LDP peers with **show mpls ldp neighbor**:

```
R4#show mpls ldp neighbor
    Peer LDP Ident: 192.1.2.2:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 172.16.24.2.646 - 192.1.4.4.25455
        State: Oper; Msgs sent/rcvd: 65/65; Downstream
        Up time: 00:45:46
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.24.2
        Addresses bound to peer LDP Ident:
          172.16.24.2     172.16.26.2     192.1.2.2
R4#
```

We see that R4 is learning the following prefixes from R2: 172.16.24.2, 172.16.26.2 and 192.1.2.2 from R2. What labels are being assigned to these prefixes.

```
R4#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     No Label      172.16.26.0/24   0             Fa0/0      172.16.24.2
17     No Label      172.16.67.0/24   0             Fa0/0      172.16.24.2
18     No Label      192.1.2.2/32     0             Fa0/0      172.16.24.2
19     No Label      192.1.5.5/32     0             Fa0/1      172.16.45.5
20     No Label      192.1.6.6/32     0             Fa0/0      172.16.24.2
21     No Label      192.1.7.7/32     0             Fa0/0      172.16.24.2
```

```
R4#
```

No routes being learned are being assigned Outgoing labels. This means that R2 is advertising no labels. But the question is, "is R2 allocating labels".

Step Two: On R2 we will need to verify the contents of the Label Information Base. If there are labels being locally assigned we will need to see if they are being advertised.

```
R2#show mpls forwarding-table interface GigabitEthernet0/0
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     172.16.45.0/24    0             Gi0/0      172.16.24.4
18     No Label      192.1.4.4/32      0             Gi0/0      172.16.24.4
19     19            192.1.5.5/32      0             Gi0/0      172.16.24.4
R2#
```

We see that local labels are being allocated, now we need to see if they are being advertised. We can do this using the **debug mpls ldp advertisements all** command.

```
R2#debug mpls ldp advertisements all
LDP label and address advertisements debugging is on for all routing table
```

Now we will clear the LDP neighbor relationships on R2.

```
R2#clear mpls ldp neighbor *
R2#
```

This will force the process to initiate label exchange.

```
R2#
%LDP-5-CLEAR_NBRS: Clear LDP neighbors (*) by console
%LDP-5-NBRCHG: LDP Neighbor 192.1.6.6:0 (3) is DOWN (User cleared session manually)
%LDP-5-NBRCHG: LDP Neighbor 192.1.4.4:0 (1) is DOWN (User cleared session manually)
R2#
tagcon: (default) Assign peer id; 192.1.4.4:0: id 1
%LDP-5-NBRCHG: LDP Neighbor 192.1.4.4:0 (2) is UP
R2#
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.24.2(default)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.26.2(default)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.2.2(default)
tagcon: (default) Assign peer id; 192.1.6.6:0: id 3
%LDP-5-NBRCHG: LDP Neighbor 192.1.6.6:0 (4) is UP
R2#
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 172.16.24.2(default)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 172.16.26.2(default)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.2.2(default)
tagcon: (default) Deassign peer id; 192.1.4.4:0: id 0
tagcon: (default) Deassign peer id; 192.1.6.6:0: id 2
R2#
```


We immediately notice that Labels are not attached to these updates between 192.1.4.4 and 192.1.6.6. We have isolated the issue, now we need to discover why. The primary thing that can stop this type of data exchange is a label filter, and it is most easily found using the **show run** command.

```
R2#show run | inc advert
no mpls ldp advertise-labels
R2#
```

We see that the **no mpls ldp advertise-labels** command.

Step Three: We observe that this command is stopping the exchange of labels, but now we face a situation that forces us to look at the parameters of the lab. In a situation where we have been instructed not to remove configuration commands, we would need to create an access-list and select which labels we wish to advertise. For the purposes of this discussion we will create an access-list that matches all ip routes and apply the **mpls ldp advertise-labels** command referencing that access-list.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#access-list 1 permit any
R2(config)#mpls ldp advertise-labels for 1
R2(config)#exit
R2#
```

Once this configuration is made the label exchange will immediately take place.

```
R2#
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.24.0/24(default), label 3
(imp-null) (#48)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.26.0/24(default), label 3
(imp-null) (#49)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.45.0/24(default), label 16
(#50)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 172.16.67.0/24(default), label 17
(#51)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.2.0/24(default), label 3
(imp-null) (#52)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.4.4/32(default), label 18
(#55)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.5.5/32(default), label 19
(#56)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.6.6/32(default), label 20
(#58)
tagcon: peer 192.1.4.4:0 (pp 0x6946C424): advertise 192.1.7.7/32(default), label 21
(#59)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 172.16.24.0/24(default), label 3
(imp-null) (#48)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5
R2#A8): advertise 172.16.26.0/24(default), label 3 (imp-null) (#49)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 172.16.45.0/24(default), label 16
(#50)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 172.16.67.0/24(default), label 17
(#51)
```



```
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.2.0/24(default), label 3
(imp-null) (#52)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.4.4/32(default), label 18
(#55)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.5.5/32(default), label 19
(#56)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.6.6/32(default), label 20
(#58)
tagcon: peer 192.1.6.6:0 (pp 0x6946C5A8): advertise 192.1.7.7/32(default), label 21
(#59)
R2#
```

We can see that R2 now has prefixes with labels assigned.

```
R4#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     172.16.26.0/24    0             Fa0/0      172.16.24.2
17     17            172.16.67.0/24    0             Fa0/0      172.16.24.2
18     No Label      192.1.2.2/32      0             Fa0/0      172.16.24.2
19     No Label      192.1.5.5/32      0             Fa0/1      172.16.45.5
20     20            192.1.6.6/32      0             Fa0/0      172.16.24.2
21     21            192.1.7.7/32      0             Fa0/0      172.16.24.2
R4#
```

And that traffic is now label switched across the entire label switched path.

```
R4#traceroute 192.1.7.7

Type escape sequence to abort.
Tracing the route to 192.1.7.7

  1 172.16.24.2 [MPLS: Label 21 Exp 0] 0 msec 0 msec 0 msec
  2 172.16.26.6 [MPLS: Label 21 Exp 0] 4 msec 0 msec 4 msec
  3 172.16.67.7 0 msec *  0 msec
R4#
```

The problem has been successfully remediated.

Chapter Challenge: Label Distribution Sample Trouble Tickets

The following section includes two sample Trouble Tickets designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. These Trouble Tickets were designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configurations provided in the file ***MPLS-CH3-LDP-TT-INITIAL.txt***. Keep in mind these sample Trouble Tickets were also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 3-3 below:

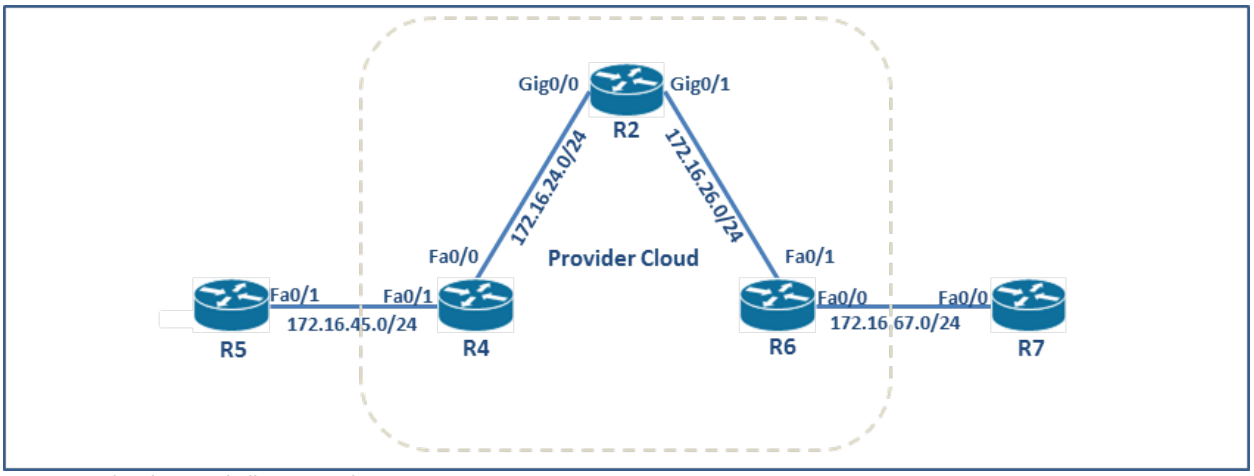


Figure 3-3: The Chapter Challenge Topology

<TROUBLE TICKETS FORTHCOMING>

Chapter 4: MPLS Layer 3 VPN

MPLS-based Layer 3 VPNs are based on a peer model that enables the provider and the customer to exchange Layer 3 routing information. The provider relays the data between the customer sites without direct customer involvement.

Introduction to Layer 3 VPNs

At this point in our exploration of MPLS, Labels, and label assignment and exchange we now find ourselves at the very edge of what will be the most foundationally critical aspect of the technology. Up to this point we have discussed all the moving parts and constituent elements that allow MPLS to operate but now with the consideration of Layer 3 VPN technologies and MPLS we are delving into the practical application of these independent protocols and mechanisms in to a unified and granular tool for the exchange of information between remote sites via the logical construct we will be discussing in-depth in this chapter. We will begin our high level overview of a MPLS Layer 3 VPN, by outlining a working definition of the concept.

What is an MPLS Layer 3 VPN

A L3VPN is best described as a virtual private network that utilizes layer virtual routing and forwarding instances to partition routing tables such that they support separate customers that wish to use the VPN service. These customers peer with the service provider router and the two typically exchange routes. These exchanged prefixes are then placed in the customer's specific routing table and through the use of MP-BGP the provider edge devices relay that data between customer sites without direct customer involvement. To summarize the basic operation of the Layer 3 VPN we will reiterate the components we have discussed in previous chapters and re-orient their concepts such that we identify what roles the play in the L3VPN.

- **Provider (P) router** — A router in the core of the provider network. P routers run MPLS switching and do not attach VPN labels (an MPLS label in each route assigned by the PE router) to routed packets. P routers forward packets based on the Label Distribution Protocol (LDP).

- **Provider edge (PE) router** — A router that attaches the VPN label to incoming packets that are based on the interface or sub-interface on which they are received. A PE router attaches directly to a CE router.

- **Customer edge (CE) router** — An edge router on the network of the provider that connects to the PE router on the network. A CE router must interface with a PE router.

MPLS Layer 3 VPN functionality is enabled at the edge of an MPLS network such that the PE router is responsible for exchanging routing updates with the CE router. The PE router then translates these updates into VPNv4 routes. Once this translation process is completed the PE will then exchange these newly created Layer 3 VPN router with the other PE routers through the use of Multiprotocol Border Gateway Protocol (MP-BGP).

The Expanded Role of VRF Instances

Up to this point we have only discussed VRF's in the context of the their operation and definition. Now when employed in the context of a L3VPN we now see how these logical instances and there assigned values take effect. For our discussion at this point we need to understand that each Layer 3 VPN will be associated with one or more VRF instance. A VRF, in the context of this explanation, will define the customers membership to a particular VPN. That VPN assignment will be made on the PE device and the

customer equipment and their associated networks have no knowledge of this VRF to VPN association. Suffice it to say that for our discussion three values will always be associated with a given VRF:

- Each Layer 3 VPN will be associated with one or more virtual routing and forwarding (VRF) instance.

- In an MPLS Layer 3 VPN, a VRF defines the VPN membership of a customer site that is attached to a given PE router.

- VRFs will consist of the following components: a discrete ip routing table, a group of interfaces that are assigned to forward for the VRF, and a set of parameters and protocols that control the information that is placed in the routing table previously mentioned.

We have to keep in mind that we are not describing a mandatory one-to-one relationship between customer sites and VPNs, because a site can be a member of multiple VPNs, but normally a single CE router can only associate with one VRF. With these items pointed out we next need to look at what composes a Layer 3 VPN before we begin to explore the concepts at the command line interface.

MPLS Layer 3 VPN Components

We will describe a Layer 3 VPN has having four primary components that are basic to its operation.

1. **VPN route target communities**—A VPN route target community is a list of all members of a Layer 3 VPN community. You must configure the VPN route targets for each Layer 3 VPN community member.

2. **Multiprotocol BGP peering of VPN community PE routers**—Multiprotocol BGP propagates VRF reachability information to all members of a VPN community. You must configure Multiprotocol BGP peering in all PE routers within a VPN community.

3. **MPLS forwarding**—MPLS transports all traffic between all VPN community members across a VPN enterprise or service provider network.

Implementation of MPLS Layer 3 VPN

So far in this chapter we have explored the elements that comprise an MPLS Layer 3 VPN, but now we will begin to explore these components and their operation at the command line interface. We will begin this exploration using the following topology, and in that topology we will walk through each of the concepts we have mentioned in the previous sections. We will being with the roles as they apply to the L3VPNs and the commands needed to implement those roles.

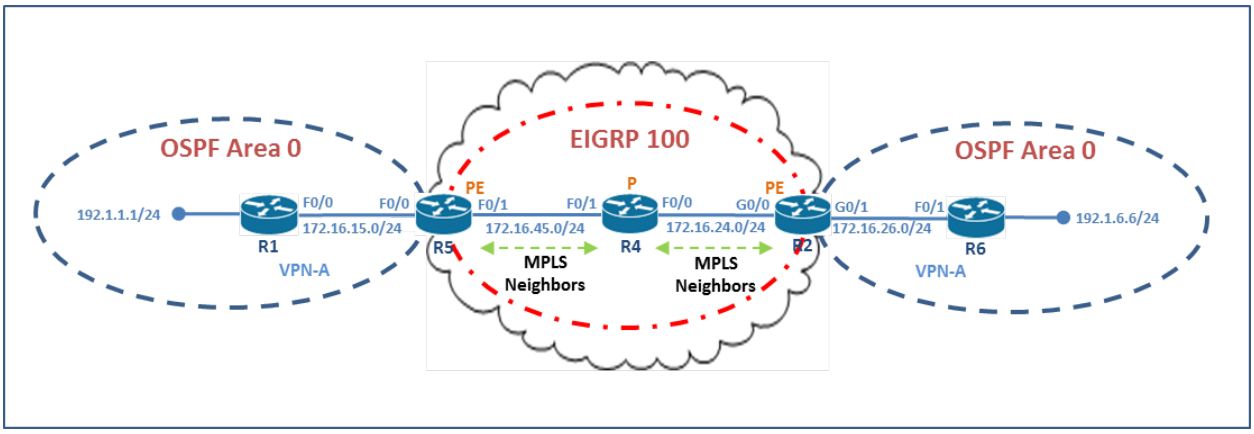


Figure 4-1: MPLS Layer 3 VPN Topology

Implement VPN route target communities

In this topology we see that the routers R5, R4 and R2 are the are in the provider cloud. In the drawing the roles of each of these devices is indicated by a "P" for provider and "PE" for provider edge. Based on the current configuration we will establish the operational components needed to create the Layer 3 VPN tunnel. We will start this on R5 and work our way to R2. The object of this exercise is to familiarize ourselves with the theory that we have discussed to date while demonstrating that theory in operation. We will begin by creating the first of the three components we discussed previously. The virtual routing and forwarding instance. To accomplish this we will make configuration changes on R5 and R2. We will start R5.

```
R5#conf t
R5(config)#ip vrf VPN-A ← VPN Community
R5(config-vrf)#rd 100:1
R5(config-vrf)#route-target 100:1 ← VPN community route-target value
R5(config-vrf)#exit
R5(config)#interface FastEthernet 0/0
R5(config-if)#ip vrf forwarding VPN-A ← Interface participating in the VPN Community
%Interface FastEthernet0/0 IP address 172.16.15.5 removed due to enabling VRF VPN-A
R5(config-if)#ip address 172.16.15.5 255.255.255.0
R5(config-if)#end
```

Remember that we discussed how the ip address will be removed at the time an interface is placed into a given VRF instance. We will repeat this process on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip vrf VPN-A ← VPN Community
R2(config-vrf)#rd 100:2
R2(config-vrf)#route-target 100:1 ← VPN community route-target value
R2(config-vrf)#exit
R2(config)#interface GigabitEthernet 0/1
R2(config-if)#ip vrf forwarding VPN-A ← Interface participating in the VPN Community
%Interface GigabitEthernet0/1 IP address 172.16.24.2 removed due to enabling VRF VPN-A
R2(config-if)#ip address 172.16.26.2 255.255.255.0
R2(config-if)#end
```

With this accomplished we have created the VRF instances and their associated community values. Now we will move to the next phase of our implementation process.

Implement Multiprotocol BGP peering of VPN community PE routers

The next step in creating the Layer 3 VPN tunnel is to build the MP-BGP peering session between the PE routers. With this accomplished we create the actual instance vpnv4 address family context. We will need to activate the neighbor under this context as well as ensure that the extended community route-target values are allowed to be sent. Note that for this configuration exercise we are disabling the default behavior of BGP to automatically create and enable ipv4 routing. We are doing this to ensure that we are in control of all aspects of the MP-BGP deployment. We will initiate this on R5 first.

```
R5#conf t
R5(config)#router bgp 65000
R5(config-router)# no bgp default ipv4-unicast
R5(config-router)# bgp log-neighbor-changes
R5(config-router)# neighbor 192.1.2.2 remote-as 65000
R5(config-router)# neighbor 192.1.2.2 update-source Loopback0
R5(config-router)# !
R5(config-router)# address-family vpnv4
R5(config-router-af)#  neighbor 192.1.2.2 activate
R5(config-router-af)#  neighbor 192.1.2.2 send-community extended
R5(config-router-af)# exit-address-family
R5(config-router)# !
R5(config-router)# address-family ipv4 vrf VPN-A
R5(config-router-af)#  no synchronization
R5(config-router-af)# exit-address-family
```

With this accomplished we need to complete other end of the tunnel by making the necessary configurations on R2.

```
R2#conf t
R2(config)#router bgp 65000
R2(config-router)# no bgp default ipv4-unicast
R2(config-router)# bgp log-neighbor-changes
R2(config-router)# neighbor 192.1.5.5 remote-as 65000
R2(config-router)# neighbor 192.1.5.5 update-source Loopback0
R2(config-router)# !
R2(config-router)# address-family vpnv4
R2(config-router-af)#  neighbor 192.1.5.5 activate
R2(config-router-af)#  neighbor 192.1.5.5 send-community extended
R2(config-router-af)# exit-address-family
R2(config-router)# !
R2(config-router)# address-family ipv4 vrf VPN-A
R2(config-router-af)#  no synchronization
R2(config-router-af)# exit-address-family
R2(config-router)#
```

We can verify that the VPNv4 tunnel is created by using the show ip bpg vpnv4 all sum command.

```
R5#show ip bgp vpnv4 all sum
BGP router identifier 192.1.5.5, local AS number 65000
```

```
BGP table version is 1, main routing table version 1

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.2.2       4        65000      11      10        1    0    0 00:08:04        0
R5#
```

We can see that we have a vpnv4 session initiated with our MP-BGP neighbor 192.1.2.2. Note that have received no information regarding learned prefixes. At this time we do not expect to learn any prefixes so this not an issue. We will cover the capabilities and features associated with MP-BGP and MPLS in Chapter 5: MP-BGP, until then we will merely point out the fact that MP-BGP is necessary in order to create an operational VPNv4 Tunnel or what is also called a Layer 3 VPN tunnel.

Implement MPLS forwarding

As we discussed in the ***Chapter 3: Label Distribution*** we will create the control plan mechanism necessary to perform MPLS forwarding. This will be performed on a hop-by-hop basis. In order to control what labels are issued by LDP we will specify the range of labels on each router using the router number as part of the process. This will translate to X00 – X99 where "X" is the router number. We will start this process on R5 and work our way across the service provider cloud.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#
R5(config)#mpls ip
R5(config)#!
R5(config)#mpls label protocol ldp
R5(config)#  mpls label range 500 599
R5(config)#  mpls ldp router-id lo 0 force
R5(config)#!
R5(config)#interface FastEthernet0/1
R5(config-if)# mpls ip
R5(config-if)# end
R5#
```

With R5 configure we will move to R4.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#
R4(config)#mpls ip
R4(config)#!
R4(config)#mpls label protocol ldp
R4(config)#  mpls label range 400 499
R4(config)#  mpls ldp router-id lo 0 force
R4(config)#!
R4(config)#interface FastEthernet0/0
R4(config-if)# mpls ip
R4(config-if)# exit
R4(config)#!
R4(config)#interface FastEthernet0/1
R4(config-if)# mpls ip
R4(config-if)# end
```

Lastly, R2 will be configured.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#mpls ip
R2(config)#!
R2(config)#mpls label protocol ldp
R2(config)#  mpls label range 200 299
R2(config)#  mpls ldp router-id lo 0 force
R2(config)#!
R2(config)#interface GigabitEthernet0/0
R2(config-if)# mpls ip
R2(config-if)# end
R2#
```

The next logical step will be to use the commands we discussed in ***Chapter 3: Label Distribution*** to verify that we have LDP adjacencies and that labels are being issued and exchanged. We will start with R5 and work our way to R2. We will look at the status of any LDP peers first and then look at the MPLS forwarding table on each device.

```
R5#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.5.5:0
        TCP connection: 192.1.4.4.646 - 192.1.5.5.33846
        State: Oper; Msgs sent/rcvd: 16/17; Downstream
        Up time: 00:07:43
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 172.16.45.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4
```

R5 is peered with R4 (192.1.4.4), There are three prefixes connected to R4. Next we need to know if R5 is generating labels and learning labels. This is best accomplished via the **show mpls forwarding-table** command.

```
R5#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     172.16.24.0/24    0             Fa0/1      172.16.45.4
17     400           192.1.2.0/24      0             Fa0/1      172.16.45.4
18     Pop Label     192.1.4.0/24      0             Fa0/1      172.16.45.4
R5#
```

R5 is generating local labels from prefixes it has in its routing table, and we are learning a label from R4 (400) that we will use to label switch outbound traffic.

```
R4#show mpls ldp neighbor
    Peer LDP Ident: 192.1.2.2:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 192.1.2.2.646 - 192.1.4.4.49525
        State: Oper; Msgs sent/rcvd: 17/18; Downstream
        Up time: 00:08:51
        LDP discovery sources:
```

```
          FastEthernet0/0, Src IP addr: 172.16.24.2
        Addresses bound to peer LDP Ident:
          172.16.24.2     192.1.2.2
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 192.1.5.5.33846 - 192.1.4.4.646
        State: Oper; Msgs sent/rcvd: 17/17; Downstream
        Up time: 00:08:25
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 172.16.45.5
        Addresses bound to peer LDP Ident:
          172.16.45.5     192.1.5.5

R4#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
400    Pop Label     192.1.2.0/24     1016          Fa0/0      172.16.24.2
401    Pop Label     192.1.5.0/24     1143          Fa0/1      172.16.45.5
```

We see that R4 is also performing as expected. Lastly we will repeat the commands on R2 where we can anticipate behavior similar to that seen on R5.

```
R2#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.4.4.49525 - 192.1.2.2.646
        State: Oper; Msgs sent/rcvd: 19/18; Downstream
        Up time: 00:09:26
        LDP discovery sources:
          GigabitEthernet0/0, Src IP addr: 172.16.24.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4

R2#show mpls forwarding-table
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
16     Pop Label     172.16.45.0/24   0             Gi0/0      172.16.24.4
17     Pop Label     192.1.4.0/24     0             Gi0/0      172.16.24.4
18     401           192.1.5.0/24     0             Gi0/0      172.16.24.4
R2#
```

We can get a lot of benefit from taking a look at what we have just accomplished on these devices from a graphical point of view. Figure 4-2 represents the elements that we have put into place thus far.

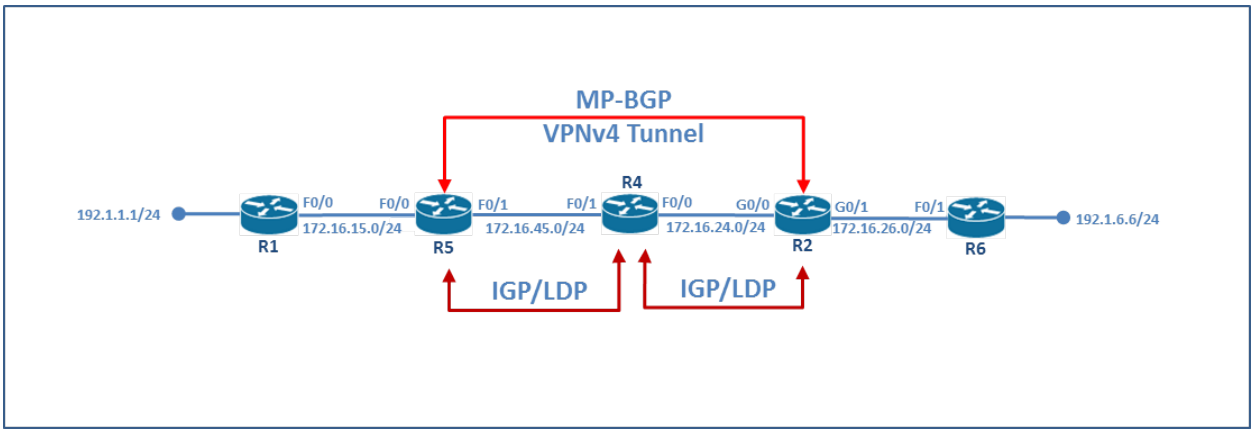


Figure 4-2: VPNv4 Tunnel Control Plane Illustration

This control plane mechanism contains all Layer-3 routing information and the label assignment information. This mechanism is what will enable the data plane to perform successful packet forwarding of both labeled and IP packets to the next hop toward the destination network.

Though not currently illustrated in our drawing we have to note that the CE Routers will be connected to PE Routers via a number of vrf aware routing protocols. We will explore what protocols we have to accomplish this in subsequent chapters in this book. In the MPLS backbone only an IGP along with LDP is used between PE routers and between P routers, and only MP-BGP is implemented between the PE's. These elements are illustrated in Figure 4-2 via the colored lines. Before we begin to look closer at the actual formation of the MPLS control plain we will create the necessary configuration needed to demonstrate that the VPNv4 Tunnel is operational. Rather than rely on a VRF aware routing protocol we will instead use static routes and static default routes to create our operational environment. This process will be covered in extensive detail in ***Chapter 6: Static PE-CE Routing***.

To create a working mpls environment we will configure a static default route on R1 pointing to the next hop ip address on the shared link between R1 and R5.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip route 0.0.0.0 0.0.0.0 172.16.15.5
```

Once the default static route has been assigned on R1 we will create vrf aware static routes to the network 192.1.1.0/24 on R1 such that we will have reachability to that network from R5. These static routes will need to be redistributed into the ipv4 addresses family that governs the vrf VPN-A. This is accomplished in the next configuration steps. Remember we are going to discuss this process in detail in Chapter 6: Static PE-CE Routing, so do not get bogged down in the operation details. We are doing this merely to generate prefixes that will be communicated via the Layer 3 VPN tunnel to R2 for testing and demonstration purposes.

```
R5#
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#ip route vrf VPN-A 192.1.1.0 255.255.255.0 172.16.15.1
```

```
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute static
R5(config-router-af)#exit-address-family
R5(config-router)#end
```

Now that this has been accomplished we will need to go to R2 and perform the same process there.

```
R2#
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip route vrf VPN-A 192.1.6.0 255.255.255.0 172.16.26.6
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute static
R2(config-router-af)#exit-address-family
R2(config-router)#end
```

Lastly R6 will need a static default route to reach any prefixes no assigned to R6 as connected or physical interfaces.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ip route 0.0.0.0 0.0.0.0 172.16.26.2
```

While we are on R6 we will perform some testing. Initially we will attempt to ping the network 192.1.1.0/24 by using the ping utility.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Note that this test fails. The test fails because R1 has no knowledge regarding how to reach a the source address of 172.16.26.6 we did not redistribute that information into the MP-BGP process. What would happen if we originated the ping from the 192.1.6.6 instead?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This ping is successfully. It works because we have advertised the NRLI to R5 via the VPNv4 tunnel we created previously. We can demonstrate how this process works by taking an addition step to redistribute the connected vrf enabled interfaces into the MP-BGP process on both R2 and R5. This will

prevent us from having to specify the loopback addresses as sources for subsequent testing. The procedures to accomplish this are as follows. We will start on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute connected
R2(config-router-af)#end
```

Then we will repeat the process on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute connected
R5(config-router-af)#end
```

Now we should be able to successfully ping without specifying the source.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This testing tells us unequivocally that the Layer 3 VPN is working. In fact we can see that this is the case via the **show ip bgp vpnv4 all summary** command. This command will give us valuable information regarding the VPNv4 tunnel . Specifically in this instance it will tell us that it is up and that we are learning 2 prefixes from the other side of the MPLS network.

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 9, main routing table version 9
6 network entries using 936 bytes of memory
6 path entries using 408 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1904 total bytes of memory
BGP activity 6/0 prefixes, 6/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000     148     148        9    0    0 02:24:27        2
```

After manually performing this configuration we can clearly anticipate that the two prefixes that R2 is learning will be 192.1.1.0/24 and 172.16.15.0/24. These being the prefixes we redistributed into the MP-

BGP process, and furthermore we should expect that the next hop address will be that of the PE neighbor because this is an MP-iBGP peering.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

There are a lot of moving parts associated with this configuration that are necessary to illustrate that the Layer 3 VPN is working. Chapters subsequent to this one will deal with each of these parameters in turn. So we will consciously ignore these and focus instead on the parameters associated with the Layer 3 VPN itself. We will start with the value of [V] that now appears in the LFIB for prefixes that are advertised into the VPNv4 tunnel. We can see these values with the **show mpls forwarding-table** command.

```
R2#show mpls forwarding-table
Local  Outgoing      Prefix            Bytes Label   Outgoing    Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     172.16.45.0/24    0             Gi0/0       172.16.24.4
17     Pop Label     192.1.4.0/24      0             Gi0/0       172.16.24.4
18     401           192.1.5.0/24      0             Gi0/0       172.16.24.4
200    No Label      192.1.6.0/24[V]   1140          Gi0/1       172.16.26.6
201    No Label      172.16.26.0/24[V] 570           aggregate/VPN-A
```

Remember from our previous discussion that the contents of the show mpls forwarding table deals with the FEC/Prefixes that are local generated and assigned. In this case the “V” indicates that a [V]PN label has been local assigned. Before we move on to an illustration of the process that we have just deployed we will spend some additional time discussing the other value we see in this output. We see that R2 is also assigning an aggregate label; this means that a layer3 lookup is needed to forward the packet and aggregate labels will be assigned by PE routers in the context of layer 3 VPNs to all local IP address for a given VRF to conserve the label space. They are assigned to locally connected networks and are also assigned to aggregate prefixes as well like instances where BGP summarization is employed.

Summary and Overview

We have covered a tremendous amount of ground so now I want to take the time to review each of these components on a step by step basis to make certain we have a firm grasp on the theory and operation of Layer 3 VPNs. I cannot stress enough the importance of this topic regarding the successful completion of the CCIE lab exam.

Control Plan Formation

To begin with we will start with R1. In our topology R1 is the first CE and it is originating the prefix 192.1.1.0/24. Rather than working with a protocol we simply configure a static default route pointing to R5. We can see this via the show ip route command on R1.

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 172.16.15.5 to network 0.0.0.0

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.15.0 is directly connected, FastEthernet0/0
C    192.1.1.0/24 is directly connected, Loopback0
S*   0.0.0.0/0 [1/0] via 172.16.15.5
```

In our topology R5 is the first PE router. It is also one end of the Layer 3 VPN tunnel. We created a vrf aware static route pointing to the network 192.1.1.0/24 on R1. This provides reachability from the PE to the 192.1.1.0/24 network on R1. We can see this static route via the **show ip vrf VPN-A** command on R5.

```
R5#show ip route vrf VPN-A

Routing Table: VPN-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.26.0 [200/0] via 192.1.2.2, 01:11:09
C       172.16.15.0 is directly connected, FastEthernet0/0
B    192.1.6.0/24 [200/0] via 192.1.2.2, 01:21:54
S    192.1.1.0/24 [1/0] via 172.16.15.1
```

We can clearly see the static route is present in the VRF VPN-A routing table. Between the static default route on R1 pointing to R5 and the VRF aware static route pointing to 192.1.1.1 we can assume we should have reachability.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
```



```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

As we can see we do have reachability. Now we have to look at the role base operations that take place on R5. I say role based because R5 is many things. It is a label switch router. It is a provider edge router and it is a VPNv4 Peer in a Layer 3 VPN. This means that R5 will perform a number of functions.

First, R5 will assign a 64 bit route-designator value of 100:1 to the prefix 192.1.1.0/24. This can be seen via the output of the show ip bgp vpnv4 rd 100:1. The prefix we are discussing appears associated with the RD value of 100:1 that we are discussing.

```
R5#show ip bgp vpnv4 rd 100:1
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?
```

Next R5 will assign a route-target value of 100:1 to the prefix 192.1.1.0/24. This route-target value is an extended BGP community that is used to specify what VPN a give prefix is a member of. As we discussed in the previous chapters the route-target is used to affect what routes are transitioned to the IGP routing table. In this instance we are not going to focus on this, but we will in subsequent chapters. we can see the route-target value being assigned via the **show ip bgp vpnv4 all 192.1.1.0** command.

```
R5#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 3
Paths: (1 available, best #1, table VPN-A)
  Advertised to update-groups:
        1
  Local
    172.16.15.1 from 0.0.0.0 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:100:1
      mpls labels in/out 500/nolabel
```

Now that we have performed the initial steps associated with R5 role as a PE router (the assignment of the RD and RT values) we will begin addressing R5's role as a VPNv4 Peer in the Layer 3 VPN. This is where the VPNv4 address is assigned to the prefix 192.1.1.0/24. The VPNv4 address is composed of the 32 bit ip address for the network (192.1.1.0/24) and the 64 bit route-distinguisher value. This equates the a 96 bit long VPNv4 address that now serves to make the prefix unique within the MPLS service provider cloud. We can see this address clear by repeating the previous show command and looking at the very first line of the output.

```
R5#show ip bgp vpnv4 all 192.1.1.0
```

```
BGP routing table entry for 100:1:192.1.1.0/24, version 3
Paths: (1 available, best #1, table VPN-A)
  Advertised to update-groups:
        1
  Local
    172.16.15.1 from 0.0.0.0 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:100:1
      mpls labels in/out 500/nolabel
```

The next step taken by R5 is to assign the VPN Label to the prefix update. We can see this value by executing the **show mpls forwarding table detail** command.

```
R5#show mpls forwarding-table detail | sec [V]
Label  Label or VC   or Tunnel Id      Switched      interface
500    No Label      192.1.1.0/24[V]   3534          Fa0/0      172.16.15.1
       VPN route: VPN-A
501    No Label      172.16.15.0/24[V] 0             aggregate/VPN-A
       VPN route: VPN-A
R5#show mpls forwarding-table detail
Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
16     Pop Label     172.16.24.0/24    0             Fa0/1      172.16.45.4
       MAC/Encaps=14/14, MRU=1504, Label Stack{}
       000AB819C921000AB819C6B18847
       No output feature configured
17     400           192.1.2.0/24      0             Fa0/1      172.16.45.4
       MAC/Encaps=14/18, MRU=1500, Label Stack{400}
       000AB819C921000AB819C6B18847 00190000
       No output feature configured
18     Pop Label     192.1.4.0/24      0             Fa0/1      172.16.45.4
       MAC/Encaps=14/14, MRU=1504, Label Stack{}
       000AB819C921000AB819C6B18847
       No output feature configured
500    No Label      192.1.1.0/24[V]   3534          Fa0/0      172.16.15.1
       MAC/Encaps=14/14, MRU=1504, Label Stack{}
       000AB86BA3F0000AB819C6B00800
       VPN route: VPN-A
       No output feature configured
       <output omitted>
```

R6 now modifies the next hop reachability information for the prefix 192.1.1.0 with its loopback address. This is because that will the address its Layer 3 VPN peer R2 will use to reach this network. We can see that the loopback address is the next hop by moving to R2 and using the **show ip bgp vpnv4 all** command.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
```

```
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5               0    100      0 ?
*>i192.1.1.0        192.1.5.5               0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5               0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                 0         32768 ?
*>i192.1.1.0        192.1.5.5               0    100      0 ?
*> 192.1.6.0        172.16.26.6             0         32768 ?
```

The final task that R5 most perform is to propagate its loopback address to R4 with an imp-null label. This can be verified by using the **show mpls ldp bindings 192.1.5.0 24** command on that router.

```
R4#show mpls ldp bindings 192.1.5.0 24
  lib entry: 192.1.5.0/24, rev 10
        local binding:  label: 401
        remote binding: lsr: 192.1.2.2:0, label: 18
        remote binding: lsr: 192.1.5.5:0, label: imp-null
```

Once R4 has received the advertisement of the imp-null label, it will generate a local label to use for this prefix we can see this label via the output of the show mpls forwarding-table command.

```
R4#show mpls forwarding-table 192.1.5.0
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
401    Pop Label     192.1.5.0/24     34244         Fa0/1      172.16.45.5
```

We can see that R4 has locally bound the label 401 to the prefix 192.1.5.0. Knowing how MPLS operates we realize that R4 will now advertise that label to its next peer R2. We can see the fact by using the show mpls forwarding-table command on that device as well.

```
R2#show mpls forwarding-table 192.1.5.0
Local  Outgoing      Prefix           Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id     Switched      interface
18     401           192.1.5.0/24     0             Gi0/0      172.16.24.4
```

The prefix 192.1.5.0/24 as locally assigned a label of 18, but we learned about the label value of 401 from R4 and as we can see R2 knows to use the next hop 172.16.24.4 to reach this prefix. We now find ourselves on R2 which just happens to be participating in the Layer 3 VPN with R5. We can see all the prefixes learned via the Layer 3 VPN by using the **show ip bgp vpn4 rd 100:1** command.

```
R2#show ip bgp vpnv4 rd 100:1
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5               0    100      0 ?
*>i192.1.1.0        192.1.5.5               0    100      0 ?
```

Data Plane Forwarding

We now have to consider the behavior of the data plane forwarding element as it relates and is involved with the Layer 3 VPN concept. In the topology and configuration we have been discussing in the chapter we can now visualize R6 sending an ip data packet to the address 192.1.1.1. For the purpose of this discussion we will specify the source address as 192.1.6.6. Once the packets arrives R2 will append the VPN label to the packet as well as the LDP label 401 that will be used as the MPLS transport label. This packet will then be sent to R4. R4 will receive the packet and pop the top label because it has received an imp-null label from R5. The resulting packet with only the VPN label remaining is forwarded to R5. R5 will pop the VPN label and forward the remain IP packet to R1.

The primary element in this description is that the VPN label is not removed until it reaches the destination PE at the other end of the Layer 3 VPN tunnel. Note that all of the MPLS forwarding between the PE devices is handled by the configured interior gateway protocol and the label distribution protocol. We can see this process by actually generating a traceroute from R6's Loopback 0 interface destined to the 192.1.1.1 address on R1 and observing the results.

```
R6#traceroute 192.1.1.1 source lo 0

Type escape sequence to abort.
Tracing the route to 192.1.1.1

  1 172.16.26.2 4 msec 0 msec 0 msec
  2 172.16.24.4 [MPLS: Labels 401/500 Exp 0] 4 msec 0 msec 4 msec
  3 172.16.15.5 [MPLS: Label 500 Exp 0] 0 msec 4 msec 0 msec
  4 172.16.15.1 4 msec *  0 msec
```

We can observe the two labels in the label stack by executing the **show ip cef vrf VPN-A 192.1.1.0 detail** command on R2.

```
R2#show ip cef vrf VPN-A 192.1.1.0 detail
192.1.1.0/24, epoch 0
  recursive via 192.1.5.5 label 500 ←VPN label
    nexthop 172.16.24.4 GigabitEthernet0/0 label 401 ←Transport Label
```

Useful PE show commands

Thus far we have used a number of show commands in this chapter that you may be unfamiliar with. At this point we will outline these commands and what they accomplish for us. We suggest you take the time to study and practice using these commands and looking critically at the key components they describe in our Layer 3 VPN tunnel environments.

show ip bgp vpnv4 all summary – the equivalent of the "show ip bgp summary" command in ipv4 BGP, and it is used to list all MP-BGP and CE peers.

```
R2#show ip bgp vpnv4 all summary
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 9, main routing table version 9
6 network entries using 936 bytes of memory
6 path entries using 408 bytes of memory
```

```
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1904 total bytes of memory
BGP activity 6/0 prefixes, 6/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000    1045    1045        9    0    0 17:27:40        2
```

show ip bgp vpnv4 all – lists all VPN prefixes advertised and learned by the PE router.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

show ip bgp vpnv4 vrf <vrf> summary – the same output as that in the previous command but for only one specific VRF.

```
R2#show ip bgp vpnv4 vrf VPN-A
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

show ip bgp vpnv4 vrf <vrf> - lists all the VPN prefixes learned in a specific VRF

```
R2#show ip bgp vpnv4 vrf VPN-A
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
```

```
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

show ip bgp vpnv4 vrf <vrf> labels – provides a list of labels for the VPN prefixes in the specified VRF.

```
R2#show ip bgp vpnv4 vrf VPN-A labels
   Network          Next Hop      In label/Out label
Route Distinguisher: 100:2 (VPN-A)
   172.16.15.0/24   192.1.5.5       nolabel/501
   172.16.26.0/24   0.0.0.0         201/nolabel(VPN-A)
   192.1.1.0        192.1.5.5       nolabel/500
   192.1.6.0        172.16.26.6     200/nolabel
```

One other useful command that we will introduce in this chapter and use extensively in the subsequent sections of this book is the following command:

show ip bgp vpnv4 all neighbors <neighbor> advertised-routes – command used to detail the routes being advertised from one vpnv4 peer to another. This tool is very handy for identifying issues affecting the operation of the provider cloud and the Layer 3 VPN itself.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

Common Issues with Layer 3 VPNs

Layer 3 VPNs have a number of issues that can surface when deployed. The most common problems relate to the three operational components that we discussed previously in the chapter.

VPN Route Target Community Issues

Failures of this type can effect what prefixes can and cannot be successfully communicated between Layer 3 VPN peers. This type of issue is typically associated with misconfigurations made on the PE devices.

Multiprotocol BGP peering Issues

Again a category of error that will predominately affect PE devices only, however Chapter 5: MP-iBGP will explore circumstances were this type of issue can be found on certain P routers in our topology as well. But the crux of this type of problem will be clearly visible it you follow the Quick-Fire approach outlined for troubleshooting MPLS VPNs.

MPLS Forwarding Issues

MPLS forwarding problems can be present on any device in the label switched path. The nature of these problems can come any form that we have discussed in the previous chapters of this book. If you do not have a firm understanding of the operation and troubleshooting of the MPLS forwarding and control plane formation at this point it is suggested that you go no further, because subsequent chapters of this book will only build on that foundation as we move forward.

Quick-Fire MPLS VPN Time Optimized Approach

As with any complex protocol the best and most efficient method we have to isolate issues is an organized and detailed approach that covers all the primary operational elements. The Quick-Fire method for MPLS Layer 3 VPNs is as follows.

1. Make sure that “export RT <x>” on the PE routers match with the “import RT <x>” on the receiving PE.
    - **show ip vrf detail VPN-A | inc Export|Import|RT|route-map**
2. If an export or import-map are configured in the VRF, then validate the RT in the “set clause”
    - **show ip vrf de <vrf> | inc router-map**
    - **show route-map <map>**

3. If BGP is not used as the PE-CE protocol, then make sure the redistribution between BGP’s VRF aware instance and the respective IGP’s VRF aware instance is performed. If the static routing is employed just verify the redistribution in to MP-BGP.
    - **show run | sec router**

4. MP-BPG neighbors must be configured to send extended-community values
    - **show run | inc send-community**

5. Make sure that labels in the MP-BGP VPN table matches the label in the FIB table for a received VPN prefix.
    - **show ip bgp vpnv4 vrf label | inc <prefix>**
    - **show ip cef vrf <vrf> <prefix>**

6. Make sure that the label in the MP-BGP VPN table matches with the label in the LFIB table for an advertise route VPN prefix
    - **show ip bgp vpnv4 vrf <vrf> label | inc <prefix>**
    - **show mpls forwarding vrf <vrf> | inc <prefix>**

7. If these checks all verify it will be necessary to troubleshoot possible label distribution issues in the service provider cloud.
    - **show mpls ldp neighbor**

Our typical approach in exploring these technologies is to discuss the common issues impacting their deployment but we always take the opportunity use the command we suggest in a working environment so that students can see what the operational output should look like.

Verification walk through

1. Make sure that "export RT 100:1" on the R5 and the Import RT 100:1 on R2 match
    - **show ip vrf detail <vrf> | inc Export|import|RT**

```
R5#show ip vrf detail VPN-A | inc Export|Import|RT|route-map
  Export VPN route-target communities
    RT:100:1
  Import VPN route-target communities
    RT:100:1
  No import route-map
  No export route-map

R2#show ip vrf detail VPN-A | inc Export|Import|RT|route-map
  Export VPN route-target communities
    RT:100:1
  Import VPN route-target communities
    RT:100:1
  No import route-map
  No export route-map
```

2. If an export or import-map are configured in the VRF, then validate the RT in the "set clause"
    - **show ip vrf de <vrf> | inc router-map**
    - **show route-map <map>**

There are no import or export maps in use in this topology.

3. If BGP is not used as the PE-CE protocol, then make sure the redistribution between BGP's VRF aware instance and the respective IGP's VRF aware instance is performed. If the static routing is employed just verify the redistribution in to MP-BGP.
    - **show run | sec router**

```
R5#show run | sec router
router eigrp 100
 network 172.16.45.5 0.0.0.0
 network 192.1.5.5 0.0.0.0
 no auto-summary
router bgp 65000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  redistribute connected
  redistribute static
  no synchronization
 exit-address-family

R2#show run | sec router
router eigrp 100
 network 172.16.24.2 0.0.0.0
 network 192.1.2.2 0.0.0.0
 no auto-summary
router bgp 65000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.1.5.5 remote-as 65000
 neighbor 192.1.5.5 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.1.5.5 activate
  neighbor 192.1.5.5 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  redistribute connected
  redistribute static
  no synchronization
 exit-address-family
```

Based on the fact that we are running static routes we see that the redistribution is correct.

4. MP-BPG neighbors must be configured to send extended-community values
    - **show run | inc send-community**

```
R5# show run | inc send-community
  neighbor 192.1.2.2 send-community extended

R2# show run | inc send-community
  neighbor 192.1.5.5 send-community extended
```

5. Make sure that labels in the MP-BGP VPN table matches the label in the FIB table for a received VPN prefix.
   - **show ip bgp vpnv4 vrf label | inc <prefix>**
   - **show ip cef vrf <vrf> <prefix>**

```
R2#show ip bgp vpnv4 vrf VPN-A label | inc 192.1.1.0
   192.1.1.0        192.1.5.5       nolabel/500

R2#show ip cef vrf VPN-A 192.1.1.0
192.1.1.0/24
  nexthop 172.16.24.4 GigabitEthernet0/0 label 401 500
```

6. Make sure that the label in the MP-BGP VPN table matches with the label in the LFIB table for an advertise route VPN prefix
   - **show ip bgp vpnv4 vrf <vrf> label | inc <prefix>**
   - **show mpls forwarding vrf <vrf> | inc <prefix>**

```
R5#show ip bgp vpnv4 vrf VPN-A label | inc 192.1.1.0
   192.1.1.0        172.16.15.1     500/nolabel

R5#show mpls forwarding-table vrf VPN-A | inc 192.1.1.0
500    No Label      192.1.1.0/24[V]   7638          Fa0/0      172.16.15.1
```

7. If these checks all verify it will be necessary to troubleshoot possible label distribution issues in the service provider cloud.

```
R5#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.5.5:0
        TCP connection: 192.1.4.4.646 - 192.1.5.5.29008
        State: Oper; Msgs sent/rcvd: 1281/1275; Downstream
        Up time: 18:30:21
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 172.16.45.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4

R4#show mpls ldp neighbor
    Peer LDP Ident: 192.1.2.2:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 192.1.2.2.646 - 192.1.4.4.49525
        State: Oper; Msgs sent/rcvd: 1291/1300; Downstream
        Up time: 18:45:33
        LDP discovery sources:
          FastEthernet0/0, Src IP addr: 172.16.24.2
        Addresses bound to peer LDP Ident:
          172.16.24.2     192.1.2.2
```

```
    Peer LDP Ident: 192.1.5.5:0; Local LDP Ident 192.1.4.4:0
        TCP connection: 192.1.5.5.29008 - 192.1.4.4.646
        State: Oper; Msgs sent/rcvd: 1276/1282; Downstream
        Up time: 18:31:22
        LDP discovery sources:
          FastEthernet0/1, Src IP addr: 172.16.45.5
        Addresses bound to peer LDP Ident:
          172.16.45.5     192.1.5.5

R2#show mpls ldp neighbor
    Peer LDP Ident: 192.1.4.4:0; Local LDP Ident 192.1.2.2:0
        TCP connection: 192.1.4.4.49525 - 192.1.2.2.646
        State: Oper; Msgs sent/rcvd: 1301/1293; Downstream
        Up time: 18:46:50
        LDP discovery sources:
          GigabitEthernet0/0, Src IP addr: 172.16.24.4
        Addresses bound to peer LDP Ident:
          172.16.24.4     172.16.45.4     192.1.4.4
```

Chapter Challenge: Layer 3 VPN Sample Trouble Tickets

The following section includes two sample Trouble Tickets designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. These Trouble Tickets were designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configurations provided in the file ***MPLS-CH4-L3VPN-TT-INITIAL.txt***. Keep in mind these sample Trouble Tickets were also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 4-3 below:

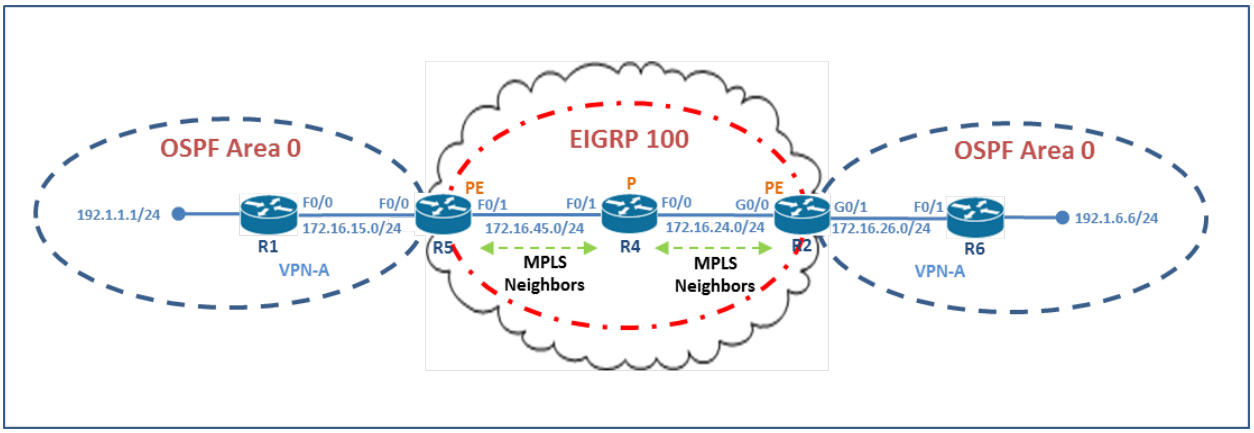


Figure 4-3: The Chapter Challenge Topology

Trouble Ticket #1

Your manager has brought to your attention the fact that the MP-BGP VPNV4 tunnel between R5 and R2 is not forming. You have been instructed to isolate why this is happening and take the necessary actions needed to correct this problem. The evidence provided by your manager that this situation is occurring is the output in Exhibit "A".

Exhibit "A"

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 1, main routing table version 1
1 network entries using 156 bytes of memory
1 path entries using 68 bytes of memory
2/0 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 584 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000       0       0        0    0    0 never    Active
```

Trouble Ticket #2

After solving Trouble Ticket #1, your supervisor while doing more testing has observed that R6 cannot successfully ping the IP address 192.1.1.1 as evidence in the output provide.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Chapter Challenge: Layer 3 VPN Sample Trouble Tickets Solutions

The following section includes the solutions to the two Trouble Tickets presented in the previous section.

Trouble Ticket #1 Solution

Your manager has brought to your attention the fact that the MP-BGP VPNV4 tunnel between R5 and R2 is not forming. You have been instructed to isolate why this is happening and take the necessary actions needed to correct this problem. The evidence provided by your manager that this situation is occurring is the output in Exhibit "A".

Exhibit "A"

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 1, main routing table version 1
1 network entries using 156 bytes of memory
1 path entries using 68 bytes of memory
2/0 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 584 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000       0       0        0    0    0 never    Active
```

Step 1 - Fault Verification:

Does the output of the show ip bgp vpnv4 all sum command match the exhibit?

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 3, main routing table version 3
1 network entries using 156 bytes of memory
1 path entries using 68 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 584 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000       0       0        0    0    0 never    Active
```

The Layer 3 VPN tunnel is not forming.

Step 2 - Fault Isolation:

The next course of action is to use the **show ip bgp vpnv4 all summary** command on R5 to see the status of that end of the tunnel.

```
R5#show ip bgp vpnv4 all sum

R5#
```

The verification clearly demonstrates that R5 is not even attempting to initiate a VPNV4 tunnel with any peer. We now need to check the MP-BGP configuration on R5 to see how the process has been configured.

```
R5#show run | sec router bgp
router bgp 65000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 !
 address-family ipv4 vrf VPN-A
  redistribute connected
  redistribute static
  no synchronization
 exit-address-family
```

We see that the MP-BGP configuration is missing any reference to the VPNv4 Unicast address-family. This configuration is essential to create the Layer 3 VPN. We will need to apply this missing configuration in order to remediate this ticket.

Step 3 - Fault Remediation:
In this scenario, the **address-family vpnv4** command should be applied under the MP-BGP process on R5.

```
R5(config)#router bgp 65000
R5(config-router)#address-family vpnv4 unicast
R5(config-router-af)#neighbor 192.1.2.2 activate
R5(config-router-af)#neighbor 192.1.2.2 send-community extended
R5(config-router-af)#end
```

We should see the BGP process come up.

```
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Up
```

We can now check the status of the VPNv4 tunnel on R5 before moving back to R2.

```
R5#
%SYS-5-CONFIG_I: Configured from console by console
R5#show ip bgp vpnv4 all sum
BGP router identifier 192.1.5.5, local AS number 65000
BGP table version is 9, main routing table version 9
```



```
4 network entries using 624 bytes of memory
4 path entries using 272 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1456 total bytes of memory
BGP activity 4/0 prefixes, 4/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.2.2       4        65000       6       6        9    0    0 00:02:08        1
```

We see that the link is up and that we are learning on prefix from R2.

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 7, main routing table version 7
5 network entries using 780 bytes of memory
5 path entries using 340 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1680 total bytes of memory
BGP activity 5/0 prefixes, 5/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000       7       7        7    0    0 00:03:08        2
```

The issue has been corrected.

Trouble Ticket #2 Solution

After solving Trouble Ticket #1, your supervisor while doing more testing has observed that R6 cannot successfully ping the IP address 192.1.1.1 as evidence in the output provide.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Can R6 ping 192.1.1.1 while sourcing packets from 192.1.6.6:

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

The ping test to the multicast group 224.9.9.9 fails. This verifies that the problem actually exists.

Step 2 - Fault Isolation:

The next course of action will be to go to R1 and see if the route exists.

```
R1#show ip route 192.1.1.0
Routing entry for 192.1.1.0/24
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Routing Descriptor Blocks:
  * directly connected, via Loopback0
      Route metric is 0, traffic share count is 1
```

We see that the router is a connected route. Next we need to realize that there is no routing protocol running between R5 and R1 only static routes. We will now move to R5 and see if it even has reachability to the prefix in the first place.

```
R5#show ip route 192.1.1.0
% Network not in table
```

At first blush it would appear that R5 does not have reachability to 192.1.1.0/24, but we have to remember that this prefix should appear in the vrf for VPN-A. So we will repeat our show route command with that in mind.

```
R5#show ip route vrf VPN-A 192.1.1.0
Routing entry for 192.1.1.0/24
  Known via "static", distance 1, metric 0
  Redistributing via bgp 65000
  Advertised by bgp 65000
  Routing Descriptor Blocks:
  * 172.16.15.1
      Route metric is 0, traffic share count is 1
```

As a final verification we will attempt to ping this address and look for reachability.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R5#
```

This eliminates any possible issues between the PE and the CE device. Now we need to see if this route is present in the MP-BGP tables on R5.

```
R5#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 7
Paths: (1 available, best #1, table VPN-A)
  Advertised to update-groups:
        1
  Local
    172.16.15.1 from 0.0.0.0 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:100:1
      mpls labels in/out 501/nolabel
```

The route is present. Next we need to see if the vpnv4 process is advertising that prefix to the VPNv4 Peer 192.1.2.2.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

The prefix is being advertised. Next we will move to R2 and see if that router learns about this prefix via the Layer 3 VPN tunnel.

```
R2#show ip bgp vpnv4 rd 100:1 neighbors 192.1.5.5 routes
BGP table version is 7, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?

Total number of prefixes 2
```

R2 is learning the prefix. The next question is can R2 reach the networks attached to the R6 CE device? Remember that this is a vrf instance VPN-A as well and must be specified in the ping command.

```
R2#ping vrf VPN-A 172.16.26.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.26.6, timeout is 2 seconds:
```



```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R2#ping vrf VPN-A 192.1.6.6

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

The PE to CE connection seems to be fully operational, but we cannot ping 192.1.1.1 from R6. Can we ping the prefix from R2?

```
R2#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
```

We have reachability to 192.1.1.1 but keep in mind that we have been instructed to source our pings from the ip address 192.1.6.6. This address originates on R6 and we know we have reachability to it from R2 based on previous tests. The question becomes is this information being advertise into the MP-BGP process.

```
R2#show ip bgp vpnv4 all 192.1.6.0
% Network not in table
```

Observe that the output indicates there is no entry in the MP-BGP process for this route or reachability information to it. The question we have to ask now is, “how does this information make it into BGP?” We know we are running static routing between the PEs and CEs in this topology, and we know that unless we are running BGP no information is going to be advertised into BGP without redistribution. So we will look for that in the configuration next.

```
R2#show run | sec router bgp
router bgp 65000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.1.5.5 remote-as 65000
 neighbor 192.1.5.5 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.1.5.5 activate
  neighbor 192.1.5.5 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  redistribute connected
  no synchronization
 exit-address-family
```

We are redistributing connected routes but we are not redistributing the static routing information R2 uses to reach R6. We will have to add that in the next step of our procedure.

Step 3 - Fault Remediation:

In this scenario, the **redistribute static** command needs to be applied to R2 under the MP-BGP routing process for the ipv4 vrf address-family:

```
R2(config)#router bgp 65000
R2(config-router)# address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute static
R2(config-router-af)#end

Next we will see if the route is now in the MP-BGP routing table.

R2#show ip bgp vpnv4 all 192.1.6.0
BGP routing table entry for 100:2:192.1.6.0/24, version 9
Paths: (1 available, best #1, table VPN-A)
Flag: 0x820
  Advertised to update-groups:
        1
  Local
    172.16.26.6 from 0.0.0.0 (192.1.2.2)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:100:1
      mpls labels in/out 204/nolabel
```

Step 4 - Verification of Remediation

Once the error has been isolated and remediated, it is highly recommended to verify that the Trouble Ticket has been repaired using the same method used to verify the fault initially:

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

The issue has been corrected.

Chapter 5: MP-iBGP

BGP functionality had to be enhanced to support the “VRF” specific routes needed in Layer 3 VPNs. A new special MP-BGP (multiprotocol BGP) address family named “VPNv4” (VPN IPv4) has been added to BGP along with a new NLRI format. In this chapter we are going to explore this concept and the basic operational enhancements that have been made to MP-BGP.

Introduction to MP-iBGP

The very nature of MPLS makes it a perfect candidate for a dynamic tunneling solution that can easily provide both granularity and scalability. The fundamental purpose of MPLS VPNs is to create a full-mesh of dynamic MPLS Label Switching Routers. These LSR's will serve the role of Provider Edge devices and form the tunnel that will be used to exchange VPNv4 routes. We observed in ***Chapter 4: MPLS Layer 3 VPN*** how in order to select the correct VRF instance on a given end-point the individual PE routers add an additional label that drives the selection of the proper FIB entry associated with the respective VRF. The results in two labels existing in the MPLS stack – the first label is the transport label (sometimes called the "Outer" or "IGP" label. This topmost label in the stack will be swapped throughout the LSP between the PE devices. The remaining label is the VPN label, this label is often called the "Inner" or the BGP label and is used by the PE to ensure the proper outgoing VRF CEF entry.

This functionality that exists natively in MPLS was found to be an ideal way manage the operation of MPLS Layer 3 VPNs, however it was necessary to find a way to distribute these VPN routes between sites. The primary issue with this distribution process as it relates to IGP protocols between LSR neighbors is the unidirectional nature of the adjacencies themselves. This presented a large hurdle because even with the deployment of bi-directional tunneling technologies it would be necessary to have huge numbers of adjacencies across the provider core to ensure packet deliver. This obviously would translate into an administrative and scaling nightmare. It was for this reason that MP-BGP was chosen to satisfy prefix redistribution component of our Layer 3 VPNs. Based on MP-BGP virtually universal ability to redistribute almost any type of prefix this was a very natural solution. Obviously a number of enhancements had to be made to the basic operation of MP-BGP to allow it to deliver these new 96 bit VRF specific addresses. This new MP-BGP enhancement came in the form of a new address-family. The VPNv4 (VPN IPv4) address family was added to the MP-BGP along with some associated changes to the traditional NLRI format used by the protocol.

As we have discussing previous chapters and seen demonstrated in ***Chapter 4: MPLS Layer 3 VPNs*** every VPNv4 prefix is created by the combination of the 32 bit ip address and the 64 bit route-distinguisher, and that new address is assigned corresponding MPLS labels as well as certain BGP attributes. This allows us to differentiate between different VPN routes while they are transported as a group. This allows best-path selection to taken place separately for each different route-distinguisher. As we sa2 previously the VPNv4 address-family must be activated per-neighbor under the address-family vpnv4 context.

We need to point out a default behavior here before we move any further. When a new BGP neighbor is created using the command neighbor <ip> remote-as <as> an address-family ipv4 unicast is active for this the specified neighbor. The automatic process can be disabled as we saw in ***Chapter 4: MPLS Layer 3 VPNs***, via the **no bgp default ipv4-unicast** command.

MP-BGP VPN Prefix Exchange

There are a number special rules associated with this approach to VPNv4 prefix exchange. Specifically, it is necessary to establish the MP-iBGP peering using the loopbacks as the update source. It is necessary to use the full 32 bit host mask to accomplish this successfully. The /32 bit mask is necessary because the MP-BGP peering will use the peering IP address as the next-hop for the locally originated VPNv4 prefixes. When the remote PE receives these routes it will perform a recursive lookup against the next hop value in order to find the appropriate label in the LFIB. This is the label that will be used as the transport label at the destination router. The shortest and possibly the simplest explanation of what is happing here is that next-hop is used to build the VPN tunnel component. This tunnel is the transport label switched path connecting the PE's. This leaves the VPN label.

The VPN label is generated as a part of the MP-BGP operational process on the originating PE device. This VPN label will directly references the local VRF route. This is one of the reasons that the /32 mask is so important. The host prefix ensures that the transport LSP aka "the tunnel" will terminate on a the actual PE router rather than a shared segment between the PE and other connected devices.

As we observed in ***Chapter 4: MPLS Layer 3 VPNs*** it is necessary to redistribute the routes in a given VRF instance into the MP-BGP process. This is accomplished under the appropriate address-family ipv4 vrf context. This process is required for all VRF-aware PE-CE routing protocols (including static routes or connected interfaces). This means that all the respective routes belonging to the specific VRF instance will be injected into the MP-BGP table. These prefixes will be installed in the table along with their route-distinguishers. At this point the local VPN Labels will be generated.

MP-BGP Import Process

At this point we now need to consider the import process and its operational impact on MP-BGPs performance. The import process is somewhat more involved than any other part of the process we have discussed thus far. Route-targets are the values that drive the import process.

Route-targets are extended community attributes that are transitive in nature, by this we mean that they are exchanged as part of the prefix advertisement between BGP neighbors. We make reference to these values being extended because they are 64-bits in length, as opposed to standard community values that are 32-bits in length. We need these values because they support the enhancements needed by MP-BGP to label VPNv4 prefixes. The necessity for route-targets comes from the fact that the operational process cannot simply rely on route-distinguishers for prefix import and export. You have to be asking, "why not?" The answer is simple. Routes with the same RD may simultaneously belong to multiple VRFs, when you share their routes in throughout the topology.

Route-target based import works because each prefix redistributed into MP-BGP are tagged with the extended community "export" route-target . That community value takes the form of #:#, and is configured under the ip vrf configuration context with the **route-target export #:#** command. You can use the command to specify as many export route-target values as you want if you want to tag prefixes with multiple extended-community attributes.

These prefixes and their assigned attributes are advertised via the MPLS Layer 3 VPN to the remote PE device. Once they arrive the VRF will import the BGP VPNv4 prefixes that have route-targets that match the locally configured **route-target import #:#** command.

This should drive home the understanding that the prefix import process is founded entirely on the assigned route-targets and has nothing to do with the route-distinguisher (RD) values. In situations where the imported routes have different RD's from the one used by the local VRF, they will be changed by the MP-BGP routing process such that they will match the local VRF value. This process is so flexible that we could assign a different RD to each VPN site and then selectively decide what prefixes will be accepted by the import process. There are a number of tools at our disposal to make this happen and we will discuss them in later in this chapter.

Multiprotocol Capabilities

MP-BGP introduced two new BGP path attributes that are exchanged as part of the MP-BGP Open Messages. These two attributes perform critical operational roles in the MP-BGP deployment that supports MPLS Layer 3 VPNs. The first of these attributes is the Multiprotocol Reach NLRI (MP_REACH_NLRI).

MP_REACH_NLRI

This attribute is used to satisfy three primary roles. First NP_REACH_NLRI servers to advertise a feasible route to an MP-BGP peer. During this advertisement the MP_REACH_NLRI advertises the loopback address of the PE as the next hop for the address family carried in the message. This is essential for the successful operation of non-ipv4 address families like that of the VPNV4 family. Lastly, this attribute reports some or all of the local IP addresses. These addresses are also known as sub-network points of attachment (SNPAs).

MP_UNREACH_NLRI

This attribute is used to indicate a prefix that was previously being advertised is no longer reachable and thus signals that the route should be withdrawn.

These two new attributes are composed of multiple values, the first two values however will always contain the Address Family Identifier (AFI) and the Subsequent Address Family Identifier (SAFI).

AFI

The Address Family Identifier is what carries the identity of the network layer protocol associated with the network address being advertised. For the purposes of our discussions we are only going to concern ourselves with the two of these values

- IPv4 – represented as a value of 1
- IPv6 – represented as a value of 2

SAFI

The Subsequent Address Family Identifier is the component that provides any additional information about the type of Network Layer Reachability Information carried in the attribute. The makes reference to a number of parameters to include:

- Unicast – 1
- Multicast – 2
- Unicast and Multicast – 3
- MPLS Label – 4
- MPLS VPN Label 128

We can best observe these values during the BGP session negotiation by using the **debug bgp all** command. To force output we will clear the MP-BGP session.

```
R2#debug bgp all
BGP debugging is on for all address families
```

Now to force the session to reset.

```
R2#clear ip bgp *
R2#
BGPNSF state: 192.1.5.5 went from nsf_not_active to nsf_not_active
BGP: 192.1.5.5 went from Established to Idle
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Down User reset
R2#
BGP: 192.1.5.5 closing
BGP: 192.1.5.5 went from Idle to Active
BGP: 192.1.5.5 open active, local address 192.1.2.2
BGP: 192.1.5.5 read request no-op
BGP: 192.1.5.5 went from Active to OpenSent
BGP: 192.1.5.5 sending OPEN, version 4, my as: 65000, holdtime 180 seconds
BGP: 192.1.5.5 send message type 1, length (incl. header) 53
BGP: 192.1.5.5 rcv message type 1, length (excl. header) 34
BGP: 192.1.5.5 rcv OPEN, version 4, holdtime 180 seconds
BGP: 192.1.5.5 rcv OPEN w/ OPTION parameter len: 24
BGP: 192.1.5.5 rcvd OPEN w/ optional parameter type 2 (Capability) len 6
BGP: 192.1.5.5 OPEN has CAPABILITY code: 1, length 4
BGP: 192.1.5.5 OPEN has MP_EXT CAP for afi/safi: 1/128
BGP: 192.1.5.5 rcvd OPEN w/ optional parameter type 2 (Capability) len 2
BGP: 192.1.5.5 OPEN has CAPABILITY code: 128, length 0
BGP: 192.1.5.5 OPEN has ROUTE-REFRESH capability(old) for all address-families
BGP: 192.1.5.5 rcvd OPEN w/ optional parameter type 2 (Capability) len 2
BGP: 192.1.5.5 OPEN has CAPABILITY code: 2, length 0
BGP: 192.1.5.5 OPEN has ROUTE-REFRESH capability(new) for all address-families
BGP: 192.1.5.5 rcvd OPEN w/ optional parameter type 2 (Capability) len 6
BGP: 192.1.5.5 OPEN has CAPABILITY code: 65, length 4
BGP: 192.1.5.5 OPEN has 4-byte ASN CAP for: 65000
BGP: 192.1.5.5 rcvd OPEN w/ remote AS 65000, 4-byte remote AS 65000
BGP: 192.1.5.5 went from OpenSent to OpenConfirm
BGP: 192.1.5.5 went from OpenConfirm to Established
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Up
```

We see that we are exchanging the MP_EXT capability for the afi/safi: 1/128. This defines the unicast vpnv4 address family. Also note that we initiate this process of exchange as we mentioned earlier with the transmission of the initial open message.

Address Families

Address families where introduced with multiprotocol BGP to create a modular element that would make BGP adaptive and scalable. The ability to simultaneously allow multiple AFI and SAFI configurations has made MP-BGP almost universal in its capabilities and attributes to its broad acceptance in the networking space. The direct payoff when using MP-BGP is the fact that many networks are increasingly relying on MP-BGP to connect an almost unlimited number of autonomous systems that are running different unicast and multicast routing protocols including MPLS, and IPv6. MP-BGP is the one protocol that can provide a single transport solution for all of these networks.

We have to point out that this new AFI approach did make some significant changes to the command line interface commands needed to work with modified internal structure of MP-BGP. This necessity is driven by the need for MP-BGP to carry routing information for multiple network protocols as well as IP multicast routes. These many "flavors" of routing information is carried in the AFI model as multiprotocol extensions. As such, each address family we define will have its own BGP database and can be managed via BGP policies on the basis of individual address families. We can think of SAFIs as subsets or sub-configurations under a parent AFI that allow us to nest more refined and granular BGP policies under the address family context.

Limitations in the older NLRI format used by IPv4 BGP have been overcome by the AFI model enhancements offered by MP-BGP. Some of these limitations included:

- No support for AFI and SAFI configuration information. Many new BGP (and other protocols such as MPLS) features are supported only in AFI and SAFI configuration modes and cannot be configured in NLRI configuration modes.
- No support for IPv6. A router that is configured in the NLRI format cannot establish peering with an IPv6 neighbor.
- Limited support for multicast inter-domain routing. In the NLRI format, not all configuration options are available and there is no support for VPNv4. The NLRI format configurations can be more complex than configurations that support the AFI model.

The new AFI model found in MP-BGP as we mentioned earlier supports multiple AFIs and SAFIs, all NLRI-based commands and policy configurations, and offers backward compatibility with routers that support only the older format. A router that is configured using this new AFI model has the following capabilities and features:

- AFI and SAFI information and configurations are supported. A router that is configured using the AFI model can carry routing information for multiple network layer protocol address families (for example, IPv4 and IPv6).
- AFI configuration is similar in all address families, making the CLI syntax easier to use than the NLRI format syntax.
- All BGP routing policy capabilities and commands are supported.

- Virtual Private Networks (VPNs) and VPN routing and forwarding (VRF) instances are supported. Unicast IPv4 for VRFs can be configured from a specific address family IPv4 VRF; this configuration update is integrated into the BGP VPNv4 database.

The BGP address family model as far as our discussion are concerned consists of three address families: VPNv4, IPv4 vrf and IPv4.

VPNv4

We have discussed the nature of the VPNV4 address and its structure. This 96-bit address is not even close to a traditional IPv4 address so needless to say a vehicle had to be created that would allow the exchange of these prefixes via MP-BGP. The VPNv4 address family was created to satisfy that specific requirement. The VPNv4 address family is identifies routing sessions for protocols such as BGP that support these standard VPNv4 address prefixes. These VPNv4 routes are treated the same as IPv4 routes, but they are not structurally the same. Remember the RD is prepended to the ipv4 address in order to support the notion that ip address in a large deployment may not be unique.

VPNs, when used with MPLS, allow several sites to transparently interconnect through a service provider's network. One service provider network can support several different IP VPNs. Each of these appears to its users as a private network, separate from all other networks. Within a VPN, each site can send IP packets to any other site in the same VPN. Each VPN is associated with one or more VPN VRFs. The router maintains a separate routing and Cisco Express Forwarding (CEF) table for each VRF. This prevents information from being sent outside the VPN and allows the same subnet to be used in several VPNs without causing duplicate IP address problems. The router using MP-BGP distributes the VPN routing information using the BGP route-target extended communities we have been discussing.

The VPN address space is isolated from the global address space by design. BGP distributes reachability information for VPN-IPv4 prefixes for each VPN using the VPNV4 multiprotocol extensions to ensure that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other. This is the fundamental role of the import process we discussed in the section of the same name.

Additional, it should be pointed out that VPNv4 routes will almost always have a label associated with each route.

These entities are created with the **address-family vpnv4** command under the MP-BGP routing process. In its simplest form this address family is responsible for iBGP prefix and label exchange between PE devices. It often referred to as the VPNV4 tunnel.

IPv4

The IPv4 address family is used to identify routing sessions for protocols such as BGP that use standard IP version 4 address prefixes. Unicast or multicast address prefixes can be specified within the IPv4 address family. Routing information for address family IPv4 unicast is advertised by default when a BGP peer is configured unless the advertisement of unicast IPv4 information is explicitly turned off.

These entities are created with the **address-family ipv4** command (or via the default behavior of MP-BGP) under the MP-BGP routing process. In its simplest form this address family is responsible for managing the native BGP sessions for standard IPv4.

IPv4 VRF

VRF instances can also be associated with IPv4 AFI configuration mode commands.

These entities are created with the **address-family ipv4 vrf <vrf>** command under the MP-BGP routing process. In its simplest form this address family is responsible enabling prefix exchange between the PE and CE within a given VRF

Route Reflectors

Route Reflectors have been used for years as tools to advertise router that where learned through an iBGP session to another iBGP neighbor. This tool overrides the problem where normal BGP operation prevents the exchange of routing information between iBGP peers as a loop preventions mechanism. For the purpose of our discussion any router that advertises iBGP peer to other iBGP peers will be a route reflector. These devices help eliminate the need for a full mesh topology in an autonomous system. Prefixes advertise by these devices are known as reflected routers.

MP-iBGP domains become more manageable and scalable with the incorporation of Route Reflectors. We need to keep I mind that even with the use of route reflectors there is still an upper limit to the number of MP-iBGP peerings that can be managed by a given device. For this reason it is suggested that route reflectors be deployed in a redundant group or administratively load sharing arrangement. Route reflector offer a number of benefits but it must be noted that these benefits come a at cost. In this instance the cost is measured in device resources like CPU utilization and memory consumption.

The most important element in this concept is the knowledge that as we increase the number of PE devices in our route reflected topology the total number of neighbors and the number of total reflected prefixes increase almost geometrically. This means that the associated CPU load and memory utilizations will also increase it a similar fashion. So at various points out networks growth and expansion it might become necessary to increase the number of Router Reflectors to better distribute the resource utilization.

Outside of all these details we have to keep in mind that the purpose of our MPLS Layer 3 VPN and the MP-BGP process that enables it is the advertisement of routes that are external to the PE devices. This encompasses interfaces that are running in particular VRF instances on a given PE as well as the networks attached to the CE devices. This means for functionality there will normally be no need for a route reflector because none of this routing information is required by the service providers core network. It could be necessary or desirable in some scenarios to have a core device act as a route reflector when the decision has been made to have a core router reflect routes between geographically regions. This means that the intra-regional connectivity can be maintain if an individual link fails. This is the primary benefit a route reflector in the service provider backbone can offer. Additionally, it should be noted that Route Reflectors in this type of deployment can be configured specify the particular route-

target values that they will reflect. As such they will need to have the send-community extended feature enabled to operate correctly.

Convergence

Convergence in MPLS is measured in the time it takes for the data traffic from the remote CE to reach the local CE after a topology change has occurred in the network. Logically we need to keep in mind that the larger our network the longer overall convergence is going to be. There are two types of convergence: up convergence and down convergence.

Up Convergence

Up convergence in Layer 3 VPNs can be defined as the time it takes for traffic to be restored between VPN sites when a new prefix is advertised and propagated from the local CE to the remote CE, or when a new site comes up on the VPN. This type of convergence is applicable in cases where there is a backup link which comes up only after the primary link has gone down or when some sort of conditional advertisement is configured. A good working definition of up convergence for the purpose of our discussions is the time needed to advertise a route from CE to CE.

Down Convergence

Down convergence is a measurement of how long it takes for traffic to be rerouted on an alternate path due to a failure in either the service provider network, the customer network or the primary PE-CE link connecting the two. Down convergence for the purpose of our discussions is the time it takes for a best path determination to be withdrawn from a given CE device.

MP-BGP Common Issues

The primary issues that affect both MP-BGP and MPLS fall into just a handful of operational domains. We have discussed the necessary parameters and enhancements made to MP-BGP such that it will support the concept and exchange of the VPNv4 prefixes. So as a general rule, and to keep the failure domains associated to the individual topic of this book we will focus on the aspects of MP-BGP that drive and enable the Layer 3 VPN used between PE devices. We will however change up our lab topology such that it will include a centrally located route reflector. This is important because we need to understand the exact role this type of device will play in the Layer 3 VPN and the issues that can be introduced.

The main issues that MP-BGP brings us fall into two categories: MP-BGP Misconfiguration and Route Reflector Issues. We will explore each of these categories in the sample topology illustrated in Figure 5-1.

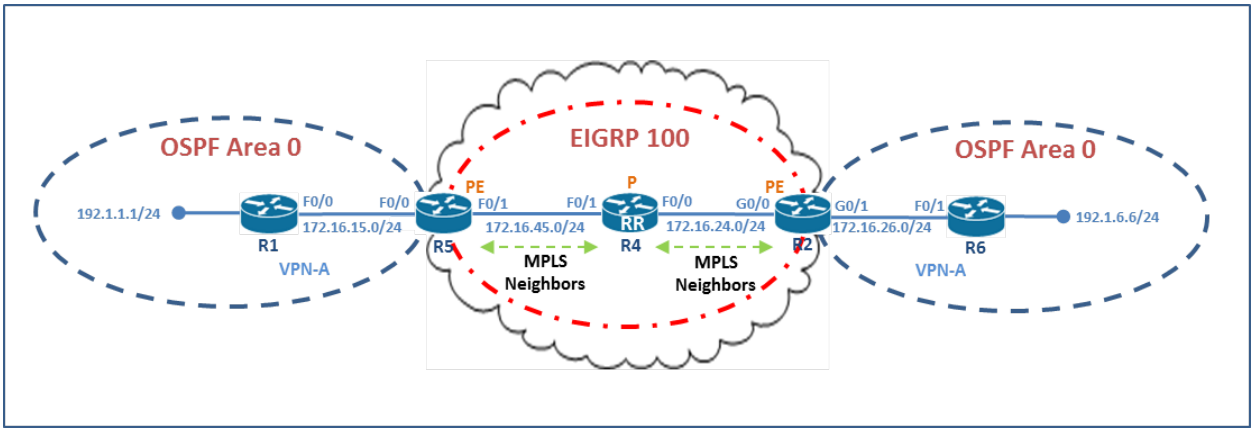


Figure 5-1: MP-BGP Topology with a Route Reflector

In this topology R4 is working as a route reflector within the service provider cloud. We will explore each failure category in turn.

MP-BGP Misconfiguration

There are a number of MP-BGP misconfiguration categories that can affect MPLS Layer 3 VPNs. We will discuss two such categories of misconfiguration that will most commonly appear in topologies.

Missing VPNV4 Address Family

In the event that the VPNV4 address-family command has not been applied under the MP-BGP routing process we will not create the VPNV4 tunnel. If this command is missing or if there are no neighbors configured under it well will not even attempt to initiate the VPNV4 tunnel. This can be seen clearly in the output of the **show ip bgp vpnv4 all summary** command. First we will remove the VPNV4 address-family from R5 and then we will observe the effect.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#no  address-family vpnv4
R5(config-router)#en
```

```
%BGP-5-ADJCHANGE: neighbor 192.1.4.4 Down Neighbor deleted
R5(config-router)#end
R5#
```

Note that the moment we remove the address-family command we see that the messages notifies us that the neighbor 192.1.4.4 is down because it was deleted. We will using show ip bgp vpnv4 all summary command now to see what the output looks like on R5.

```
R5#show ip bgp vpnv4 all sum

R5#
```

Note that there is absolutely no information provided. If we execute the same command on R4 we will see the following:

```
R4#show ip bgp vpnv4 all sum
BGP router identifier 192.1.4.4, local AS number 65000
BGP table version is 17, main routing table version 17
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 2) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 6/4 prefixes, 9/7 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.2.2       4        65000      91      92       17    0    0 00:20:48        2
192.1.5.5       4        65000      63      65        0    0    0 00:00:11 Idle
```

The output notifies us immediately that the VPNV4 connection is not up with the neighbor 192.1.5.5 (R5) on R4. Just to observe this process we will add the address-family back on R5 and verify in progression as we modify the configuration.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)# address-family vpnv4
```

Before we activate the neighbor under the address-family we will verify if any information appears with the show ip bgp vpnv4 all summary command.

```
R5(config-router-af)#do show ip bgp vpnv4 all summary

R5(config-router-af)#
```

There is still no output. Now we will activate the neighbor. Remember in this configuration we are peering with R4 not with R2 directly.

```
R5(config-router-af)#neighbor 192.1.4.4 activate
```

```
%BGP-5-ADJCHANGE: neighbor 192.1.4.4 Up
R5(config-router-af)#
```

We see the neighbor comes up with 192.1.4.4, but what about the output of the **show ip bgp vpnv4 all summary** command now.

```
R5(config-router-af)#do show ip bgp vpnv4 all summary
BGP router identifier 192.1.5.5, local AS number 65000
BGP table version is 31, main routing table version 31
4 network entries using 624 bytes of memory
4 path entries using 272 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 2) using 32 bytes of memory
BGP using 1480 total bytes of memory
BGP activity 12/8 prefixes, 14/10 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000      70      67       31    0    0 00:00:05        2
R5(config-router-af)#
```

Now we see that the peering is up and we are learning/receiving 2 prefixes via the link. What prefixes are we learning? We can discover that with the **show ip bgp vpnv4 all neighbor 192.1.4.4 routes** command.

```
R5#show ip bgp vpnv4 all neighbor 192.1.4.4 routes
BGP table version is 33, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?

Total number of prefixes 4
```

We are learning about the routes 172.16.26.0/24 and 192.1.6.0/24. The question is do we have reachability to these prefixes from R1? We can verify with a ping test to 192.1.6.6.

```
R1#ping 192.1.6.6 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
Packet sent with a source address of 192.1.1.1
!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This indicates that the configuration is working.

VPNV4 Not Sending Extended Communities

We have to point out that the ability to send extended communities under the address-family VPNV4 is automatically the default in IOS. We can observe this via the **show run | sec address-family vpnv4** command.

```
R5#show run | sec address-family vpnv4
 address-family vpnv4
  neighbor 192.1.4.4 activate
  neighbor 192.1.4.4 send-community extended
```

We did not add this command in our configuration. The send-community extended capability was created the moment we activated the neighbor under the address family. This default behavior is something that can become a pitfall. Before the Layer 3 VPN can function it is necessary to be able to send the route-target community attributes. Remember that without these values the MPLS import process cannot operate. This means that we will send the prefixes but they will never be installed in the tables on the remote PE. We will observe this behavior by manually removing the **neighbor 192.1.4.4 send-community extended** command on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)# address-family vpnv4
R5(config-router-af)#no neighbor 192.1.4.4 send-community extended
R5(config-router-af)#end
```

With the configuration missing we will know take a close look at the Layer 3 VPN. We will start with the **show ip bgp vpnv4 all summary** command.

```
R5#show ip bgp vpnv4 all sum
BGP router identifier 192.1.5.5, local AS number 65000
BGP table version is 33, main routing table version 33
6 network entries using 936 bytes of memory
6 path entries using 408 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 2) using 32 bytes of memory
BGP using 1928 total bytes of memory
BGP activity 14/8 prefixes, 16/10 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000      93      90       33    0    0 00:23:19        2
```

The VPNV4 tunnel to R4 is up. Now we know that we are not sending the extended community values to R4. What exactly does that mean to the operation of our VPN. To find out we will skip R4 and move

directly to R2 and look to see what routes arrive and what routes are imported. First we will use the **show ip bgp vpnv4 all summary** command to look at the status of the VPNV4 tunnel.

```
R2#show ip bgp vpnv4 all summary
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 21, main routing table version 21
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 16/14 prefixes, 16/14 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000     137     133       21    0    0 01:10:05        0
```

This tells us that we are not importing any routes. Just as a verification we will move to R4 to see if that device is sending any prefixes to R2.

```
R4#show ip bgp vpnv4 rd 100:1 neighbors 192.1.2.2 advertised-routes
BGP table version is 21, local router ID is 192.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?

Total number of prefixes 2
```

R4 is advertising the prefixes to R2 with a next hop of R5, but R2 is not importing them. This is because there is no route-target value assigned to these prefixes. We can see that on R4 by using the **show ip bgp vpnv4 all 192.1.1.0** command.

```
R4#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 20
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    192.1.5.5 (metric 156160) from 192.1.5.5 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      mpls labels in/out nolabel/506
```

Observe that there is no RT value. This means that as the prefix arrives on R2 it will not be imported because there is no RT value. We will demonstrate the significance of this by going to R6 and proving that we do not have reachability to the address 192.1.1.1.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
U.U.U
Success rate is 0 percent (0/5)
```

We will now enable the send of the extended community values on R5 and track the prefixes across the service provider cloud.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family vpnv4 unicast
R5(config-router-af)#neighbor 192.1.4.4 send-community extended
R5(config-router-af)#end
```

Now we will return to R4 and look again at the route 192.1.1.0/24.

```
R4#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 22
Paths: (1 available, best #1, no table)
Flag: 0xA00
  Advertised to update-groups:
        2
  Local, (Received from a RR-client)
    192.1.5.5 (metric 156160) from 192.1.5.5 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/506
```

Note that the extended community value is attached. The next question is will the prefixes now appear on R2?

```
R2#show ip bgp vpnv4 all
BGP table version is 25, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

We see the prefixes are now in the table on R2. We can see the RT value assigned using the **show ip bgp vpnv4 rd 100:1 192.1.1.0** command.

```
R2#show ip bgp vpnv4 rd 100:1 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 23
Paths: (1 available, best #1, no table)
Flag: 0x820
  Not advertised to any peer
  Local
    192.1.5.5 (metric 158720) from 192.1.4.4 (192.1.4.4)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      Originator: 192.1.5.5, Cluster list: 192.1.4.4
      mpls labels in/out nolabel/506
```

This community value of 100:1 matches the import route-target established for the ip vrf VPN-A instance. We can see this with the **show ip vrf detail | sec Import** command.

```
R2#show ip vrf detail | sec Import
  Import VPN route-target communities
    RT:100:1
```

As such the prefixes from 192.1.4.4 will be imported. We know this because the address 192.1.1.1 is now reachable from R6.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Route Reflector Issues

In situations where the design of the network does include a Route Reflector there are a number of issues where this design can be problematic. We have discussed the issue regarding the sending of the extended community values in the previous section. We find ourselves having to focus on a few elements. Beyond the simple structure of the route-reflector-client command we have to look at the operational role of a route reflector. Created for the purpose of reflecting iBGP routes such that we do not have to create fully meshed environments a route reflector also participates in the Layer 3 VPN. These devices are responsible for “forwarding” the route-target values as well. The operational parameters of the route reflection process to not extend to communities, this is compensated by the default behavior of the address family VPNV4 sub-context the same way it was regarding basic MPLS Layer 3 VPN operation in the previous section. We will explore this process by removing the address family VPNv4 configuration on R4 and then observe the configuration requirements necessary to restore the Layer 3 VPN. For the purpose of our discussion these issues regarding router reflectors will distill down to three categories: Missing VPNV4 Address Family, Missing Route Reflector Configuration and Missing Send Community Extended Commands.

Missing VPNV4 Address Family

We will start with an operational environment again as we discuss this particular issue. We will see the effects of removing the **address-family vpnv4 unicast** configuration and then explore the effects via ping and show commands.

```
R6#ping 192.1.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This shows that the configuration is working as we would expect. We will remove the **address-family vpnv4 unicast** command.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#router bgp 65000
R4(config-router)# address-family vpnv4
R4(config-router-af)#no address-family vpnv4
R4(config-router)#
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Down Neighbor deleted
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Down Neighbor deleted
R4(config-router)#
```

Once the address-family is removed we clearly see that the neighbor relationships with the PEs go down because they are now deleted. We can see the effect first by trying to ping 192.1.1.1 from R6 again.

```
R6#ping 192.1.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
```

We see that there is no reachability between PEs in this network now. This is because no prefixes are being exchanged between the PEs. Removing the address family vpnv4 configuration we have completely broken the MPLS Layer 3 VPN. We can see this with the show ip bgp vpnv4 all sum command on the three devices in the provider core.

```
R2#show ip bgp vpnv4 all summary
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 29, main routing table version 29
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 2) using 28 bytes of memory
```

```
BGP using 836 total bytes of memory
BGP activity 20/18 prefixes, 20/18 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000     309     304        0    0    0 00:04:52 Active
```

The active state indicates that the neighbor relationship is actively trying to form. The issue here is that it will never come up, because R4 is no longer capable of peering with either R2 or R5. We can see this via the same command executed on R4.

```
R4#show ip bgp vpnv4 all summary

R4#
```

Just as we saw before the absence of any information at all from this command indicates the absence of the **address-family vpnv4 unicast** command on that device. We can see that the absence of this configuration has disrupted the formation of the VPNV4 peers on both ends of the provider cloud.

```
R5#show ip bgp vpnv4 all summary
BGP router identifier 192.1.5.5, local AS number 65000
BGP table version is 37, main routing table version 37
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 2) using 28 bytes of memory
BGP using 836 total bytes of memory
BGP activity 14/12 prefixes, 16/14 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000     272     269        0    0    0 00:09:28 Active
```

Again we see the Active state. The VPNV4 tunnel will never come up, until we add the configuration back. Remember that we will need to activate both neighbors under the address family once we create it.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#router bgp 65000
R4(config-router)#address-family vpnv4 unicast
R4(config-router-af)#neighbor 192.1.5.5 activate
R4(config-router-af)#neighbor 192.1.2.2 activate
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Up
R4(config-router-af)#neighbor 192.1.2.2 activate
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Up
R4(config-router-af)#end
```

We need to look to see if any configuration has been added by default. We will do this via the **show run | sec address-family vpnv4** command.

```
R4#show run | sec address-family vpnv4
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
  neighbor 192.1.5.5 activate
  neighbor 192.1.5.5 send-community extended
```

Be sure to note that the send-community extended commands that are so critical to the workings of the route-reflector clients in our topology have been added automatically by the IOS. However, this will not restore reachability in our network. We can verify this via the ping on R6 once more.

```
R6#ping 192.1.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
```

Missing Route Reflector Configuration

The reason this is not going to restore reachability is based squarely for the router reflector function missing on R4. Without this command not even R4 will learn the prefixes advertised by R5. We will look first to demonstrate that R5 is advertising the two prefixes 192.1.1.0/24 and 172.16.15.0/24 via the **show ip bgp vpnv4 all neighbor 192.1.4.4 advertised-routes** command.

```
R5#show ip bgp vpnv4 all neighbors 192.1.4.4 advertised-routes
BGP table version is 37, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

R5 is advertising the prefixes, but as we expect R4 will not have any knowledge of them in its database.

```
R4#show ip bgp vpnv4 all

R4#
```

What would happen if we add the router-reflector-client command for only one neighbor? We will make R5 a route-reflector-client and observe what happens.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
R4(config)#router bgp 65000
R4(config-router)#address-family vpnv4 unicast
R4(config-router-af)#neighbor 192.1.5.5 route-reflector-client
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Down RR client config change
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Up
R4(config-router-af)#
```

Now we will verify what prefixes R4 is learning (if any).

```
R4#show ip bgp vpnv4 all
BGP table version is 5, local router ID is 192.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?
```

Note that we are now learning routes (all four routes to be exact). The question remains, “is this sufficient to allow end-to-end reachability between the two VPN sites?” We can find out by pinging 192.1.1.1 while sourcing it from R6 loopback.

```
R6#ping 192.1.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

We have reachability. Just the ensure we have good design we will add the route-reflector-client command for R2 before we move on to the next fault domain.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#  router bgp 65000
R4(config-router)#  address-family vpnv4 unicast
R4(config-router-af)#  neighbor 192.1.2.2 route-reflector-client
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Down RR client config change
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Up
R4(config-router-af)#
```

Missing Send-Community Extended Commands.

We noted the fact that the **send-community extended** command was added to R4 under the **address-family vpnv4 unicast** context by default. We are going to beg the question, “what happens if the send-

community configuration is removed from the route reflector?" We will find out by removing those configurations and then explore the results with pings and show commands.

```
R4#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#  router bgp 65000
R4(config-router)#  address-family vpnv4 unicast
R4(config-router-af)#no neighbor 192.1.5.5 send-community extended
R4(config-router-af)#no neighbor 192.1.2.2 send-community extended
R4(config-router-af)#end
```

Once this is accomplished we will ping 192.1.1.1 from R6 to see if we have reachability across the Layer 3 VPN.

```
R6#ping 192.1.1.1 source loopback 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
R6#
```

We have no reachability to the networks in question. We will move to R5 and look at the routes that are advertised to the route-reflector and there extended community RT values.

```
R5#show ip bgp vpnv4 all neighbors 192.1.4.4 advertised-routes
BGP table version is 61, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

Are the prefixes being sent with the extended community: RT:100:1?

```
R5#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 29
Paths: (1 available, best #1, table VPN-A)
  Advertised to update-groups:
        1
  Local
    172.16.15.1 from 0.0.0.0 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:100:1
      mpls labels in/out 506/nolabel
```

The next question is do the arrive on R4 with that same extended community value?

```
R4#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 4
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        1
  Local, (Received from a RR-client)
    192.1.5.5 (metric 156160) from 192.1.5.5 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/506
```

The prefix is advertised to R2 from R4.

```
R4#show ip bgp vpnv4 rd 100:1 neighbor 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?

Total number of prefixes 2
```

Are these prefixes being learned on R2?

```
R2#show ip bgp vpnv4 all
BGP table version is 29, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2 (default for vrf VPN-A)
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
R2#
```

R2 is learning these prefixes, but we have to ask if the route-target value matches.

```
R4#show ip bgp vpnv4 all 192.1.1.0
BGP routing table entry for 100:1:192.1.1.0/24, version 4
Paths: (1 available, best #1, no table)
  Advertised to update-groups:
        1
  Local, (Received from a RR-client)
    192.1.5.5 (metric 156160) from 192.1.5.5 (192.1.5.5)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:1
      mpls labels in/out nolabel/506
```

We need to look at R2 now to see if the prefixes are imported into the database.

```
R2#show ip bgp vpnv4 all 192.1.1.1
% Network not in table
```

On R2 we will activate the **debug ip bgp vpnv4 unicast import** and the **debug ip bgp rib-filter** features to observe the process that is taking place. We will then reset the bgp process and monitor the results.

```
R2#debug ip bgp vpnv4 unicast import
Tag VPN import processing debugging is on
R2#debug ip bgp rib-filter
BGP Rib filter debugging is on
R2#
BGP- ATF: Debuging is ON
R2#
R2#clear ip bgp *
R2#
%BGP-5-ADJCHANGE: neighbor 192.1.4.4 Down User reset
R2#
%BGP-5-ADJCHANGE: neighbor 192.1.4.4 Up
R2#
BGP- ATF: T 172.16.26.6/32 (1) c=0x70D4F44C EVENT Track stop
BGP- ATF: T 172.16.26.6/32 (1) c=0x70D4F44C Removing
BGP- ATF: R 172.16.26.0/24 (1) -> Gi0/1 0.0.0.0 Scheduled to check for deletion
BGP- ATF: R 172.16.26.0/24 (1) -> Gi0/1 0.0.0.0 Deleting
VPN(5): Scanning for import check is done.
R2#
BGP- ATF: EVENT 172.16.26.6/32 (1) Track start
BGP- ATF: 172.16.26.6/32 (1) Adding track
BGP- ATF: EVENT Query 172.16.26.6/32 (1) found route
BGP- ATF: 172.16.26.0/24 (1) Adding route
BGP- ATF: R 172.16.26.0/24 (1) -> Updating route
BGP- ATF: R 172.16.26.0/24 (1) -> Gi0/1 0.0.0.0 Notifying
BGP- ATF: T 172.16.26.6/32 (1) skip notify (32 > 24)
BGP- ATF: Notifying 172.16.26.6/32 (1)
BGP- ATF: T 172.16.26.6/32 (1) c=0x70CB6C70 Adding to client notification queue
R2#
BGP(4): Import walker start version 0, end version 3
BGP(4): ... start import cfg version = 0
vpn(4): Start import processing for: 100:2:172.16.26.0/24
vpn(4): Import check for VPN-A; flags match, import
vpn(4): Import for VPN-A permitted; import flags match, import
vpn(4): Same RD import for VPN-A
vpn(4): VPN-A same RD import, do best path
vpn(4): 100:2:172.16.26.0/24 (ver 4), imported as:
vpn(4): 100:2:172.16.26.0/24 (ver 4)
vpn(4): Start import processing for: 100:2:192.1.6.0/24
vpn(4): Import check for VPN-A; flags match, import
vpn(4): Import for VPN-A permitted; import flags match, import
vpn(4): Same RD import for VPN-A
vpn(4): VPN-A same RD import, do best path
R2#
vpn(4): 100:2:192.1.6.0/24 (ver 5), imported as:
```

```
vpn(4): 100:2:192.1.6.0/24 (ver 5)
```

This monitoring the import process we have been discussing. Please note that the prefixes for 172.16.15.0/24 and 192.1.1.0/24 are not part of the process. This is due to the absence of the route-target value. We will add the send-community extended command on R4 and return to this debug process.

```
R4(config)#router bgp 65000
R4(config-router)#address-family vpnv4 unicast
R4(config-router-af)#neighbor 192.1.5.5 send-community extended
R4(config-router-af)#neighbor 192.1.2.2 send-community extended
R4(config-router-af)#end
```

Now on R2 watch what happens. Give the slow “up convergence” associated with MP-BGP it may take some time for the results to begin to appear.

```
R2#
BGP- ATF: EVENT 192.1.5.5/32 (0) Track start
BGP- ATF: 192.1.5.5/32 (0) Adding track
BGP- ATF: EVENT Query 192.1.5.5/32 (0) found route
BGP- ATF: 192.1.5.0/24 (0) Adding route
BGP- ATF: R 192.1.5.0/24 (0) -> Updating route
BGP- ATF: R 192.1.5.0/24 (0) -> Gi0/0 172.16.24.4 Notifying
BGP- ATF: T 192.1.5.5/32 (0) skip notify (32 > 24)
BGP- ATF: Notifying 192.1.5.5/32 (0)
BGP- ATF: T 192.1.5.5/32 (0) c=0x70CB6C5C Adding to client notification queue
R2#
BGP(4): Import walker start version 5, end version 7
BGP(4): ... start import cfg version = 0
vpn(4): Start import processing for: 100:1:172.16.15.0/24
vpn(4): Import check for VPN-A; flags match
vpn(4): Import for VPN-A permitted; import flags match
BGP(4): Prefix 100:1:172.16.15.0/24 to be imported as 100:2:172.16.15.0/24 --
Permitted
nexthop 192.1.5.5, origin ?, localpref 100, metric 0, originator 192.1.5.5,
clusterlist 192.1.4.4, extended community RT:100:1
Path added
vpn(4): 100:1:172.16.15.0/24 (ver 6), imported as:
vpn(4): 100:2:172.16.15.0/24 (ver 8)
vpn(4): Start import processing for: 100:1:192.1.1.0/24
vpn(4): Import check for VPN-A; flags match
R2#
vpn(4): Import for VPN-A permitted; import flags match
BGP(4): Prefix 100:1:192.1.1.0/24 to be imported as 100:2:192.1.1.0/24 -- Permitted
nexthop 192.1.5.5, origin ?, localpref 100, metric 0, originator 192.1.5.5,
clusterlist 192.1.4.4, extended community RT:100:1
Path added
vpn(4): 100:1:192.1.1.0/24 (ver 7), imported as:
vpn(4): 100:2:192.1.1.0/24 (ver 9)
BGP- ATF: EVENT 172.16.15.0/24 RIB update UP
BGP- ATF: EVENT 192.1.1.0/24 RIB update UP
```


There is a lot of output to look at now. We see that the two prefixes miraculously appear, we see that the attribute information about the route is visible to include the intact extended community value. There is one thing that should be pointed out here. We mentioned in the section of this book that discussed the MPLS Layer 3 VPN import process what happens when the RD values do not match. This output clearly illustrates that process. Note that both prefixes arrived in the format 100:1:X.X.X.X/24 but they each where imported using the local RD value with the new format of 100:2:X.X.X.X/24.

We will want to look on more time at the impact of this process. We will do so with the **show ip bgp vpnv4 all** command.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24    192.1.5.5                0    100      0 ?
*>i192.1.1.0         192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24    192.1.5.5                0    100      0 ?
*> 172.16.26.0/24    0.0.0.0                  0         32768 ?
*>i192.1.1.0         192.1.5.5                0    100      0 ?
*> 192.1.6.0         172.16.26.6              0         32768 ?
R2#
```

This explains why the prefixes appear twice in the table; once with the original RD that they were learned with, and then with the local default RD that was used during the import process.

After this point we will transition our discussions to the operational mechanism that allow and facilitate PE to CE communications. Any further discussion from this chapter forward will assume a fully operational label switched path, a correctly configured MPLS Layer 3 VPN and route reflector operations. We will only interact with the MP-BGP routing processes when it becomes necessary to redistribute the vrf aware IGPs prefixes into it.

Chapter Challenge: MP-BGP Sample Trouble Tickets

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH4-MPBGP-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 5-2 below:

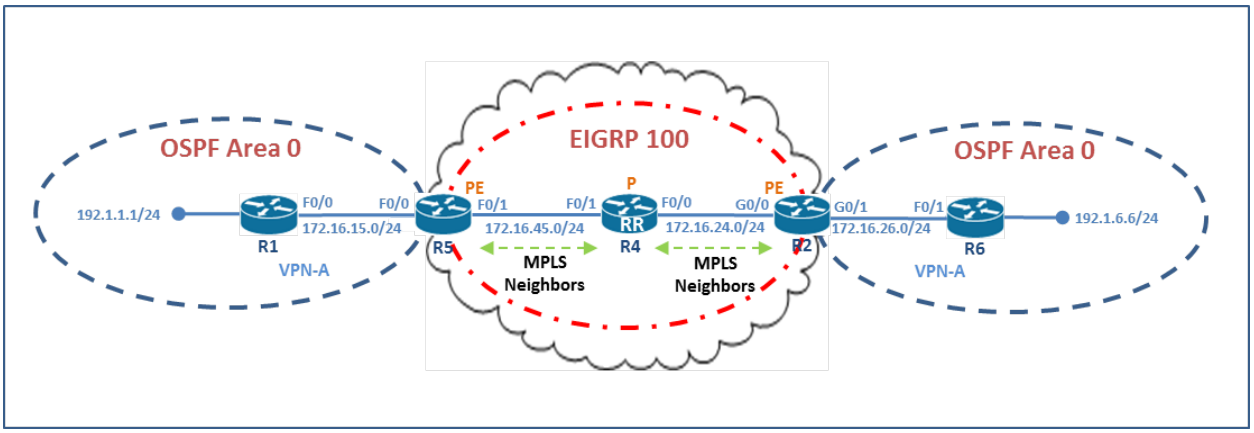


Figure 4-3: The Chapter Challenge Topology

Trouble Ticket #1

Your manager has brought to your attention the fact that the MP-BGP VPNV4 tunnel between R5 and R2 is not forming. You have been instructed to isolate why this is happening and take the necessary actions needed to correct this problem. When this issue has been corrected the output of the show ip bgp vpnv4 all summary command on R2 will match the output in Exhibit "A".

Exhibit "A"

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 5, main routing table version 5
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000       5       6        5    0    0 00:01:32        2
```

Chapter Challenge: Layer 3 VPN Sample Trouble Tickets Solutions

The following section includes the solutions to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

Your manager has brought to your attention the fact that the MP-BGP VPNV4 tunnel between R5 and R2 is not forming. You have been instructed to isolate why this is happening and take the necessary actions needed to correct this problem. When this issue has been corrected the output of the show ip bgp vpnv4 all summary command on R2 will match the output in Exhibit "A".

Exhibit "A"

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 5, main routing table version 5
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000       5       6        5    0    0 00:01:32        2
```

Step 1 - Fault Verification:

Does the output of the show ip bgp vpnv4 all sum command match the exhibit?

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 5, main routing table version 5
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000      39      39        5    0    0 00:35:17        0
```

The Layer 3 VPN tunnel is formed but we are learning no prefixes where the exhibit states that we should have two (2).

Step 2 - Fault Isolation:

The next course of action is to use the **show ip bgp vpnv4 all summary** command on R5 to see the status of that end of the tunnel.

```
R5#show ip bgp vpnv4 all sum
BGP router identifier 192.1.5.5, local AS number 65000
BGP table version is 5, main routing table version 5
2 network entries using 312 bytes of memory
2 path entries using 136 bytes of memory
2/1 BGP path/bestpath attribute entries using 336 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 840 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000      39       41        5    0    0 00:37:46        0
```

The verification clearly demonstrates that R5 is participating in the VPN. Is R5 advertising prefixes?

```
R5#show ip bgp vpnv4 all neighbors 192.1.4.4 advertised-routes
BGP table version is 5, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

We see that the R5 is advertising two prefixes to R4. Remember that according to our diagram R4 is a Route Reflector. We need to see if R4 is learning these prefixes.

```
R4#show ip bgp vpnv4 all neighbors 192.1.5.5 routes
BGP table version is 9, local router ID is 192.1.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?

Total number of prefixes 2
```

Is R4 advertising there prefixes to R2?

```
R4#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes

Total number of prefixes 0
```

What would stop R4 from advertising prefixes to R2? Looking at the diagram we would expect R4 to be configured as a router reflector per the diagram. To see if this is true we will use the **show run | sec router bgp** command.

```
R4#sh run | sec router bgp
router bgp 65000
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 neighbor 192.1.5.5 remote-as 65000
 neighbor 192.1.5.5 update-source Loopback0
 !
 address-family ipv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.5.5 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
  neighbor 192.1.5.5 activate
  neighbor 192.1.5.5 send-community extended
 exit-address-family
```

We see there is no route reflector configuration applied to R4.

Step 3 - Fault Remediation:

In this scenario, the **route-reflector-client** command should be applied under the MP-BGP process on R4.

```
R4(config)#
%SYS-5-CONFIG_I: Configured from console by console
R4(config)#router bgp 65000
R4(config-router)#address-family ipv4
R4(config-router-af)#neighbor 192.1.2.2 route-reflector
R4(config-router-af)#neighbor 192.1.2.2 route-reflector
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Down RR client config change
R4(config-router-af)#neighbor 192.1.5.5 route-reflector
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Down RR client config change
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Up
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Up
R4(config-router-af)#exit-address-family
R4(config-router)#address-family vpnv4 unicast
R4(config-router-af)#neighbor 192.1.2.2 route-reflector
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Down RR client config change
```

```
R4(config-router-af)#neighbor 192.1.5.5 route-reflector
R4(config-router-af)#
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Down RR client config change
R4(config-router-af)#exit
%BGP-5-ADJCHANGE: neighbor 192.1.5.5 Up
R4(config-router-af)#exit-address-family
R4(config-router)#end
%BGP-5-ADJCHANGE: neighbor 192.1.2.2 Up
R4#
```

We see the BGP process come up after the RR client configuration change is applied.

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 9, main routing table version 9
6 network entries using 936 bytes of memory
6 path entries using 408 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP rrinfo entries using 24 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1928 total bytes of memory
BGP activity 6/0 prefixes, 7/1 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.4.4       4        65000       8       7        9    0    0 00:04:09        2
```

The issue has been corrected.

Chapter 6: Static

Static PE-CE routing is possibly the most common routing technique used in MPLS VPNs today. This is because it proves itself ideal in client site environments with a single CE to PE peering, or in client networks with only a small group of subnets.

Introduction to Static PE-CE Routing

At this juncture in our discussions we are now going to take all the information we have discussed to date and begin to implement various communication solutions between our PE and CE devices. Based on the fact that we have spent the first half of this book working with MPLS, Label distribution protocol, Layer 3 VPNs and MP-BGP we are going to assume that everything between the PE devices is configured and operating correctly. That means in all subsequent chapters in the book we will only deal with the routing protocol running between the PE and CE and any of its specific behaviors, capabilities or shortfalls. We will first use static routes between the PE and CE.

To create a working mpls environment we will configure a static default route on R1 pointing to the next hop ip address on the shared link between R1 and R5.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip route 0.0.0.0 0.0.0.0 172.16.15.5
```

Once the default static route has been assigned on R1 we will create vrf aware static routes to the network 192.1.1.0/24 on R1 such that we will have reachability to that network from R5. These static routes will need to be redistributed into the ipv4 addresses family that governs the vrf VPN-A. This is accomplished in the next configuration steps.

```
R5#
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#ip route vrf VPN-A 192.1.1.0 255.255.255.0 172.16.15.1
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute static
R5(config-router-af)#exit-address-family
R5(config-router)#end
```

Now that this has been accomplished we will need to go to R2 and perform the same process there.

```
R2#
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#ip route vrf VPN-A 192.1.6.0 255.255.255.0 172.16.26.6
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute static
R2(config-router-af)#exit-address-family
R2(config-router)#end
```

Lastly R6 will need a static default route to reach any prefixes no assigned to R6 as connected or physical interfaces.

```
R6#conf t
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ip route 0.0.0.0 0.0.0.0 172.16.26.2
```

While we are on R6 we will perform some testing. Initially we will attempt to ping the network 192.1.1.0/24 by using the ping utility.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Note that this test fails. The test fails because R1 has no knowledge regarding how to reach the source address of 172.16.26.6. We would have to redistribute that information into the MP-BGP process. What would happen if we originated the ping from the 192.1.6.6 instead?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This ping is successful. It works because we have advertised the NRLI to R5 via the VPNv4 tunnel we created previously. We can demonstrate how this process works by taking an additional step to redistribute the connected vrf enabled interfaces into the MP-BGP process on both R2 and R5. This will prevent us from having to specify the loopback addresses as sources for subsequent testing. The procedures to accomplish this are as follows. We will start on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute connected
R2(config-router-af)#end
```

Then we will repeat the process on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute connected
R5(config-router-af)#end
```

Now we should be able to successfully ping without specifying the source.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
```



```
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This testing tells us unequivocally that the Layer 3 VPN is working. In fact we can see that this is the case via the **show ip bgp vpnv4 all summary** command. This command will give us valuable information regarding the VPNv4 tunnel . Specifically in this instance it will tell us that it is up and that we are learning 2 prefixes from the other side of the MPLS network.

```
R2#show ip bgp vpnv4 all sum
BGP router identifier 192.1.2.2, local AS number 65000
BGP table version is 9, main routing table version 9
6 network entries using 936 bytes of memory
6 path entries using 408 bytes of memory
3/2 BGP path/bestpath attribute entries using 504 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory
BGP using 1904 total bytes of memory
BGP activity 6/0 prefixes, 6/0 paths, scan interval 15 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.1.5.5       4        65000     148     148        9    0    0 02:24:27        2
```

After manually performing this configuration we can clearly anticipate that the two prefixes that R2 is learning will be 192.1.1.0/24 and 172.16.15.0/24. These being the prefixes we redistributed into the MP-BGP process, and furthermore we should expect that the next hop address will be that of the PE neighbor because this is an MP-iBGP peering.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

This is not the first time that we have explored this process. As you probably remember we made the configuration for this network in ***Chapter 4: MPLS L3VPN***. In that chapter we were only interested in generating routes so that we could demonstrate that the Layer 3 VPN was working as expected. Now however, we want to take just few moments to study limited options this method of configuration provides us.

Rather than making reference to options it might be best to mention the fact that this method is least feature rich method of PE-CE routing. But even as such static routing is possibly the most common routing technique used in MPLS VPNs today. This is because it proves itself ideal in client site environments with a single CE to PE peering, or in client networks with small groups of subnets. Another very significant benefit is that static PE to CE routing prevents the possibility of the customer inadvertently corrupting the service providers routing table. Actually, this “protection” works both ways, meaning that the service provider has little to no capability of flooding false routes to the customer as well. The service provider has complete control over the clients routing capabilities.

The drawback to this method comes in the form of administrative overhead for the service provider. This is true because there is no dynamic functionality at all. Where even the most basic of routing protocols would automatically provide rerouting features during periods of network failure or when new prefixes are added to the network the static approach mandates that the service provider take manual control to ensure reachability through the cloud is maintained.

Static PE-CE Common Issues

The primary issues that affect both PE-CE static routing configurations fall into just a handful of operational domains. As a general rule, and to keep the failure domains associated to the individual topic of this chapter we will focus on the aspects of static routing as they apply to the PE – CE communication process.

The main issues that PE-CE Static Routing brings us fall into three categories: Missing or Incorrect Static Routes, Missing or Incorrect Default Static routes and Missing or Incorrect Redistribution. We will explore each of these categories in the sample topology illustrated in Figure 6-1.

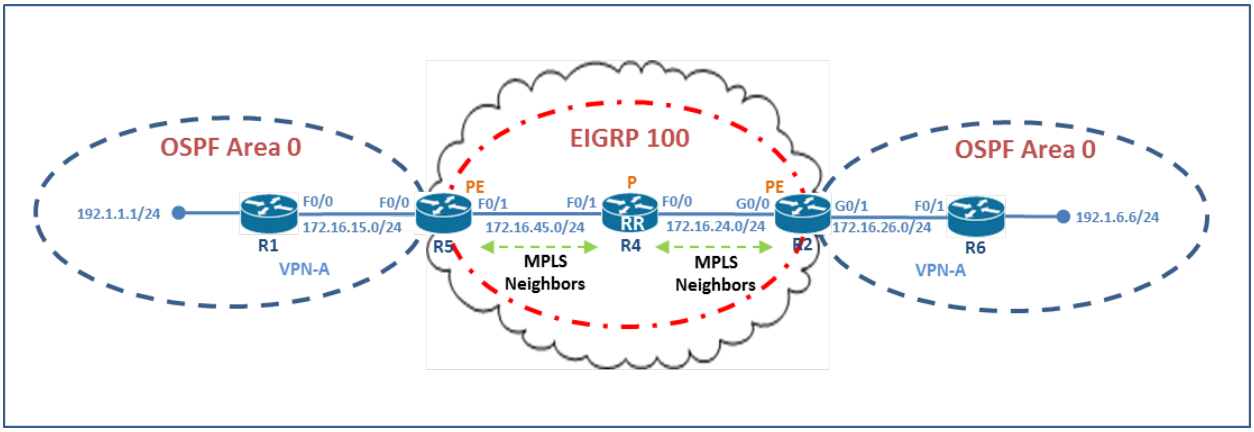


Figure 6-1: Static PE-CE Topology with a Route Reflector

In this topology R4 is working as a route reflector within the service provider cloud. We will explore each failure category in turn. Remember that for the purposes of our discussions we will consider the service provider cloud to be operation.

Missing or Incorrect Static Routes

To demonstrate the effect that a missing Static route would have on our topology we will remove the individual static route on R1 and then perform some tests. Using the information from these show and ping command we will walk through the troubleshooting process that we recommend to use to identify these issues. Before we do anything we will verify that R1 can reach the address 192.1.6.6 while sourced from 192.1.1.1.

```
R1#ping 192.1.6.6 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
Packet sent with a source address of 192.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

We will remove the static route on R5 now.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#
%SYS-5-CONFIG_I: Configured from console by console
```

```
R5(config)#no ip route vrf VPN-A 192.1.1.0 255.255.255.0 172.16.15.1
R5(config)#end
```

We will see now that R1 cannot reach 192.1.6.6.

```
R1#ping 192.1.6.6 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
Packet sent with a source address of 192.1.1.1
.....
Success rate is 0 percent (0/5)
```

The issue here is that we are not communicating anything regarding how to reach R1's loopback 0 interface to either PE-PEs. We will follow this process starting from R6 to prove this point. We will use traceroute on R6 to 192.1.1.1.

```
R6#traceroute 192.1.1.1

Type escape sequence to abort.
Tracing the route to 192.1.1.1

  1 172.16.26.2 0 msec 0 msec 0 msec
  2 172.16.26.2 !H  *  !H
```

The trace stops at R2. We will go there and find out why. We will explore the vrf VPN-A routing table to find out if there is any record of the prefix.

```
R2#show ip route vrf VPN-A

Routing Table: VPN-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.26.0 is directly connected, GigabitEthernet0/1
B       172.16.15.0 [200/0] via 192.1.5.5, 01:46:36
S    192.1.6.0/24 [1/0] via 172.16.26.6
```

We have a static route to reach 192.1.6.6 but there is no route to 192.1.1.1. Oddly enough there is a route to reach 172.16.15.0/24. This is the link between the PE and CE. Can we ping that interface?

```
R2#ping vrf VPN-A 172.16.15.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.15.1, timeout is 2 seconds:
```

```
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

This means that we have reachability to some prefixes on R1 but not others. How does R2 learn about these prefixes? The answer is via the Layer 3 VPN between R5 and R2. We will take a look and see what information we are learning from R5.

```
R2#show ip bgp vpnv4 all
BGP table version is 15, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

We are learning about 172.16.15.0/24 but not 192.1.1.1. We need to go to R5 to learn why. We will explore the problem with the **show ip bgp vpnv4 all neighbor 192.1.4.4 advertised-routes** command.

```
R5#show ip bgp vpnv4 all neighbor 192.1.4.4 advertised-routes
BGP table version is 37, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?

Total number of prefixes 1
```

We are only advertising 1 prefix. We will take a look at the contents of the vrf VPN-A routing table and see what routes can be found there.

```
R5#show ip route vrf VPN-A

Routing Table: VPN-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
```

```
B       172.16.26.0 [200/0] via 192.1.2.2, 01:56:02
C       172.16.15.0 is directly connected, FastEthernet0/0
B    192.1.6.0/24 [200/0] via 192.1.2.2, 01:56:02
```

We see that we have no static route to reach 192.1.1.1 like we would expect. We can add this back to the configuration of R5 and restore connectivity. But first we will demonstrate that R5 has no reachability to 192.1.1.1.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

There is no reachability.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#ip route vrf VPN-A 192.1.1.0 255.255.255.0 172.16.15.1
R5(config)#end
```

Now we will repeat the test.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Just as a final verification we will look to see if R2 is learning 2 routes now, and then test from R1 as we did in the first portion of the discussion.

```
R2#show ip bgp vpnv4 all
BGP table version is 17, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?
*> 192.1.6.0        172.16.26.6              0         32768 ?
```

R2 is learning both routes from R5. The next question is, “have we restored connectivity?”

```
R1#ping 192.1.6.6 source lo 0
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.6.6, timeout is 2 seconds:
Packet sent with a source address of 192.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

As we can see reachability has been restored.

Missing or Incorrect Default Static Routes

The only reachability information to be found in our current CE topology will come in the form of the static default route we have configured on each. This route can be seen via the **show ip route** command.

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 172.16.15.5 to network 0.0.0.0

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.15.0 is directly connected, FastEthernet0/0
C    192.1.1.0/24 is directly connected, Loopback0
S*   0.0.0.0/0 [1/0] via 172.16.15.5
R1#
```

It is this default route that will allow any traffic destined to any address not found on R1 to exit the FastEthernet0/0 interface. Without this static route no traffic will ever leave R1 or R6. This can be seen by removing the static route and using the traceroute utility.

```
R1#traceroute 192.1.6.6

Type escape sequence to abort.
Tracing the route to 192.1.6.6

  1  *  *  *
  2  *  *  *
  3  *  *  *
  4  *  *  *
  5  *  *  *
  6  *  *  *
  <output omitted>
```

We will restore our configuration and move on.

```
R1#traceroute 192.1.6.6

Type escape sequence to abort.
Tracing the route to 192.1.6.6
```

```
  1 172.16.15.5 0 msec 4 msec 0 msec
  2 172.16.45.4 [MPLS: Labels 401/204 Exp 0] 4 msec 0 msec 4 msec
  3 172.16.26.2 [MPLS: Label 204 Exp 0] 0 msec 0 msec 4 msec
  4 172.16.26.6 0 msec *  0 msec
```

Missing or Incorrect Redistribution

We have demonstrated that an absence of or incomplete static routing information on either end of the PE-CE connection results in a failure of our MPLS domain. When it comes to troubleshooting any PE-CE routing issue we will always confirm that the PE can reach the networks of the CE and vice-versa. Then once this is confirmed we have to look at the redistribution process. However, in a static routing arrangement between PE and CE it is only necessary to redistribute the static routes into the MP-BGP process between the individual PE devices. Again we will remove the redistribute commands from one device and observe the results. We will focus on R5 for this demonstration.

```
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no redistribute static
R5(config-router-af)#end
R5#
```

This will be as far as we will go into the Layer 3 VPN. First we will note that we have reachability between the PE and the CE.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Now we will move to R2 and see if we can ping.

```
R2#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

We are not learning about 192.1.1.0/24 via R5 because of the lack of redistribution. We can see this via the output of the **show ip bgp vpnv4 rd 100:1 neighbor 192.1.4.4 routes** command.

```
R2#show ip bgp vpnv4 rd 100:1 neighbor 192.1.4.4 routes
BGP table version is 19, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
```

```
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?

Total number of prefixes 1
```

Now we are only learning the one route as we expected. If we restore the redistribute command this will return to 2 prefixes.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute static
R5(config-router-af)#end
R5#
```

Now back to R2.

```
R2#show ip bgp vpnv4 rd 100:1 neighbor 192.1.4.4 routes
BGP table version is 21, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                0    100      0 ?

Total number of prefixes 2
```

We have restored reachability in our network as evidenced by the successful pings on R6.

```
R6#ping 192.1.1.1 repeat 10

Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 1/2/4 ms
```

Chapter Challenge: Static Sample Trouble Tickets

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH6-STATIC-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 6-2 below:

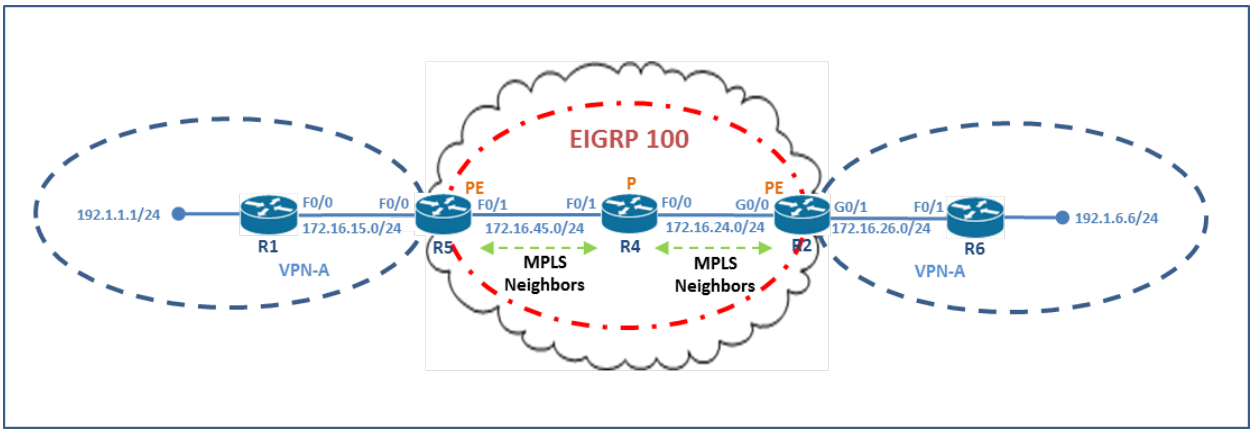


Figure 6-2: The Chapter Challenge Topology

Trouble Ticket #1

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
```

Chapter Challenge: Layer 3 VPN Sample Trouble Tickets Solutions

The following section includes the solutions to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Does the ping to 192.1.1.1 work or fail on R6?

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
U.U.U
Success rate is 0 percent (0/5)
```

The pings are unsuccessful.

Step 2 - Fault Isolation:

First we will make sure that R5 can reach 192.1.1.1.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R5#
```

R5 can reach the address in question. Now we need to see if R5 is advertising this prefix to the remote PE (R2).

```
R5#show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes
BGP table version is 7, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?

Total number of prefixes 1
```

We see that R5 is not advertising the prefix. We need to see why this is taking place. We know that the static routes should be redistributed into the MP-BGP process so we will start there.

```
router bgp 65000
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 192.1.2.2 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  redistribute connected
  redistribute static route-map FILTER
  no synchronization
 exit-address-family
```

We see that there is a route-map titled FILTER that is attached to the redistribution process. We can see what values are configured in the route-map with the **show route-map** command.

```
R5#show route-map
route-map FILTER, deny, sequence 10
  Match clauses:
    ip address (access-lists): 1
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
route-map FILTER, permit, sequence 20
  Match clauses:
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
```

Observe that sequence 10 says to deny anything that matches access-list 1 from being redistributed. We can see what this entails via the **show access-list** command.

```
R5#show access-list 1
Standard IP access list 1
    10 permit 192.1.1.0, wildcard bits 0.0.0.255 (6 matches)
```

We see that the address 192.1.1.0/24 matches access-list 1, and anything that match this access-list will be denied by the route-map.

Step 3 - Fault Remediation:

In this scenario, we will edit the access-list such that it will not allow 192.1.1.1 to be filtered by the route-map. To do this will change the logic from permit to deny and see if that remediates the issue with the network.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#ip access-list standard 1
R5(config-std-nacl)#no 10
R5(config-std-nacl)#10 deny 192.1.1.0 0.0.0.255
R5(config-std-nacl)#end
```

Now we will look to see if the route is in the VPNV4 table now.

```
R5#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2                0    100      0 ?
```

We see the route in the table now. Before we move to the final remediation step we will see if R5 is advertising this prefix to R2 now.

```
R5#show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              0         32768 ?

Total number of prefixes 2
```

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R6#
```

The issue has been corrected.

Chapter 7: RIPv2

RIPv2 employs address-families to manage virtual routing and forwarding instances. Just like in MP-BGP we will enable the **address-family ipv4 vrf <name>** under the router context. RIPv2 will be our simplest PE-CE dynamic routing protocol and as such it will be a great place to start learning the mechanisms required to support IGP protocols in unison with MPLS.

Introduction to RIP PE-CE Routing

When it comes to MPLS Layer 3 VPN we will always use MP-BGP as the "carrier" for our traffic, but it must be noted that this lack of options does not extend to our CE-PE routing decisions. There are a wide array of IGP protocols that are VRF aware. The issue that we have to keep in mind when we use these dynamic options is that each of these protocols can and more often than not will run independently. This means the very nature of our MPLS VPN environment with its reliance on MP-BGP to exchange routing information via redistribution is tailored to preserve some very valuable routing protocol information that would be lost without the enhancements that have been made to MP-BGP. These exact features vary from protocol to protocol. In this chapter we will explore the simplest dynamic protocol available to us: RIP Version 2.

RIPv2 employs address-families to manage virtual routing and forwarding instances. This configuration and behavior is very close to the concepts we discussed in ***Chapter 5: MP-iBGP*** in how they are implemented. Just like in MP-BGP we will enable the address-family ipv4 vrf <name> under the router context. At this time we can configure the network command under the address family. It is important to understand that parameters like timers, version, and auto-summarization will apply to all VRFs when they are applied in this manner.

Just like we explored when we worked with static routing between the PE and CE device we will rely on redistribution here as well. The VRF aware instance of a given RIP process will need to be redistributed into MP-BGP so that these prefixes will be advertised via the Layer 3 VPN tunnel to any remote PEs. However, in order to ensure reachability between all prefixes in our topology it will be necessary to perform mutual redistribution between RIPv2 and MP-BGP. This concept introduces a number of new elements that we will explore in this chapter.

We will explore this configuration in by deploying RIPv2 as the CE-PE routing protocol according to Figure 7-1.

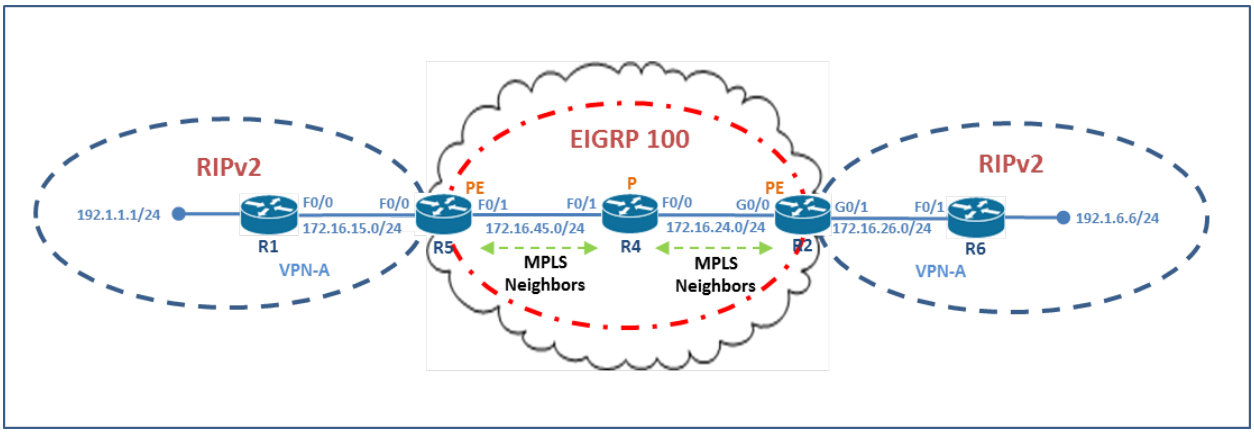


Figure 7-1: RIPv2 CE-PE Routing Topology

VRF Aware RIPv2

We will start this process first on R1 and R5, by creating the RIPv2 VRF aware process on R5.

```
R5(config)#
R5(config)#router rip
R5(config-router)#version 2
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#version 2
R5(config-router-af)#no auto-summary
R5(config-router-af)#network 172.16.0.0
R5(config-router-af)#end
R5#
```

We will go to R1 and create the RIP version process on that router as well. Please not that we are not using anything special in this process. We will not create an address-family on R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto
R1(config-router)#network 172.16.0.0
R1(config-router)#network 192.1.1.0
R1(config-router)#end
```

Now if we did this correctly R5 will have a route to the network 192.1.1.0/24.

```
R5#sh ip route vrf VPN-A rip
R    192.1.1.0/24 [120/1] via 172.16.15.1, 00:00:22, FastEthernet0/0
```

We see the route exists, but the final test is to successfully ping the address 192.1.1.1 from R5.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

We will repeat this process on R2 and R6.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#network 172.16.0.0
R2(config-router-af)#end
```

Now for the normal routing process on R6.

```
R6#conf t
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router rip
R6(config-router)#version 2
R6(config-router)#no auto-summary
R6(config-router)#network 172.16.0.0
R6(config-router)#network 192.1.6.0
R6(config-router)#end
```

Again we will look to see if R2 has learned the prefix for 192.1.6.0/24 via RIPv2.

```
R2#show ip route vrf VPN-A rip
R    192.1.6.0/24 [120/1] via 172.16.26.6, 00:00:13, GigabitEthernet0/1
```

Now that we have created the individual RIPv2 routing processes between devices it is going to be necessary to redistribute these RIP routes into the MP-BGP process under the **address-family ipv4 vrf VPN-A** context. This will be accomplished simply via the **redistribute rip** command.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute rip
R2(config-router-af)#end
```

The same process will need to be deployed on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute rip
R5(config-router-af)#end
```

We have just advertised the prefixes into the Layer 3 VPN. We can see that they are being communicated by using the **show ip bgp vpnv4** all command.

```
R5#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*> 192.1.1.0        172.16.15.1              1         32768 ?
*>i192.1.6.0        192.1.2.2                1    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2                1    100      0 ?
```

Now to repeat the process on R2.

```
R2#show ip bgp vpnv4 all
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                1    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5                1    100      0 ?
*> 192.1.6.0        172.16.26.6              1         32768 ?
```

We see that the PEs know about the routes originated on either side of the Layer 3 VPN. There is one element missing from our topology that will create a functional network. If we look at R1 and R6 we will see that these devices have no knowledge of the routes on either end of the network. For instance R1 has no knowledge of either of the routes 172.16.26.0/24 and 192.1.6.0/24.

```
R1#show ip route rip

R1#
```

We will repeat the process on R6 where we will no knowledge of the routes 192.1.1.0/24 and 172.16.15.0/24.

```
R6#sh ip route rip

R6#
```

Based on the fact that we have already discussed the need for mutual redistribution between RIPv2 and MP-BGP, and the knowledge that we have not redistributed the MP-BGP routes into RIPv2 on either R5 or R2 we should not expect the CE to have any routes to each other. This is what we are seeing now. In the next section we will address this shortfall in our topology.

Redistribution of MP-BGP into VRF Aware RIPv2

We will now go to R5 and redistribute the contents of the MP-BGP routing table into the VRF aware instance of RIPv2. First we will look at the contents of the MP-BGP routing process on R5. The goal is to identify the exact prefixes that will be redistributed into the **address-family ipv4 vrf VPN-A** process.

```
R5#show ip bgp vpnv4 rd 100:1 neighbors 192.1.2.2 routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
```

```
*>i192.1.6.0         192.1.2.2                1    100      0 ?

Total number of prefixes 2
```

Redistribution with Seed Metric

We will redistribute MP-BGP into RIPv2 on R5, the net result will be the appearance of 192.1.6.0/24 and 172.16.26.0/24 in R1's routing table for the RIPv2 process. We will specify a seed metric of 10 when we do this.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router rip
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute bgp 65000 metric 10
R5(config-router-af)#end
```

We will no look at R1 to see if the routes have been injected into the routing table for RIPv2.

```
R1#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.26.0 [120/10] via 172.16.15.5, 00:00:08, FastEthernet0/0
R    192.1.6.0/24 [120/10] via 172.16.15.5, 00:00:08, FastEthernet0/0
```

We see the routes are now present and we see the metric of each is a value of 10. Now we will perform the redistribution on R2, but this time we will use the metric transparent value.

Metric Transparent

The **metric transparent** command allows us to redistribute our MP-BGP prefixes into the VRF aware RIPv2 process while preserving the RIPv2 learned metric. We will observe first the prefixes that we learn via the Layer 3 VPN.

```
R2#show ip bgp vpnv4 rd 100:2 neighbors 192.1.5.5 routes
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                1    100      0 ?

Total number of prefixes 2
```

We see the prefixes that we will inject into the routing table of R6 but this time we will use the seed metric value of transparent.

```
R6#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.15.0 [120/1] via 172.16.26.2, 00:00:29, FastEthernet0/1
R    192.1.1.0/24 [120/2] via 172.16.26.2, 00:00:29, FastEthernet0/1
```

Note that we have two different metrics. These metrics are preserved as the redistributed packets make their way through the Layer 3 VPN. We will artificially increase these metrics and look again on R6 to see the overall effect.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router rip
R1(config-router)#offset-list 0 out 8 FastEthernet 0/0
R1(config-router)#end
```

Now we will look at R6 again.

```
R6#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.15.0 [120/1] via 172.16.26.2, 00:00:24, FastEthernet0/1
R    192.1.1.0/24 [120/10] via 172.16.26.2, 00:00:24, FastEthernet0/1
```

Please observe that the prefix 192.1.1.0/24 now has a metric of 10.

RIPv2 PE-CE Common Issues

The primary issues that affect PE-CE RIPv2 routing configurations fall into just a handful of operational domains. As a general rule, and to keep the failure domains associated to the individual topic of this chapter we will focus on the aspects of RIPv2 VRF aware routing as they apply to the PE – CE communication process.

The main issues that PE-CE Static Routing brings us fall into three categories: RIPv2 Configuration Errors, Redistribution Issues and Filtering Issues. We will explore each of these categories in the sample topology illustrated in Figure 7-2.

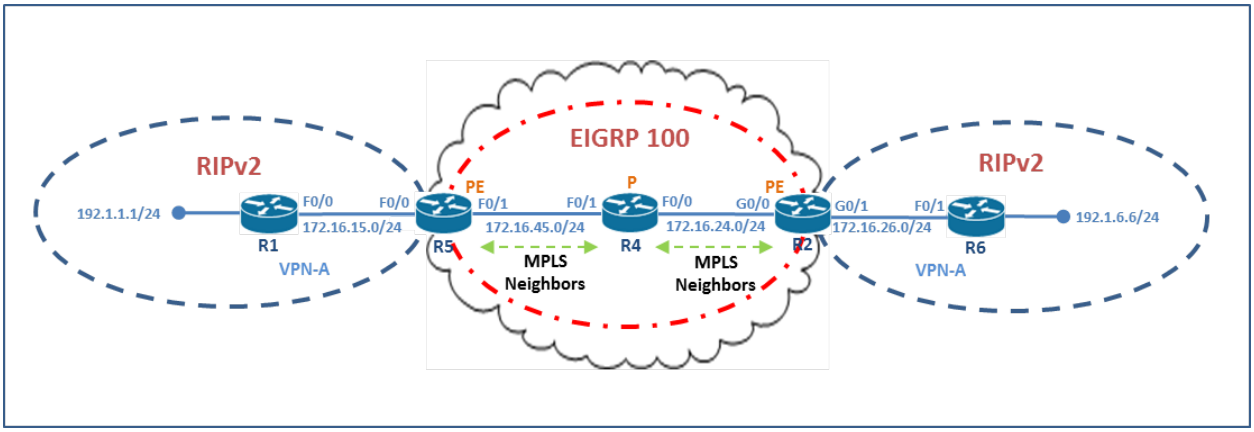


Figure 7-2: RIPv2 PE-CE Topology

We will explore each failure category in turn. Remember that for the purposes of our discussions we will consider the service provider cloud to be operational.

RIPv2 Configuration Errors

There are actually two areas where RIPv2 configuration issues can impact the performance of our MPLS domain. Primarily these two locations are at the CE and at the PE. We will look at the simplest of these locations first.

CE RIPv2

At the CE we will use the standard RIPv2 routing protocol that we have used throughout out studies. This is simple distance vector routing. Due to the fact that the CE router has no knowledge of the concept of the virtual routing and forwarding instance we have no level of complication introduced at the phase. The typical issues associated with RIPv2 is to forget configure the version, turn of auto-summary, or to leave out the network commands.

PE RIPv2

VRF aware routing is required at the PE in order to get RIPv2 to work as the routing protocol between the PE and CE devices while simultaneously supporting the Layer 3 VPN essential to our service provider cloud. In this case the most common problem is to forget to configure the VRF instance with the **address-family ipv4 vrf VPN-A** or to misconfigure the network command.

We can explore what happens when this happens, by removing the command from R5 and observing the results on R1 and R6.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router rip
R5(config-router)# address-family ipv4 vrf VPN-A
R5(config-router-af)#no network 172.16.0.0
R5(config-router-af)#network 172.15.0.0
R5(config-router-af)#end
```

In this example we are misconfiguring the network statement. This will result in a loss of prefix exchange between the PE and CE as demonstrated via the **show ip route** command on R1.

```
R1#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.26.0 [120/10] via 172.16.15.5, 00:02:39, FastEthernet0/0
R    192.1.6.0/24 [120/10] via 172.16.15.5, 00:02:39, FastEthernet0/0
```

Oddly enough we see the routes in the table on R1, but keep in mind that this is RIP we are working with and it will take a long time for the routes to expire without changing the timer values or clearing the routing table manually.

```
R1#clear ip route *
R1#show ip route rip

R1#
```

After manually clearing the process we see that the routes from R6 are not present. But what is happening on R6?

```
R6#show ip route rip

R6#
```

We are not learning any routes here, because they are not being originated into the Layer 3 VPN by the redistribution process. We will restore the correct network commands on R5 and see that the topology comes back up.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router rip
R5(config-router)#add
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no network 172.15.0.0
R5(config-router-af)#network 172.16.0.0
R5(config-router-af)#end
```

Once this is accomplished our routes will return to both R1 and R6.

```
R1#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
```



```
R       172.16.26.0 [120/10] via 172.16.15.5, 00:00:24, FastEthernet0/0
R    192.1.6.0/24 [120/10] via 172.16.15.5, 00:00:24, FastEthernet0/0
```

On R6 we see the same.

```
R6#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.15.0 [120/1] via 172.16.26.2, 00:00:25, FastEthernet0/1
R    192.1.1.0/24 [120/10] via 172.16.26.2, 00:00:25, FastEthernet0/1
```

We also have restored reachability as a result.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Redistribution Issues

As we have mentioned earlier it is a necessity to use redistribution in our topology for all PE-CE routing protocols other than eBGP peering. So this means that we need an understanding of the issues related to the incompatibility of metrics used by our individual protocols. This extends to both our MP-BGP and RIPv2 protocols. We will explore the nature of these issues as they relate to RIPv2 in our topology by altering the configuration on our PEs. These aspects as the relate to redistribution will impact both our IGP Redistribution and MP-BGP Redistribution.

IGP Redistribution

We will explore the issue related to redistribution and our VRF aware IGPs by looking at some of the possible problems related to the configuration process. As an example we will alter the configuration on R5 under the router rip configuration such that we will not specify a seed metric.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router rip
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no redistribute bgp 65000
R5(config-router-af)#redistribute bgp 65000
R5(config-router-af)#end
```

When we go to R1 to look at the results we see that there are no routes being injected.

```
R1#show ip route rip

R1#
```

Another option would be to redistribute the routes into RIPv2 using a seed metric that was too high. We will redistribute the prefixes using a seed metric of 15 and see what happens.

```
R5#conf t
```



```
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router rip
R5(config-router)# version 2
R5(config-router)# redistribute bgp 65000
R5(config-router)# !
R5(config-router)# address-family ipv4 vrf VPN-A
R5(config-router-af)#  redistribute bgp 65000 metric 15
```

We will go to R1 and see the results of this configuration.

```
R1#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.26.0 [120/15] via 172.16.15.5, 00:00:27, FastEthernet0/0
R    192.1.6.0/24 [120/15] via 172.16.15.5, 00:00:27, FastEthernet0/0
```

In this instance the metric used to inject the prefixes is sufficient to allow the routes to make it to R1 but in this instance none of these prefixes could be advertised to a neighbor within the RIPv2 domain at the customer site.

MP-BGP Redistribution

Regarding redistribution of the IGP prefixes into the MP-BGP protocol the most common issues that can be encountered fall into one of two categories.

Lack of Redistribution – When redistribution is not used under the MP-BGP process, we will never see any prefixes advertised to the remote PE devices. We can demonstrate this by removing the redistribution command and using the show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no redistribute rip
R5(config-router-af)#end
```

We can clearly see that we are not advertising prefixes any longer.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes

Total number of prefixes 0
```

Use of an Artificially High Metric – When the redistribution process is used and a maximum value seed metric is assigned it is impossible for the route to be redistributed into the VRF aware RIPv2 process. We can see this process by modifying the metric to the maximum value.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute rip metric ?
  <0-4294967295>  Default metric
```

```
R5(config-router-af)#redistribute rip metric 4294967295
R5(config-router-af)#end
```

We can see that the prefix is advertised to the remote PE with the newly assigned max metric.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 15, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1     4294967295         32768 ?

Total number of prefixes 2
```

We will go to R2 and look at the R5 and see if the prefix is found in the table as being learned from R5.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 15, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1     4294967295         32768 ?

Total number of prefixes 2
```

The prefix arrives on R5 but is it sent to R6 as a RIPv2 route?

```
R6#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.26.0 is directly connected, FastEthernet0/1
R       172.16.15.0 [120/1] via 172.16.26.2, 00:00:02, FastEthernet0/1
C    192.1.6.0/24 is directly connected, Loopback0
```

Observe that there is no record of the prefix 192.1.1.0/24 only the connected prefix 172.16.15.0/24. This is because the prefix was not valid at the time of redistribution because of the maximum metric used when it was redistributed.

Filtering Issues

As with any routing protocol, filtering whether it is in the form of access-lists, offset-lists, route-maps and distribute-lists can block prefixes from being allowed to enter the routing table of a device. VRF aware routing protocols like RIPv2 are no exception. These tools are can very easily be used to induce issues associated with our CE-PE routing. As an example we will use an offset list on R1 to filter out all routes being learned from R5.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router rip
R1(config-router)#offset-list 0 in 15 FastEthernet 0/0
R1(config-router)#end
```

Note that now we have no routes appearing on R1.

```
R1#show ip route rip

R1#
```

There are a number of tools that can be used to affect routes like the offset command. At this point it is more important to note that issue can be generated using them than to experiment with each of them.

Chapter Challenge: RIPv2 Sample Trouble Tickets

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH7-RIP-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 7-3 below:

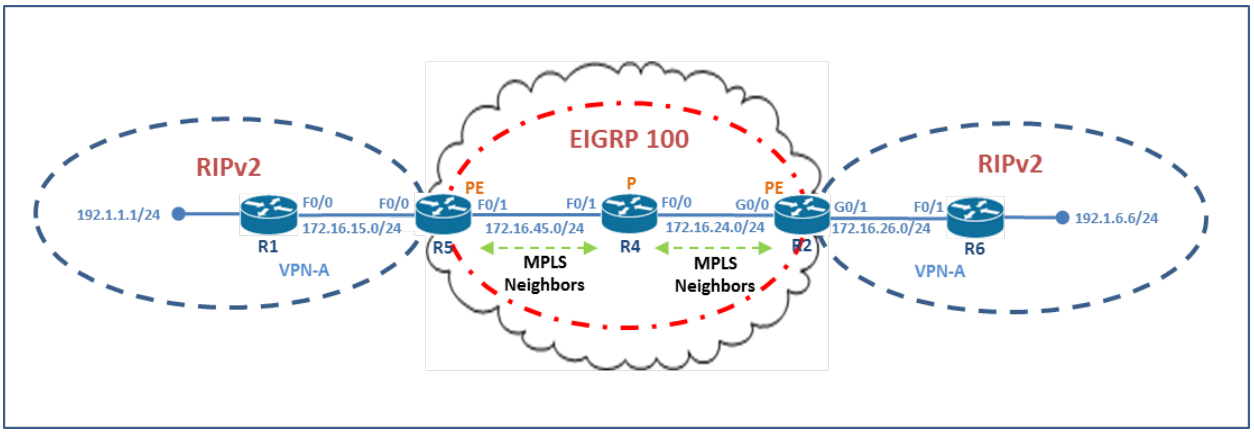


Figure 7-3: The Chapter Challenge Topology

Trouble Ticket #1

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Chapter Challenge: RIPv2 Sample Trouble Tickets Solutions

The following section includes the solutions to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Does the ping to 192.1.1.1 work or fail on R6?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

The pings are unsuccessful.

Step 2 - Fault Isolation:

First we will make sure that R5 can reach 192.1.1.1.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

R5 cannot reach the address in question. Now we need to see if R1 is advertising this prefix to the local PE (R5).

```
R1#show ip protocol
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 3 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
```

```
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    FastEthernet0/0       2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    172.16.0.0
    192.1.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    172.16.15.5           120      00:00:16
  Distance: (default is 120)
```

We see that R1 is not advertising the prefix 192.1.1.0. We need to see why this is taking place.

```
R1#show run | sec router rip
router rip
 version 2
 network 172.16.0.0
 network 192.1.0.0
 no auto-summary
```

We see that this network statement is incorrect it should read network 192.1.1.0.

Step 3 - Fault Remediation:

In this scenario, we will edit the network statement such that it will advertise the correct address.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router rip
R1(config-router)#network 192.1.1.0
R1(config-router)#end
```

Note that our instructions do not allow us to remove configuration so we will leave the original network statement in place. Now we will look to see if the route makes it to R5's RIPv2 routing table for vrf VPN-A.

```
R5#show ip route vrf VPN-A

Routing Table: VPN-A
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.26.0 [200/0] via 192.1.2.2, 00:08:50
```

```
C       172.16.15.0 is directly connected, FastEthernet0/0
B    192.1.6.0/24 [200/1] via 192.1.2.2, 00:08:50
R    192.1.1.0/24 [120/1] via 172.16.15.1, 00:00:06, FastEthernet0/0
```

We see the route in the table now. Before we move to the final remediation step we will see if R5 is advertising this prefix to R2 now (remember we know we have multiple issues in this topology).

```
R5#show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes

Total number of prefixes 0
```

R5 is not advertising any prefixes to R2. We know that these prefixes make their way to into the MP-BGP routing process via redistribution. So the easiest method to confirm this would be to look at the output of the **show run | sec bgp** command.

```
R5#show run | sec bgp
  redistribute bgp 65000 metric 5
router bgp 65000
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 192.1.2.2 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  no synchronization
 exit-address-family
```

We can see that there is no redistribution of RIPv2 under the **address-family ipv4 vrf VPN-A** context. We will take measures to add this configuration.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute rip
R5(config-router-af)#end
```

With this accomplished R5 should now advertise the prefixes to R2.

```
R5#show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
```

```
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              1         32768 ?

Total number of prefixes 2
```

We see that they are being advertise, but is R2 accepting these prefixes?

```
R2#show ip bgp vpnv4 rd 100:1 neighbor 192.1.5.5 routes
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5                1    100      0 ?

Total number of prefixes 2
```

The prefixes are being accepted. Are they being redistributed into the RIPv2 vrf VPN-A instance?

```
R6#show ip route rip
     172.16.0.0/24 is subnetted, 2 subnets
R       172.16.15.0 [120/5] via 172.16.26.2, 00:00:21, FastEthernet0/1
R    192.1.1.0/24 [120/5] via 172.16.26.2, 00:00:21, FastEthernet0/1
```

Based on this we should have full reachability.

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R6#
```

The issues have been corrected.

Chapter 8: EIGRP

EIGRP as a PE-CE routing protocol is used by service providers for customers who use EIGRP as their IGP routing protocol and, hence, prefer to use EIGRP to exchange routing information between the customer sites across an MPLS VPN backbone.

Introduction to EIGRP PE-CE Routing

EIGRP as a PE-CE routing protocol is used by service providers for customers who use EIGRP as their IGP routing protocol and, hence, prefer to use EIGRP to exchange routing information between the customer sites across an MPLS VPN backbone. In an MPLS VPN environment, to achieve this, the original EIGRP metrics must be carried inside MP-BGP updates as we just discussed. This is achieved by using MP-BGP extended community attributes to carry and preserve EIGRP metrics when crossing the MP-iBGP domain. These communities define the native characteristics associated with EIGRP, such as the AS number or EIGRP cost metric like bandwidth, delay, load, reliability, and MTU.

Route propagation in MPLS VPN networks using EIGRP PE-CE routing is based on the EIGRP AS configured on the PE routers. In an MPLS VPN environment, the EIGRP AS can be the same on all PE routers or different on all PE routers.

Step 1. Enable the global EIGRP routing process

Enable the global EIGRP routing process on PE routers (R2 and R3).

Step 2. Define per VRF EIGRP routing context and parameters

In this step, configure the following:

- Per VRF EIGRP routing context for VRF CUST_A and CUST_B under the EIGRP routing process defined in Step 1. This number can be the same or different from the EIGRP AS number configured under the routing context.

- Configure the networks that need to be enabled for EIGRP using the **network** command.

- Ensure that **no auto-summary** is configured; otherwise, EIGRP summarizes networks at their classful boundaries. The command **no auto-summary** may be enabled by default depending on the version of IOS in use.

- To allow a single global EIGRP process to be used, the EIGRP AS has to be configured within the EIGRP address family configuration mode. This is accomplished by configuring **autonomous-system** *as-number* in address-family mode. This allows the same EIGRP AS number to be used in multiple VRFs.

Step 3. Configure the MP-BGP VPNv4 Backbone

Configuring BGP PE-PE routing between the PE routers is the next step in an MPLS VPN

deployment. The purpose of this step is to ensure that VPNv4 routes can be transported across the service provider backbone using MP-iBGP.

Step 4. Redistribute BGP VPNv4 routes in EIGRP

In this step, you redistribute the BGP VPNv4 routes from the remote PE routers in EIGRP.

Step 5. Redistribute EIGRP routes in BGP

In this step, the EIGRP routes received from the local CE router are redistributed in BGP on the PE router

We will explore this configuration in detail by deploying EIGRP as the CE-PE routing protocol according to Figure 8-1.

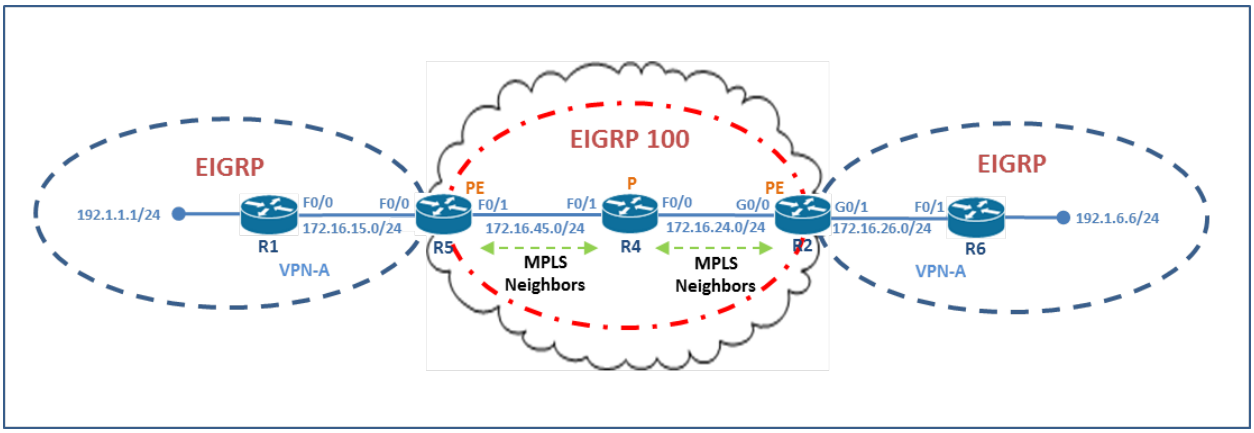


Figure 8-1: RIPv2 CE-PE Routing Topology

VRF Aware EIGRP (Same AS)

We will start this process first on R1 and R5, by creating the EIGRP VRF aware process on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router eigrp 1
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no auto
R5(config-router-af)#network 172.16.15.5 0.0.0.0
R5(config-router-af)#autonomous-system 101
R5(config-router-af)#end
```

We will go to R1 and create the EIGRP version process on that router as well. Please not that we are not using anything special in this process. We will not create an address-family on R1.

```
R1#conf t
R1(config)#router eigrp 101
R1(config-router)# network 172.16.15.1 0.0.0.0
```

```
R1(config-router)# network 192.1.1.1 0.0.0.0
R1(config-router)# no auto-summary
R1(config-router)# end
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 101: Neighbor 172.16.15.5 (FastEthernet0/0) is up: new
adjacency
```

Now if we did this correctly R5 will have a route to the network 192.1.1.0/24.

```
R5#sh ip route vrf VPN-A eigrp
D    192.1.1.0/24 [90/156160] via 172.16.15.1, 00:01:04, FastEthernet0/0
```

We see the route exists, but the final test is to successfully ping the address 192.1.1.1 from R5.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

We will repeat this process on R2 and R6.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#no router rip
R2(config)#router eigrp 1
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#no auto
R2(config-router-af)#network 172.16.26.2 0.0.0.0
R2(config-router-af)#autonomous-system 101
R2(config-router-af)#end
```

Now for the normal routing process on R6.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router eigrp 101
R6(config-router)#no auto
R6(config-router)#network 172.16.26.6 0.0.0.0
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 101: Neighbor 172.16.26.2 (FastEthernet0/1) is up: new
adjacency
R6(config-router)#network 192.1.6.6 0.0.0.0
R6(config-router)#end
```

Again we will look to see if R2 has learned the prefix for 192.1.6.0/24 via EIGRP.

```
R2#show ip route vrf VPN-A eigrp
D    192.1.6.0/24 [90/156160] via 172.16.26.6, 00:01:06, GigabitEthernet0/1
```

Now that we have created the individual EIGRP routing processes between devices it is going to be necessary to redistribute these EIGRP routes into the MP-BGP process under the **address-family ipv4 vrf VPN-A** context. This will be accomplished simply via the **redistribute eigrp 101** command.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute eigrp 101
R2(config-router-af)#end
```

The same process will need to be deployed on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#  address-family ipv4 vrf VPN-A
R5(config-router-af)#  redistribute eigrp 101
R5(config-router-af)#  end
```

We have just advertised the prefixes into the Layer 3 VPN. We can see that they are being communicated by using the **show ip bgp vpnv4** all command.

```
R5#show ip bgp vpnv4 all
BGP table version is 23, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*> 192.1.1.0        172.16.15.1         156160         32768 ?
*>i192.1.6.0        192.1.2.2           156160    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2           156160    100      0 ?
```

Now to repeat the process on R2.

```
R2#show ip bgp vpnv4 all
BGP table version is 23, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5           156160    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.0        192.1.5.5           156160    100      0 ?
*> 192.1.6.0        172.16.26.6         156160         32768 ?
```

We see that the PEs know about the routes originated on either side of the Layer 3 VPN. There is one element missing from our topology that will create a functional network. If we look at R1 and R6 we will see that these devices have no knowledge of the routes on either end of the network. For instance R1 has no knowledge of either of the routes 172.16.26.0/24 and 192.1.6.0/24.

```
R1#show ip route eigrp

R1#
```

We will repeat the process on R6 where we will no knowledge of the routes 192.1.1.0/24 and 172.16.15.0/24.

```
R6#sh ip route eigrp

R6#
```

Based on the fact that we have already discussed the need for mutual redistribution between EIGRP and MP-BGP, and the knowledge that we have not redistributed the MP-BGP routes into EIGRP on either R5 or R2 we should not expect the CE to have any routes to each other. This is what we are seeing now. In the next section we will address this shortfall in our topology.

Redistribution of MP-BGP into VRF Aware EIGRP

We will now go to R5 and redistribute the contents of the MP-BGP routing table into the VRF aware instance of EIGRP. First we will look at the contents of the MP-BGP routing process on R5. The goal is to identify the exact prefixes that will be redistributed into the **address-family ipv4 vrf VPN-A** process.

```
R5#show ip bgp vpnv4 rd 100:1 neighbors 192.1.2.2 routes
BGP table version is 23, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.0        192.1.2.2           156160    100      0 ?

Total number of prefixes 2
```

Redistribution with Seed Metric

We will redistribute MP-BGP into EIGRP on R5, the net result will be the appearance of 192.1.6.0/24 and 172.16.26.0/24 in R1's routing table for the EIGRP process. We will specify a seed metric of 1 1 1 1 1 when we do this.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router eigrp 1
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute bgp 65000 metric 1 1 1 1 1
R5(config-router-af)#end
```

We will no look at R1 to see if the routes have been injected into the routing table for RIPv2.

```
R1#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.26.0 [90/30720] via 172.16.15.5, 00:00:28, FastEthernet0/0
D    192.1.6.0/24 [90/158720] via 172.16.15.5, 00:00:28, FastEthernet0/0
```

We will now repeat this process on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#  router eigrp 1
R2(config-router)#  address-family ipv4 vrf VPN-A
R2(config-router-af)#  redistribute bgp 65000 metric 1 1 1 1 1
R2(config-router-af)#  end
```

We now will look at R6 to see if the EIGRP routes are being learned.

```
R6#sh ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.15.0 [90/30720] via 172.16.26.2, 00:00:50, FastEthernet0/1
D    192.1.1.0/24 [90/158720] via 172.16.26.2, 00:00:50, FastEthernet0/1
```

The prefixes are being learned and we have reachability as demonstrated by the ping utility.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

VRF Aware EIGRP (Different AS)

We have just created a working configuration with EIGRP as the PE-CE routing protocol. But there are a few things we need to point out at this juncture of our discussion. Note that we use Autonomous System 101 on both sides of the topology. What would happen if we used different AS numbers?

Before we answer that question we will look at our routing table on R1 again.

```
R1#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.26.0 [90/30720] via 172.16.15.5, 00:00:28, FastEthernet0/0
D    192.1.6.0/24 [90/158720] via 172.16.15.5, 00:00:28, FastEthernet0/0
```

Observe that the routes that are exchanged are showing up as internal EIGRP routes. Keep this in mind as we make changes between the routing protocols on R1 and R5. For the purpose of demonstration we will change the AS to 100 on these devices and see what happens.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
R5(config)#router eigrp 1
R5(config-router)#no address-family ipv4 vrf VPN-A
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no auto
R5(config-router-af)#network 172.16.15.5 0.0.0.0
R5(config-router-af)#redistribute bgp 65000 metric 1 1 1 1 1
R5(config-router-af)#autonomous-system 100
R5(config-router-af)#end
```

Now for R1.

```
R1#conf t
R1(config)#no router eigrp 101
R1(config)#router eigrp 100
R1(config-router)#no auto
R1(config-router)#network 192.1.1.1 0.0.0.0
R1(config-router)#network 172.16.15.1 0.0.0.0
R1(config-router)#end
R1#
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 100: Neighbor 172.16.15.5 (FastEthernet0/0) is up: new
adjacency
```

We have an adjacency between R1 and R5 now. With this accomplished we need to look at the routing table on R1 and see what if anything has changed.

```
R1#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D EX    172.16.26.0
           [170/2560002816] via 172.16.15.5, 00:00:07, FastEthernet0/0
D EX 192.1.6.0/24 [170/2560002816] via 172.16.15.5, 00:00:07, FastEthernet0/0
```

Observe that these prefixes are now listed as external routes. Keep in mind the operational parameters of EIGRP. Routes learned from a different AS will be listed as external routes.

EIGRP PE-CE Common Issues

The primary issues that affect PE-CE EIGRP routing configurations fall into just a handful of operational domains. As a general rule, and to keep the failure domains associated to the individual topic of this chapter we will focus on the aspects of EIGRP VRF aware routing as they apply to the PE – CE communication process.

The main issues that PE-CE EIGRP Routing brings us fall into three categories: EIGRP Configuration Errors, Redistribution Issues and Filtering Issues. We will explore each of these categories in the sample topology illustrated in Figure 8-2.

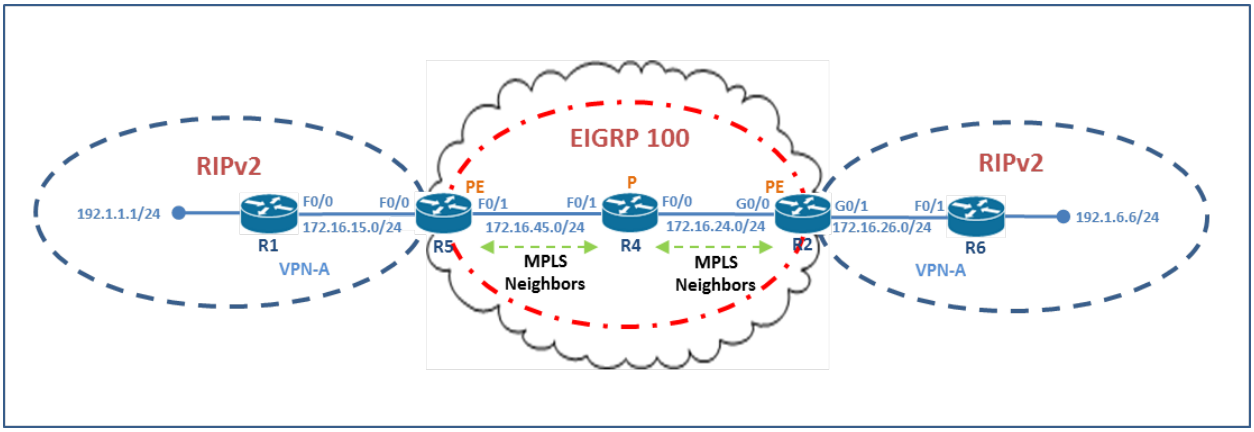


Figure 8-2: EIGRP PE-CE Topology

We will explore each failure category in turn. Remember that for the purposes of our discussions we will consider the service provider cloud to be operational.

EIGRP Configuration Errors

There are actually two areas where EIGRP configuration issues can impact the performance of our MPLS domain. Primarily these two locations are at the CE and at the PE. We will look at the simplest of these locations first.

CE EIGRP

At the CE we will use the standard EIGRP routing protocol that we have used throughout our studies. Due to the fact that the CE router has no knowledge of the concept of the virtual routing and forwarding instance we have no level of complication introduced at this phase. The typical issues associated with EIGRP is to forget turn off auto-summary, to leave out the network commands or to specify the wrong inverse mask.

PE EIGRP

VRF aware routing is required at the PE in order to get EIGRP to work as the routing protocol between the PE and CE devices while simultaneously supporting the Layer 3 VPN essential to our service provider cloud. In this case the most common problem is to forget to configure the VRF instance with the **address-family ipv4 vrf VPN-A** or to misconfigure the network command.

We can explore what happens when this happens, by removing the command from R5 and observing the results on R1 and R6.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router eigrp 1
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no network 172.16.15.5 0.0.0.0
R5(config-router-af)#
%DUAL-5-NBRCHANGE: IP-EIGRP(1) 101: Neighbor 172.16.15.1 (FastEthernet0/0) is down:
interface down
R5(config-router-af)#end
```

In this example we are removed the network statement. This will result in a loss of prefix exchange between the PE and CE as demonstrated by the NBRCHANGE message. We can also see this via the **show ip route eigrp** command on R1.

```
R1#show ip route eigrp

R1#
```

We are not learning any routes here, because they are not being originated into the Layer 3 VPN by the redistribution process. We will restore the correct network commands on R5 and see that the topology comes back up.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router eigrp 1
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#network 172.16.15.5 0.0.0.0
R5(config-router-af)#end
R5#
%DUAL-5-NBRCHANGE: IP-EIGRP(1) 101: Neighbor 172.16.15.1 (FastEthernet0/0) is up: new
adjacency
```

Once this is accomplished our routes will return to both R1 and R6.

```
R1#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.26.0 [90/30720] via 172.16.15.5, 00:00:32, FastEthernet0/0
D    192.1.6.0/24 [90/158720] via 172.16.15.5, 00:00:32, FastEthernet0/0
```

On R6 we see the same.

```
R6#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.15.0 [90/30720] via 172.16.26.2, 00:01:04, FastEthernet0/1
D    192.1.1.0/24 [90/158720] via 172.16.26.2, 00:00:49, FastEthernet0/1
```

We also have restored reachability as a result.

```
R6#ping 192.1.1.1 source lo 0
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Redistribution Issues

As we have mentioned earlier it is a necessity to use redistribution in our topology for all PE-CE routing protocols other than eBGP peering. So this means that we need an understanding of the issues related to the incompatibility of metrics used by our individual protocols. This extends to both our MP-BGP and EIGRP protocols. We will explore the nature of these issues as they relate to EIGRP in our topology by altering the configuration on our PEs. These aspects as the relate to redistribution will impact both our IGP Redistribution and MP-BGP Redistribution.

IGP Redistribution

We will explore the issue related to redistribution and our VRF aware IGPs by looking at some of the possible problems related to the configuration process. As an example we will alter the configuration on R5 under the router rip configuration such that we will not specify a seed metric.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router eigrp 1
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no redistribute bgp 65000
R5(config-router-af)#redistribute bgp 65000
R5(config-router-af)#end
```

When we go to R1 to look at the results we see that there are no routes being injected.

```
R1#show ip route eigrp

R1#
```

MP-BGP Redistribution

Regarding redistribution of the IGP prefixes into the MP-BGP protocol the most common issues that can be encountered fall into one of two categories.

Lack of Redistribution – When redistribution is not used under the MP-BGP process, we will never see any prefixes advertised to the remote PE devices. We can demonstrate this by removing the redistribution command and using the show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#no redistribute eigrp 101
R5(config-router-af)#end
```

We can clearly see that we are not advertising prefixes any longer.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes

Total number of prefixes 0
```

Use of an Artificially High Metric – When the redistribution process is used and a maximum value seed metric is assigned it is impossible for the route to be redistributed into the VRF aware RIPv2 process. We can see this process by modifying the metric to the maximum value.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute eigrp 101 metric ?
  <0-4294967295>  Default metric

R5(config-router-af)#redistribute eigrp metric 4294967295
R5(config-router-af)#end
```

We can see that the prefix is advertised to the remote PE with the newly assigned max metric.

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 15, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1     4294967295         32768 ?

Total number of prefixes 2
```

We will go to R2 and look at the routes learned from R5 and see if the prefix is found in the table as being learned from R5.

```
R2#show ip bgp vpnv4 all neighbors 192.1.5.5 routes
BGP table version is 15, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1     4294967295         32768 ?

Total number of prefixes 2
```

The prefix arrives on R2 but is it sent to R6 as an EIGRP route?

```
R6#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.26.0 is directly connected, FastEthernet0/1
D       172.16.15.0 [90/30720] via 172.16.26.2, 00:11:41, FastEthernet0/1
C    192.1.6.0/24 is directly connected, Loopback0
```

Observe that there is no record of the prefix 192.1.1.0/24 only the connected prefix 172.16.15.0/24. This is because the prefix was not valid at the time of redistribution because of the maximum metric used when it was redistributed.

Filtering Issues

As with any routing protocol, filtering whether it is in the form of access-lists, offset-lists, route-maps and distribute-lists can block prefixes from being allowed to enter the routing table of a device. VRF aware routing protocols like EIGRP are no exception. These tools are can very easily be used to induce issues associated with our CE-PE routing.

Chapter Challenge: EIGRP Sample Trouble Ticket

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH8-EIGRP-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 8-3 below:

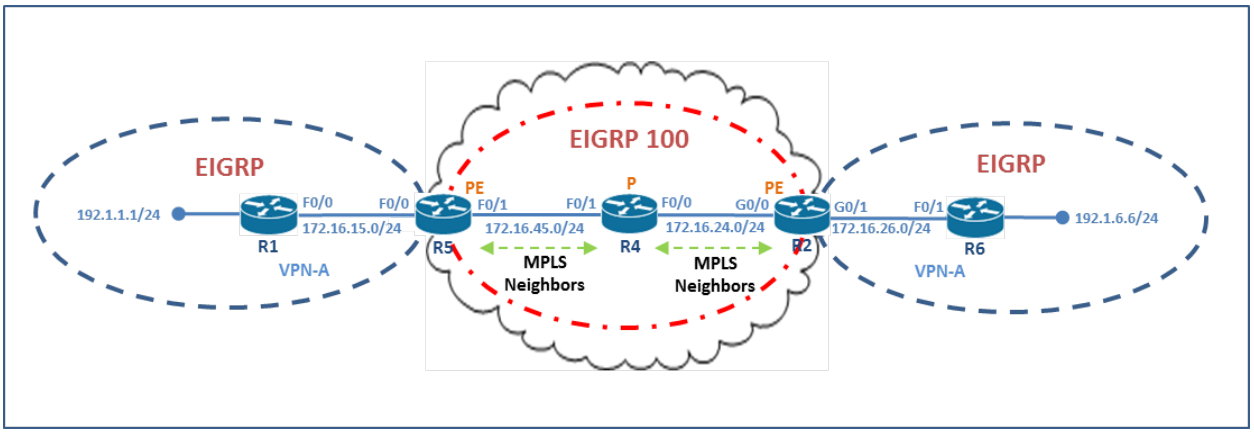


Figure 8-3: The Chapter Challenge Topology

Trouble Ticket #1

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Chapter Challenge: EIGRP Sample Trouble Ticket Solution

The following section includes the solutions to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Does the ping to 192.1.1.1 work or fail on R6?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

The pings are unsuccessful.

Step 2 - Fault Isolation:

First we will make sure that R5 can reach 192.1.1.1.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

R5 can reach the address in question. Now we need to see if R5 is advertising this prefix to the remote PE (R5).

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes

Total number of prefixes 0
```

We know that the VRF aware EIGRP process is injected into the VPNv4 tunnel via redistribution. The best and simplest way to verify this process is with the **show run | sec router bgp** command.

```
R5#show run | sec router bgp
router bgp 65000
 bgp log-neighbor-changes
 neighbor 192.1.2.2 remote-as 65000
 neighbor 192.1.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 192.1.2.2 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 192.1.2.2 activate
  neighbor 192.1.2.2 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  no synchronization
 exit-address-family
```

We see that there is no redistribution taking place under the **address-family ipv4 vrf VPN-A** context. We will correct this issue.

Step 3 - Fault Remediation:

In this scenario, we will add the **redistribute eigrp 101** command under the address-family.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)# address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute eigrp 101
R5(config-router-af)#end
```

Now we will recheck to ensure that the prefixes are actually being advertised to the remote PE.

```
R5#show ip bgp vpnv4 all neighbor 192.1.2.2 advertised-routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*> 192.1.1.0        172.16.15.1              1         32768 ?

Total number of prefixes 2
```

We see that they are being advertise, but is R2 accepting these prefixes?

```
R2#show ip bgp vpnv4 rd 100:1 neighbor 192.1.5.5 routes
```

```
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.0        192.1.5.5           156160    100      0 ?
```

The prefixes are being accepted. Are they being redistributed into the EIGRP vrf VPN-A instance?

```
R6#show ip route eigrp

R6#
```

The prefixes in question are not appearing on R6. Now we will need to see if they are being redistributed into the EIGRP address-family vrf VPN-A. We can see this via the **show run | sec router eigrp 1_** command.

```
R2#show run | sec router eigrp 1_
router eigrp 1
 no auto-summary
 !
 address-family ipv4 vrf VPN-A
  redistribute bgp 65000
  network 172.16.0.0
  no auto-summary
  autonomous-system 101
 exit-address-family
```

As we can see we are redistributing the BGP process, but we are not specifying a metric. To satisfy the need for a seed metric we will use 1 1 1 1 1.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router eigrp 1
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute bgp 65000 metric 1 1 1 1 1
R2(config-router-af)#end
```

Now we will look again at R6.

```
R6#show ip route eigrp
     172.16.0.0/24 is subnetted, 2 subnets
D       172.16.15.0 [90/30720] via 172.16.26.2, 00:00:57, FastEthernet0/1
D    192.1.1.0/24 [90/158720] via 172.16.26.2, 00:00:57, FastEthernet0/1
```

Based on this we should have full reachability.

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R6#
```

The issues have been corrected.

Chapter 9: OSPF

OSPF is preferred as the PE-CE routing protocol of choice in most network environments. In this section we are going to explore issues associated with implementing traditional OSPF routing models in an MPLS VPN environment, and the concept of the OSPF "super backbone" and how it resolves them.

Introduction to OSPF PE-CE Routing

OSPF is preferred as the PE-CE routing protocol of choice in most network environments. In this section we are going to explore issues associated with implementing traditional OSPF routing models in an MPLS VPN environment, and the concept of the OSPF "super backbone" and how it resolves them. Additionally, we will work with OSPF sham-links to correct suboptimal routing that can be caused by backdoor links between OSPF sites that are attached via an MPLS VPN environment.

In each of the following scenarios we will deploy MPLS and the PE-CE protocols using the following step-by-step process:

Step 1. **Enable per VRF OSPF Routing**

Enable per VRF OSPF routing for each VRF instance in use in the MPLS domain.

Step 2. **Create the MP-BGP VPNv4 Backbone**

Configuring BGP PE-PE routing between the PE routers is the next step in an MPLS VPN deployment. The purpose of this step is to ensure that VPNv4 routes can be transported across the service provider backbone using MP-iBGP. The P routers are transparent to this entire process and, therefore, does not carry any customer routes. This is referred to as a "BGP Free Core".

Step 2. **Redistribute OSPF Routes into BGP**

All OSPF routes received from the local CE routers are redistributed in MP-iBGP. It is necessary to include the **match** command option; otherwise, only OSPF internal routes will be redistributed into BGP.

Step 3. **Redistribute MP-IBGP into OSPF**

Next we will redistribute the BGP VPNv4 routes into OSPF on the PE routers. For this process to work we must use the **subnets** keyword when configuring redistribution; otherwise, Cisco IOS redistributes only the major classful networks.

We will explore this configuration in detail by deploying OSPF as the CE-PE routing protocol according to Figure 9-1.

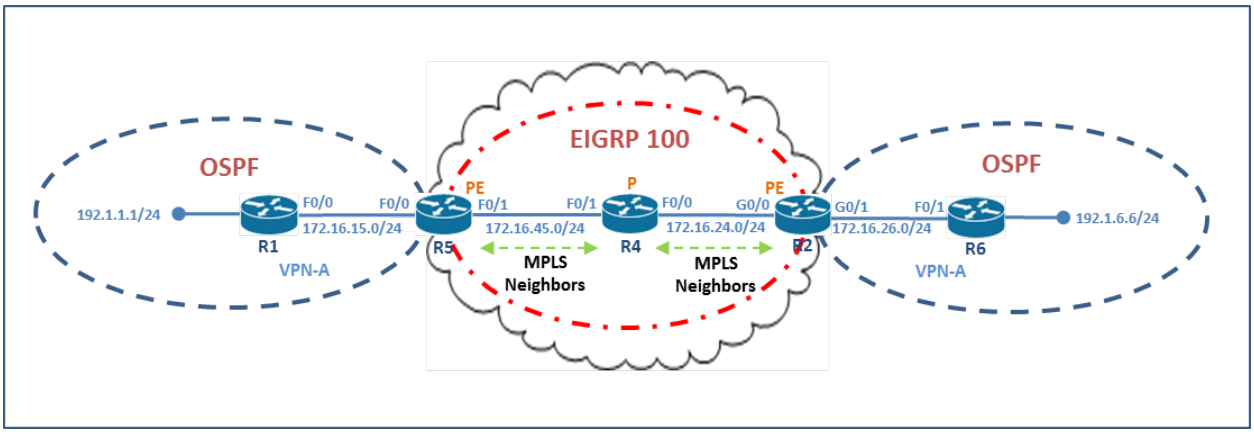


Figure 9-1:OSPF CE-PE Routing Topology

VRF Aware OSPF (Same Process ID)

We will start this process first on R1 and R5, by creating the OSPF VRF aware process on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 1 vrf VPN-A
R5(config-router)#router-id 192.1.5.5
R5(config-router)#network 172.16.15.5 0.0.0.0 area 0
R5(config-router)#exit
```

We will go to R1 and create the OSPF process on that router as well. Please note that we are not using anything special in this process. We will not create an address-family on R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#router-id 192.1.1.1
R1(config-router)#network 192.1.1.1 0.0.0.0 area 0
R1(config-router)#network 172.16.15.1 0.0.0.0 area 0
R1(config-router)#end
R1#
%OSPF-5-ADJCHG: Process 1, Nbr 192.1.5.5 on FastEthernet0/0 from LOADING to FULL,
Loading Done
%SYS-5-CONFIG_I: Configured from console by console
```

Now if we did this correctly R5 will have a route to the network 192.1.1.0/24.

```
R5#show ip route vrf VPN-A ospf

Routing Table: VPN-A

     192.1.1.0/32 is subnetted, 1 subnets
O       192.1.1.1 [110/2] via 172.16.15.1, 00:01:03, FastEthernet0/0
```

We see the route exists, but the final test is to successfully ping the address 192.1.1.1 from R5.

```
R5#ping vrf VPN-A 192.1.1.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

We will repeat this process on R2 and R6.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router ospf 1 vrf VPN-A
R2(config-router)#router-id 192.1.2.2
R2(config-router)#network 172.16.26.2 0.0.0.0 area 0
R2(config-router)#end
```

Now for the normal routing process on R6.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router ospf 1
R6(config-router)#router-id 192.1.6.6
R6(config-router)#network 192.1.6.6 area 0
                                    ^
% Invalid input detected at '^' marker.

R6(config-router)#network 192.1.6.6 0.0.0.0 area 0
R6(config-router)#network 172.16.26.6 0.0.0.0 area 0
R6(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R6#
%OSPF-5-ADJCHG: Process 1, Nbr 192.1.2.2 on FastEthernet0/1 from LOADING to FULL,
Loading Done
```

Again we will look to see if R2 has learned the prefix for 192.1.6.0/24 via OSPF.

```
R2#show ip route vrf VPN-A ospf

Routing Table: VPN-A

     192.1.6.0/32 is subnetted, 1 subnets
O       192.1.6.6 [110/2] via 172.16.26.6, 00:01:20, GigabitEthernet0/1
```

Now that we have created the individual OSPF routing processes between devices it is going to be necessary to redistribute these OSPF routes into the MP-BGP process under the **address-family ipv4 vrf VPN-A** context. This will be accomplished simply via the **redistribute ospf 1** command.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute ospf 1
R2(config-router-af)#end
```

The same process will need to be deployed on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute ospf 1
R5(config-router-af)#end
```

We have just advertised the prefixes into the Layer 3 VPN. We can see that they are being communicated by using the **show ip bgp vpnv4 all** command.

```
R5#show ip bgp vpnv4 all
BGP table version is 23, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24   0.0.0.0                  0         32768 ?
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*> 192.1.1.1/32     172.16.15.1              2         32768 ?
*>i192.1.6.6/32     192.1.2.2                2    100      0 ?
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.6/32     192.1.2.2                2    100      0 ?
```

Now to repeat the process on R2.

```
R2#show ip bgp vpnv4 all
BGP table version is 23, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.1/32     192.1.5.5                2    100      0 ?
Route Distinguisher: 100:2 (default for vrf VPN-A)
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*>i192.1.1.1/32     192.1.5.5                2    100      0 ?
*> 192.1.6.6/32     172.16.26.6              2         32768 ?
```

We see that the PEs know about the routes originated on either side of the Layer 3 VPN. There is one element missing from our topology that will create a functional network. If we look at R1 and R6 we will see that these devices have no knowledge of the routes on either end of the network. For instance R1 has no knowledge of either of the routes 172.16.26.0/24 and 192.1.6.0/24.

```
R1#show ip route ospf

R1#
```

We will repeat the process on R6 where we will no knowledge of the routes 192.1.1.0/24 and 172.16.15.0/24.

```
R6#sh ip route ospf

R6#
```

Based on the fact that we have already discussed the need for mutual redistribution between OSPF and MP-BGP, and the knowledge that we have not redistributed the MP-BGP routes into OSPF on either R5 or R2 we should not expect the CEs to have any routes to each other. This is what we are seeing now. In the next section we will address this shortfall in our topology.

Redistribution of MP-BGP into VRF Aware OSPF

We will now go to R5 and redistribute the contents of the MP-BGP routing table into the VRF aware instance of OSPF. First we will look at the contents of the MP-BGP routing process on R5. The goal is to identify the exact prefixes that will be redistributed into the **address-family ipv4 vrf VPN-A** process.

```
R5#show ip bgp vpnv4 rd 100:1 neighbors 192.1.2.2 routes
BGP table version is 23, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.6/32     192.1.2.2                2    100      0 ?

Total number of prefixes 2
```

Redistribution with Subnets

We will redistribute MP-BGP into OSPF on R5, the net result will be the appearance of 192.1.6.0/24 and 172.16.26.0/24 in R1's routing table for the OSPF process. We will not specify a seed metric but we will indeed what to insure that we preserve the individual subnets we are learning.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 1 vrf VPN-A
R5(config-router)#redistribute bgp 65000 subnets
R5(config-router)#end
```

We will now look at R1 to see if the routes have been injected into the routing table for OSPF.

```
R1#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.26.0 [110/2] via 172.16.15.5, 00:01:10, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O IA    192.1.6.6 [110/3] via 172.16.15.5, 00:01:10, FastEthernet0/0
```

We will now repeat this process on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router ospf 1 vrf VPN-A
R2(config-router)#redistribute bgp 65000 subnets
R2(config-router)#end
```

We now will look at R6 to see if the EIGRP routes are being learned.

```
R6#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.15.0 [110/2] via 172.16.26.2, 00:00:55, FastEthernet0/1
     192.1.1.0/32 is subnetted, 1 subnets
O IA    192.1.1.1 [110/3] via 172.16.26.2, 00:00:55, FastEthernet0/1
```

The prefixes are being learned and we have reachability as demonstrated by the ping utility.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

VRF Aware OSPF (Different Process ID)

We have just created a working configuration with OSPF as the PE-CE routing protocol. But there are a few things we need to point out at this juncture of our discussion. Note that we used OSPF Process 1 on both sides of the topology. What would happen if we used a different Process IDs?

Before we answer that question we will look at our routing table on R1 again.

```
R6#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.15.0 [110/2] via 172.16.26.2, 00:00:55, FastEthernet0/1
     192.1.1.0/32 is subnetted, 1 subnets
O IA    192.1.1.1 [110/3] via 172.16.26.2, 00:00:55, FastEthernet0/1
```

Observe that the routes that are exchanged are showing up as intra area (IA) OSPF routes. Keep this in mind as we make changes between the routing protocols on R1 and R5. For the purpose of demonstration we will change the Process ID to 100 on these devices and see what happens.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#no router ospf 1 vrf VPN-A
%OSPF-5-ADJCHG: Process 1, Nbr 192.1.1.1 on FastEthernet0/0 from FULL to DOWN,
Neighbor Down: Interface down or detached
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#router-id 192.1.5.5
R5(config-router)#network 172.16.15.5 0.0.0.0 area 0
R5(config-router)#redistribute bgp 65000 subnets
R5(config-router)#end
```

Because we removed the entire OSPF routing process we will need to redo the redistribution command under the BGP process.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute ospf 100
R5(config-router-af)#end
```

Now for R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#no router ospf 1
R1(config)#router ospf 100
R1(config-router)#router-id 192.1.1.1
R1(config-router)#network 172.16.15.1 0.0.0.0 area 0
%OSPF-5-ADJCHG: Process 100, Nbr 192.1.5.5 on FastEthernet0/0 from LOADING to FULL,
Loading Done
R1(config-router)#network 192.1.1.1 0.0.0.0 area 0
R1(config-router)#end
```

We have an adjacency between R1 and R5 now. With this accomplished we need to look at the routing table on R1 and see what if anything has changed.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
O E2    172.16.26.0 [110/1] via 172.16.15.5, 00:05:03, FastEthernet0/0
C       172.16.15.0 is directly connected, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O E2    192.1.6.6 [110/2] via 172.16.15.5, 00:05:03, FastEthernet0/0
C    192.1.1.0/24 is directly connected, Loopback0
```

Observe that these prefixes are now listed as external type 2 routes.

OSPF Domain-ID

As we have demonstrated in the previous section when we use OSPF as the routing protocol on a MPLS Provider Edge to customer edge (PE-CE) links. The PE will mark all OSPF routes with the domain attribute derived from the OSPF process number, this tells the routers whether the route originated within the same OSPF domain or from another OSPF domain. If the OSPF process numbering is inconsistent on the

PE routers in the MPLS VPN, the **domain-id** command can be used to remark the OSPF prefixes with the same numbers such that the routes will think the prefixes are part of the same OSPF domain.

This means that we can get O IA routes instead of E2 routes by using matching OPSF domain-id on each end of the Layer 3 VPN tunnel. We will demonstrate this by adding the **domain-id 1.1.1.1** command to both R5 and R2.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#domain-id 1.1.1.1
R5(config-router)#end
```

Now for R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router ospf 1 vrf VPN-A
R2(config-router)#domain-id 1.1.1.1
R2(config-router)#end
```

With this accomplished the Layer 3 VPN will exchange the newly defined OSPF DOMAIN-ID attribute. We can see this by looking at the specific route. As an example we will focus on 192.1.6.6/32.

```
R5#show ip bgp vpnv4 rd 100:1 192.1.6.6
BGP routing table entry for 100:1:192.1.6.6/32, version 50
Paths: (1 available, best #1, table VPN-A)
Flag: 0xA00
  Not advertised to any peer
  Local, imported path from 100:2:192.1.6.6/32
    192.1.2.2 (metric 158720) from 192.1.2.2 (192.1.2.2)
      Origin incomplete, metric 2, localpref 100, valid, internal, best
      Extended Community: RT:100:1 OSPF DOMAIN ID:0x0005:0x010101010200
        OSPF RT:0.0.0.0:2:0 OSPF ROUTER ID:192.1.2.2:0
      mpls labels in/out nolabel/204
```

Note the domain value is 01010101 which is the hexadecimal version 1.1.1.1. Because the DOMAIN values match we are telling the OSPF vrf VPN-A process that these routes are part of the same domain. We can see that these prefixes will be redistributed as O IA routes on R1 and R6.

```
R1#sh ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.26.0 [110/2] via 172.16.15.5, 00:05:15, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O IA    192.1.6.6 [110/3] via 172.16.15.5, 00:05:15, FastEthernet0/0
```

The route type has changed from external to inter-area.

OPSF Down-Bit

We need to take this opportunity to discuss another MP-BGP attribute. This attribute is known as the down-bit. In our topology R5 and R1 are redistributing our OSPF routes into the MP-BGP process. When this takes place the PEs will copy the OSPF cost value into the Multi Exit Discriminator (MED) attribute while they simultaneously setting the MP-BGP extended community in such a fashion as to tell us the LSA type that was used to derive the route.

The remote PE will then redistribute this information back into OSPF. This means that the original LSA type and the OSPF cost value that has been recorded in the MED attribute will be used to generate an inter-area summary LSA. This Type 3 LSA will always be generated due to the fact that the PE router is acting as an ABR between the elements of the super backbone and the OSPF areas.

When VRF aware OSPF is used as the routing protocol between PE and CE devices the PE must redistribute information into that VPN's OSPF instance. These prefixes are those that have been installed in the MP-BGP routing table. Similarly, the PE router must redistribute the routes that have been installed in the VRF-specific OSPF routing tables into MP-BGP.

When the PE receives type 3, 5, or 7 LSA with the down-bit set from a remote CE, the information from that LSA must not be used during the OSPF calculations for that route. As a result, the LSA is not translated into a BGP route.

The down bit is used between the PE routers to indicate which routes were inserted into the OSPF topology database from the MPLS VPN super backbone and thus are not to be redistributed back in the MPLS VPN super backbone. The PE router that redistributes the MP-BGP route as an OSPF route into the OSPF topology database is the device that sets the down bit. The other PE routers will use the down bit to prevent this route from being redistributed back into MP-BGP thus prevent the formation or routing loops.

This prevents routes learned via MP-BGP from being redistributed back into MP-BGP. This restriction is identical to the typical OSPF loop prevention mechanism that states that inter-area routes learned from area 0 are never passed back to area 0.

In our topology we can see the status of this down-bit in the output of the **show ip ospf database summary** command.

```
R5#show ip ospf database summary

            OSPF Router with ID (192.1.5.5) (Process ID 100)

                Summary Net Link States (Area 0)

  LS age: 307
  Options: (No TOS-capability, DC, Downward)
```



```
  LS Type: Summary Links(Network)
  Link State ID: 172.16.26.0 (summary Network Number)
  Advertising Router: 192.1.5.5
  LS Seq Number: 80000002
  Checksum: 0x55C2
  Length: 28
  Network Mask: /24
        TOS: 0  Metric: 1

  LS age: 307
  Options: (No TOS-capability, DC, Downward)
  LS Type: Summary Links(Network)
  Link State ID: 192.1.6.6 (summary Network Number)
  Advertising Router: 192.1.5.5
  LS Seq Number: 80000002
  Checksum: 0xAF70
  Length: 28
  Network Mask: /32
        TOS: 0  Metric: 2
```

We must note that we can eliminate these operation steps at the PE by using the **capability vrf-lite** command.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#capability vrf-lite
R5(config-router)#end
```

The use of this command will in effect remove all information pertinent to the prefixes learned via the MPLS Layer 3 VPN. This will force R1 to see these prefixes as external type 2 routes. But before we look at R1 we will look at the contents of the database. By using the capability vrf-lite command we should not be generating the Type 3 LSA we were previously.

```
R5#show ip ospf database summary

            OSPF Router with ID (192.1.5.5) (Process ID 100)

R5#
```

As expect this LSA has been removed from our database. Additionally, the routes should be E2 in the ospf table on R1.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set
```

```
     172.16.0.0/24 is subnetted, 2 subnets
O E2    172.16.26.0 [110/1] via 172.16.15.5, 00:06:22, FastEthernet0/0
C       172.16.15.0 is directly connected, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O E2    192.1.6.6 [110/1] via 172.16.15.5, 00:06:22, FastEthernet0/0
C    192.1.1.0/24 is directly connected, Loopback0
```

We will remove the capability vrf-lite command from R5 before we move any further.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#no capability vrf-lite
R5(config-router)#end
```

We will see that the prefixes return to O IA on R1 as a result of this action.

```
R1#sh ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.26.0 [110/2] via 172.16.15.5, 00:00:26, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O IA    192.1.6.6 [110/3] via 172.16.15.5, 00:00:26, FastEthernet0/0
```

OSPF Sham-link

We can see that in the previous section our routes are being learned as O IA. This may create issues for us in environments where we have OPSF “backdoor” links configured. The issue specifically, is the fact that OSPF will select “O” routes before it selects “O IA” routes. So we need a feature that will allows us the transition our O IA’s to O routes. That is the feature that OSPF Sham-links bring to the table.

An OSPF sham-link allows us to create an OSPF Point-to-Point adjacency over the MPLS Layer 3 VPN. This allows us to extend the area 0 network through the provider cloud. In a nutshell when we create an OSPF Sham-link we are going to assign a source and destination for the link. The source will be an interface on the local PE and the destination will be an interface on the remote PE. We will implement an OSPF sham-link on R5 and R2 in our topology and observe the result on R1 and R6 (the CE’s). We will start on R5. We will create two new interfaces to accomplish this task on these devices and place them in the VPN-A vrf. In addition to this we will advertise these interface into MP-BGP and then lastly create the sham-link itself.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#int lo 100
R5(config-if)#ip vrf forwarding VPN-A
R5(config-if)#ip address 25.25.25.5 255.255.255.255
R5(config-if)#exit
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#network 25.25.25.5 mask 255.255.255.255
R5(config-router-af)#exit
R5(config-router)#exit
```

```
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#area 0 sham-link 25.25.25.5 25.25.25.2 cost 1
R5(config-router)#end
```

Now to repeat the process on R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#int lo 100
R2(config-if)#ip vrf forwarding VPN-A
R2(config-if)#ip address 25.25.25.2 255.255.255.255
R2(config-if)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#network 25.25.25.2 mask 255.255.255.255
R2(config-router-af)#exit
R2(config-router)#exit
R2(config)#router ospf 1 vrf VPN-A
R2(config-router)#area 0 sham-link 25.25.25.2 25.25.25.5 cost 1
R2(config-router)#end
%OSPF-5-ADJCHG: Process 1, Nbr 192.1.5.5 on OSPF_SL0 from LOADING to FULL, Loading
Done
```

We can see the status of our newly created sham-link via the **show ip ospf sham-link** command.

```
R2#show ip ospf sham-links
Sham Link OSPF_SL0 to address 25.25.25.5 is up
Area 0 source address 25.25.25.2
  Run as demand circuit
  DoNotAge LSA allowed. Cost of using 1 State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40,
    Hello due in 00:00:01
    Adjacency State FULL (Hello suppressed)
    Index 2/2, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Now based on our expectation we should see the prefixes on R1 and R6 are now O type routes for 192.1.1.0/24, 192.1.6.0/24, 172.16.15.0/24 and 172.16.26.0/24 respectively. Also we should expect to see external type 2 prefixes for the host routes 25.25.25.5/32 and 25.25.25.2/32. First we will look at R1.

```
R1#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O       172.16.26.0 [110/3] via 172.16.15.5, 00:00:30, FastEthernet0/0
     25.0.0.0/32 is subnetted, 2 subnets
O E2    25.25.25.2 [110/1] via 172.16.15.5, 00:00:55, FastEthernet0/0
O E2    25.25.25.5 [110/1] via 172.16.15.5, 00:13:14, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O       192.1.6.6 [110/4] via 172.16.15.5, 00:00:30, FastEthernet0/0
```

Now we will repeat the test on R6.

```
R6#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
```

```
O        172.16.15.0 [110/3] via 172.16.26.2, 00:01:49, FastEthernet0/1
     25.0.0.0/32 is subnetted, 2 subnets
O E2     25.25.25.2 [110/1] via 172.16.26.2, 00:09:23, FastEthernet0/1
O E2     25.25.25.5 [110/1] via 172.16.26.2, 00:02:04, FastEthernet0/1
     192.1.1.0/32 is subnetted, 1 subnets
O        192.1.1.1 [110/4] via 172.16.26.2, 00:01:49, FastEthernet0/1
```

Everything is as we have anticipated. Lastly we will verify reachability on our topology from R6.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

OSPF PE-CE Common Issues

The primary issues that affect PE-CE OSPF routing configurations fall into just a handful of operational domains. As a general rule, and to keep the failure domains associated to the individual topic of this chapter we will focus on the aspects of OSPF VRF aware routing as they apply to the PE – CE communication process.

The main issues that PE-CE OSPF Routing brings us fall into three categories: OSPF Configuration Errors, Redistribution Issues and Filtering Issues. We will explore each of these categories in the sample topology illustrated in Figure 9-2.

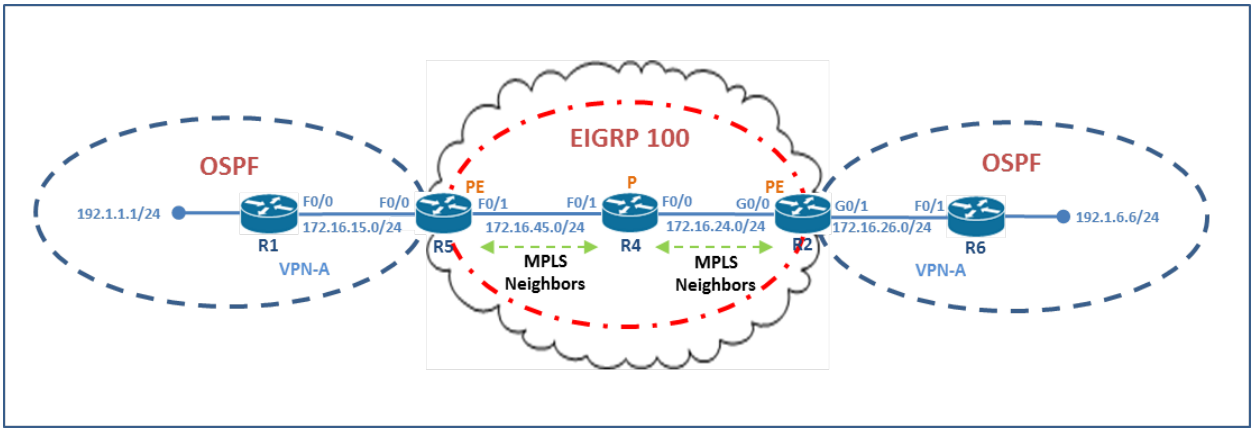


Figure 9-2: OSPF PE-CE Topology

We will explore each failure category in turn. Remember that for the purposes of our discussions we will consider the service provider cloud to be operational.

OSPF Configuration Errors

There are actually two areas where OSPF configuration issues can impact the performance of our MPLS domain. Primarily these two locations are at the CE and at the PE. We will look at the simplest of these locations first.

CE OSPF

At the CE we will use the standard OSPF routing protocol that we have used throughout our studies. Due to the fact that the CE router has no knowledge of the concept of the virtual routing and forwarding instance we have no level of complication introduced at this phase. The typical issues associated with EIGRP is to forget turn off auto-summary, to leave out the network commands or to specify the wrong inverse mask.

PE OSPF

VRF aware routing is required at the PE in order to get OSPF to work as the routing protocol between the PE and CE devices while simultaneously supporting the Layer 3 VPN essential to our service provider cloud. In this case the most common problem is to forget to configure the VRF instance with the **router ospf # vrf VPN-A** or to misconfigure the network command.

We can explore what happens when this happens, by removing the **network** command from R5 and observing the results on R1 and R6.

```
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#no network 172.16.15.5 0.0.0.0 area 0
R5(config-router)#end
```

In this example we removed the network statement. This will result in a loss of prefix exchange between the PE and CE. We can also see this via the **show ip route ospf** command on R1.

```
R1#show ip route ospf

R1#
```

We are not learning any routes here, because they are not being originated into the Layer 3 VPN by the redistribution process. We will restore the correct network commands on R5 and see that the topology comes back up.

```
R5#conf t
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#network 172.16.15.5 0.0.0.0 area 0
R5(config-router)#end
R5#
%OSPF-5-ADJCHG: Process 100, Nbr 192.1.1.1 on FastEthernet0/0 from LOADING to FULL,
Loading Done
```

Once this is accomplished our routes will return to both R1 and R6.

```
R1#sh ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O       172.16.26.0 [110/3] via 172.16.15.5, 00:00:32, FastEthernet0/0
     25.0.0.0/32 is subnetted, 2 subnets
O E2    25.25.25.2 [110/1] via 172.16.15.5, 00:00:32, FastEthernet0/0
O E2    25.25.25.5 [110/1] via 172.16.15.5, 00:00:32, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O       192.1.6.6 [110/4] via 172.16.15.5, 00:00:32, FastEthernet0/0
```

On R6 we see the same.

```
R6#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O       172.16.15.0 [110/3] via 172.16.26.2, 00:01:19, FastEthernet0/1
     25.0.0.0/32 is subnetted, 2 subnets
O E2    25.25.25.2 [110/1] via 172.16.26.2, 00:18:55, FastEthernet0/1
O E2    25.25.25.5 [110/1] via 172.16.26.2, 00:11:36, FastEthernet0/1
     192.1.1.0/32 is subnetted, 1 subnets
O       192.1.1.1 [110/4] via 172.16.26.2, 00:01:09, FastEthernet0/1
```

We also have restored reachability as a result.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Redistribution Issues

As we have mentioned earlier it is a necessity to use redistribution in our topology for all PE-CE routing protocols other than eBGP peering. So this means that we need an understanding of the issues related to the incompatibility of metrics used by our individual protocols. This extends to both our MP-BGP and OSPF protocols. We will explore the nature of these issues as they relate to OSPF in our topology by altering the configuration on our PEs. These aspects as they relate to redistribution will impact both our IGP Redistribution and MP-BGP Redistribution.

IGP Redistribution

We will explore the issue related to redistribution and our VRF aware IGPs by looking at some of the possible problems related to the configuration process.

MP-BGP Redistribution

Regarding redistribution of the IGP prefixes into the MP-BGP protocol the most common issues that can be encountered fall into one category.

Lack of Redistribution – When redistribution is not used under the MP-BGP process, we will never see any prefixes advertised to the remote PE devices.

Filtering Issues

As with any routing protocol, filtering whether it is in the form of access-lists, offset-lists, route-maps and distribute-lists can block prefixes from being allowed to enter the routing table of a device. VRF aware routing protocols like OSPF are no exception. These tools are can very easily be used to induce issues associated with our CE-PE routing.

Chapter Challenge: OSPF Sample Trouble Ticket

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH9-OSPF-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 9-3 below:

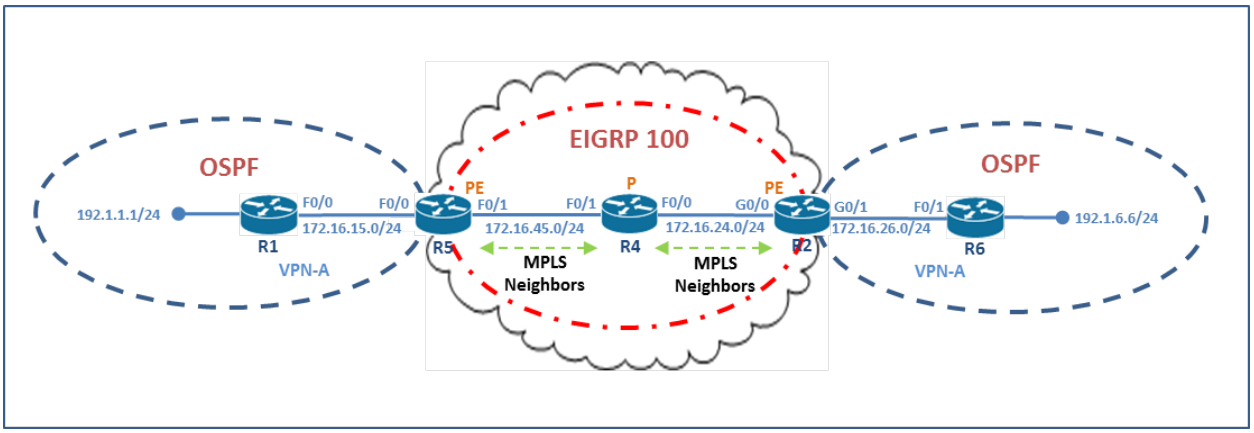


Figure 9-3: The Chapter Challenge Topology

Trouble Ticket #1

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Chapter Challenge: OSPF Sample Trouble Ticket Solution

The following section includes the solution to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Does the ping to 192.1.1.1 work or fail on R6?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

The pings are unsuccessful.

Step 2 - Fault Isolation:

First we will make sure that R5 can reach 192.1.1.1.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

R5 can reach the address in question. Now we need to see if R5 is advertising this prefix to the remote PE (R5).

```
R5#show ip bgp vpnv4 all neighbors 192.1.2.2 advertised-routes
BGP table version is 5, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
```

```
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 172.16.15.0/24    0.0.0.0                  0         32768 ?
*> 192.1.1.1/32     172.16.15.1              2         32768 ?

Total number of prefixes 2
```

The prefixes are being advertised. Are the routes making it into R2's database?

```
R2#show ip bgp vpnv4 rd 100:1 neighbors 192.1.5.5 routes
BGP table version is 5, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i172.16.15.0/24   192.1.5.5                0    100      0 ?
*>i192.1.1.1/32     192.1.5.5                2    100      0 ?

Total number of prefixes 2
```

These routes are making it into the routing table, but are they being advertise to R6?

```
R6#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.15.0 [110/2] via 172.16.26.2, 00:03:56, FastEthernet0/1
     192.1.1.0/32 is subnetted, 1 subnets
O IA    192.1.1.1 [110/3] via 172.16.26.2, 00:03:56, FastEthernet0/1
```

This mean that R6 has reachability to 192.1.1.1. Remember we are sourcing route routes from 192.1.6.6, and we need to see if we have reachability to R1 without specifying the source.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

This tells us that R1 has no reachability information for the either the network 192.1.6.6/32 or 172.16.26.0/24. We can see this via the output of the **show ip route ospf** command on R1.

```
R1#show ip route ospf

R1#
```

There is no knowledge of these routes on R1. We will move to R2 and see if these prefixes are being advertised via the Layer 3 VPN tunnel.

```
R2#show ip bgp vpnv4 all neighbors 192.1.5.5 advertised-routes

Total number of prefixes 0
```

These prefixes are not advertised via the tunnel. We know that they would need to be redistributed into the routing process. This can best be verified using the **show run | sec router bgp** command.

```
R2#show run | sec router bgp
router bgp 65000
 bgp log-neighbor-changes
 neighbor 192.1.5.5 remote-as 65000
 neighbor 192.1.5.5 update-source Loopback0
 !
 address-family ipv4
  neighbor 192.1.5.5 activate
  no auto-summary
  no synchronization
 exit-address-family
 !
 address-family vpnv4
  neighbor 192.1.5.5 activate
  neighbor 192.1.5.5 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf VPN-A
  redistribute eigrp 101
  no synchronization
 exit-address-family
```

We are redistributing the wrong process under the MP-BGP process.

Step 3 - Fault Remediation:

In this scenario, we will add the **redistribute ospf 1** command under the address-family.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute ospf 1
R2(config-router-af)#end
```

Now we will recheck to ensure that the prefixes are actually being advertised to the remote PE.

```
R2#show ip bgp vpnv4 all neighbors 192.1.5.5 advertised-routes
BGP table version is 9, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2 (default for vrf VPN-A)
*> 172.16.26.0/24   0.0.0.0                  0         32768 ?
*> 192.1.6.6/32     172.16.26.6              2         32768 ?

Total number of prefixes 2
```

We see that they are being advertise, but is R5 accepting these prefixes?

```
R5#show ip bgp vpnv4 rd 100:2 neighbor 192.1.2.2 routes
BGP table version is 9, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:2
*>i172.16.26.0/24   192.1.2.2                0    100      0 ?
*>i192.1.6.6/32     192.1.2.2                2    100      0 ?

Total number of prefixes 2
```

The prefixes are being accepted. Are they being redistributed into the OSPF vrf VPN-A instance?

```
R1#show ip route ospf

R1#
```

The prefixes in question are not appearing on R1. Now we will need to see if they are being redistributed into the **OSPF 100 vrf VPN-A**. We can see this via the **show run | sec router ospf 100** command.

```
R5#show run | sec router ospf 100
router ospf 100 vrf VPN-A
 router-id 192.1.5.5
 domain-id 1.1.1.1
 log-adjacency-changes
 area 0 sham-link 25.25.25.5 25.25.25.2
 network 172.16.15.5 0.0.0.0 area 0
```

As we can see we are not redistributing the BGP process. To satisfy the need for this requirement we will redistribute MP-BGP into OSPF.

```
5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 100 vrf VPN-A
R5(config-router)#redistribute bgp 65000 subnets
R5(config-router)#end
```

Now we will look again at R1.

```
R1#show ip route ospf
     172.16.0.0/24 is subnetted, 2 subnets
O IA    172.16.26.0 [110/2] via 172.16.15.5, 00:01:54, FastEthernet0/0
     192.1.6.0/32 is subnetted, 1 subnets
O IA    192.1.6.6 [110/3] via 172.16.15.5, 00:01:54, FastEthernet0/0
```

Based on this we should have full reachability.

Step 4 - Verification of Remediation

Once the error has been isolated and remediated it is highly recommended to verify that the Trouble Ticket has been repaired using the same method of the initial fault verification.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R6#
```

The issues have been corrected.

Chapter 10: eBGP

BGP is the current standard protocol for inter-domain exterior routing. MPLS VPN networks use a version of this protocol called MP-BGP that plays the foundational role in the transportation of VPNv4 prefixes via a service provider's network.

Introduction to eBGP PE-CE Routing

BGP is the current standard protocol for inter-domain exterior routing. MPLS VPN networks use a version of this protocol called MP-BGP that plays the foundational role in the transportation of VPNv4 prefixes via a service provider's network. Traditionally, customers prefer to use BGP in their networks and, therefore, they want to deploy BGP as the PE-CE routing protocol of choice when migrating from a non-MPLS based to an MPLS VPN based network. This creates a homogeneous end-to-end routing policy. In an MPLS VPN network, BGP attributes for a VPN site are transported across the service provider backbone to another site in the same VPN transparently. This happens due to the fact that there is a single routing protocol being used across the service provider core and the customer sites. Redistribution doesn't play a role in this type of deployment, because we are using the same protocol end-to-end.

Using BGP as the PE-CE peering protocol allows us to create the MPLS VPN environment in two different ways:

- BGP PE-CE VPN sites using different AS numbers
- BGP PE-CE VPN sites the same AS numbers

In this section we will implement both scenarios and explore the issues associated with each method. The stages of deploying BGP as the PE-CE protocol, no matter the AS combination follow this outline:

Step 1. Configure per VRF BGP routing contexts on PE routers

Configure per VRF BGP routing contexts for the VRF instances under the BGP routing process running on the PE routers.

Step 2. Define and activate BGP CE neighbors

Under each BGP VRF routing context we created, the remote BGP CE neighbors must defined on the PE routers and activated.

We will explore this configuration in detail by deploying eBGP as the CE-PE routing protocol according to Figure 10-1.

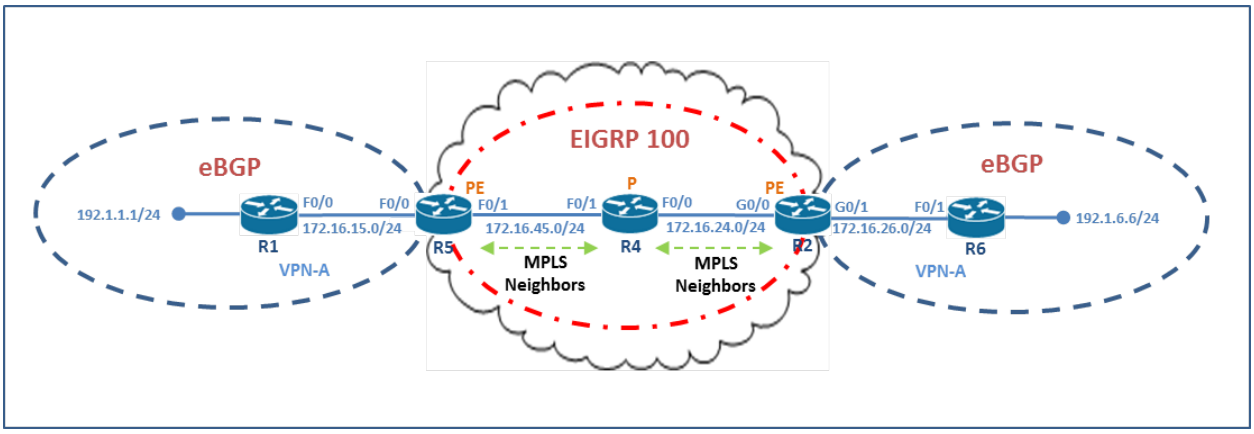


Figure 10-1: eBGP CE-PE Routing Topology

VRF Aware eBGP (Different ASN)

We will start this process first on R1 and R5, by modifying the ipv4 VRF aware process on R5.

```
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#neighbor 172.16.15.1 remote-as 1
R5(config-router-af)#neighbor 172.16.15.1 activate
R5(config-router-af)#end
```

We will go to R1 and create the eBGP process on that router as well. Please note that we are not using anything special in this process. We will not create an address-family on R1.

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router bgp 1
R1(config-router)#neighbor 172.16.15.5 remote-as 65000
R1(config-router)#network 192.1.1.0 mask 255.255.255.0
%BGP-5-ADJCHANGE: neighbor 172.16.15.5 Up
R1(config-router)#end
```

Now if we did this correctly R5 will have a route to the network 192.1.1.0/24.

```
R5#show ip route vrf VPN-A bgp
B    192.1.1.0/24 [20/0] via 172.16.15.1, 00:01:56
```

We see the route exists, but the final test is to successfully ping the address 192.1.1.1 from R5.

```
R5#ping vrf VPN-A 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

We will repeat this process on R2 and R6.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#neighbor 172.16.26.6 remote-as 6
R2(config-router-af)#neighbor 172.16.26.6 activate
R2(config-router-af)#end
```

Now for the normal routing process on R6.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router bgp 6
R6(config-router)#neighbor 172.16.26.2 remote-as 65000
R6(config-router)#network 192.1.6.0
%BGP-5-ADJCHANGE: neighbor 172.16.26.2 Up
```

Again we will look to see if R2 has learned the prefix for 192.1.6.0/24 via OSPF.

```
R2#show ip route vrf VPN-A bgp
B    192.1.6.0/24 [20/0] via 172.16.26.6, 00:00:47
B    192.1.1.0/24 [200/0] via 192.1.5.5, 00:05:52
```

Based on the fact that we are running eBGP as our PE-CE routing protocol we do not have to concern ourselves with redistribution. As we can see in the output provided above R2 knows about the network 192.1.1.0/24 already. We can see this prefix being communicated into the Layer 3 VPN by using the **show ip bgp vpnv4 all** command on R5.

```
R5#show ip bgp vpnv4 rd 100:1 neighbors 192.1.2.2 advertised-routes
BGP table version is 18, local router ID is 192.1.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN-A)
*> 192.1.1.0        172.16.15.1              0             0 1 i

Total number of prefixes 1
```

Now to repeat the process on R2 we can see what routes we learn from R5.

```
R2#show ip bgp vpnv4 rd 100:1 neighbors 192.1.5.5 routes
BGP table version is 18, local router ID is 192.1.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i192.1.1.0        192.1.5.5                0    100      0 1 i

Total number of prefixes 1
```

We are learning this prefix and placing it into the routing table and we can see that the prefixes are being communicated to the CE by the current eBGP processes running between them.

```
R1#show ip route bgp
B    192.1.6.0/24 [20/0] via 172.16.15.5, 00:05:54
```

We will repeat the process on R6.

```
R6#show ip route bgp
B    192.1.1.0/24 [20/0] via 172.16.26.2, 00:06:44
```

The question now is do we have reachability?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

We do have reachability, but we have a situation now that is very similar to the encounted with static routing. R1 and R6 only know about the prefixes that they are exchanging. They know nothing of the prefixes that connect them to their respective PEs. We can see this by pinging 192.1.1.1 without specifying the source.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

We will need to address this issue of reachability. At this time we will simply advertise the connected interfaces on R5 and R2 just like we did with static routing.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router bgp 65000
R5(config-router)#address-family ipv4 vrf VPN-A
R5(config-router-af)#redistribute connected
R5(config-router-af)#end
```

Now for R2.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#redistribute connected
R2(config-router-af)#end
```

With this done we will return to R1 and R6 and look at the routing tables again.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.26.0 [20/0] via 172.16.15.5, 00:00:46
C       172.16.15.0 is directly connected, FastEthernet0/0
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.15.5, 00:00:46
B       25.25.25.5 [20/0] via 172.16.15.5, 00:01:20
B    192.1.6.0/24 [20/0] via 172.16.15.5, 00:03:58
C    192.1.1.0/24 is directly connected, Loopback0
```

On R6.

```
R6#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.26.0 is directly connected, FastEthernet0/1
B       172.16.15.0 [20/0] via 172.16.26.2, 00:01:56
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.26.2, 00:01:30
B       25.25.25.5 [20/0] via 172.16.26.2, 00:01:56
C    192.1.6.0/24 is directly connected, Loopback0
B    192.1.1.0/24 [20/0] via 172.16.26.2, 00:04:26
```

Based on this we should have complete reachability.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
R6#
```

VRF Aware eBGP (Same ASN)

We know need to look at what happens when we perform eBGP PE-CE routing and use identical AS numbers on each end of the topology. We will emulate this by modifying the configuration on R2 such that it will be using AS 1 to communicate to R6.

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router bgp 65000
R2(config-router)#address-family ipv4 vrf VPN-A
R2(config-router-af)#no neighbor 172.16.26.6
R2(config-router-af)#neighbor 172.16.26.6 remote-as 1
R2(config-router-af)#neighbor 172.16.26.6 activate
R2(config-router-af)#end
```

Now we will move to R6 and remove the bgp process and recreate it.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#no router bgp 6
R6(config)#
%BGP-5-ADJCHANGE: neighbor 172.16.26.2 Down BGP protocol initialization
R6(config)#router bgp 1
R6(config-router)#no synchronization
R6(config-router)# bgp log-neighbor-changes
R6(config-router)# network 172.16.0.0
R6(config-router)# network 192.1.6.0
R6(config-router)# neighbor 172.16.26.2 remote-as 65000
R6(config-router)# no auto-summary
R6(config-router)#end
R6#
%BGP-5-ADJCHANGE: neighbor 172.16.26.2 Up
```

What behavior can we expect in this topology now? Remember how BGP prevents loops.

```
R6#show ip route bgp
     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.15.0 [20/0] via 172.16.26.2, 00:01:05
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.26.2, 00:01:05
B       25.25.25.5 [20/0] via 172.16.26.2, 00:01:05
```

Note that R6 does not see the routes originated on R1 in AS 1. We see all the connected prefixes originating on R5, but nothing from R1. We will also look at R1.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.26.0 [20/0] via 172.16.15.5, 00:00:44
C       172.16.15.0 is directly connected, FastEthernet0/0
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.15.5, 00:00:44
B       25.25.25.5 [20/0] via 172.16.15.5, 00:00:44
C    192.1.1.0/24 is directly connected, Loopback0
```

Again we are not learning anything from R6. Why is this?

The prefixes are not being advertised because of the fact that they have the same ASN in their updates. We have a few tools that will allow us to handle this problem.

AS-Override

Our first option would be to configure one of the PE routers such that it will override a given site's ASN with the providers ASN. We do this on a neighbor-by-neighbor basis. We will illustrate this process on R5.

```
R5#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)# router bgp 65000
R5(config-router)#   address-family ipv4 vrf VPN-A
R5(config-router-af)#  neighbor 172.16.15.1 as-override
R5(config-router-af)# end
```

Now we will look at R1 and see if we can see the 192.1.6.0/24 prefix.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.26.0 [20/0] via 172.16.15.5, 00:02:58
C       172.16.15.0 is directly connected, FastEthernet0/0
```



```
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.15.5, 00:02:58
B       25.25.25.5 [20/0] via 172.16.15.5, 00:02:58
B    192.1.6.0/24 [20/0] via 172.16.15.5, 00:00:28
C    192.1.1.0/24 is directly connected, Loopback0
```

The prefix is present. We can look closer at it via the **show ip bgp 192.1.6.0** command.

```
R1#show ip bgp 192.1.6.0
BGP routing table entry for 192.1.6.0/24, version 8
Paths: (1 available, best #1, table Default-IP-Routing-Table)
Flag: 0x820
  Not advertised to any peer
  65000 65000
    172.16.15.5 from 172.16.15.5 (192.1.5.5)
      Origin IGP, localpref 100, valid, external, best
```

Note that we have overridden the source ASN with the service providers ASN of 65000. Thus the prefix is allowed to enter the routing table.

AllowAS-in

There is another option that we can use to allow these prefixes to appear in the CE routing table. We will use this tool on R6. This is a very simple approach to the problem. We will merely inform R6 to allow its own AS path to be attached to routes it learns. In effect we are turning off the loop prevention process to allow this to happen. So take very careful measure if you ever use this method.

We will proceed to R6 and make the configuration changes.

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router bgp 1
R6(config-router)#neighbor 172.16.26.2 allowas-in 1
R6(config-router)#end
```

We can use the show ip route bgp to see if the prefix for 192.1.1.0/24 is now in our routing table.

```
R6#show ip route bgp
     172.16.0.0/24 is subnetted, 2 subnets
B       172.16.15.0 [20/0] via 172.16.26.2, 00:12:06
     25.0.0.0/32 is subnetted, 2 subnets
B       25.25.25.2 [20/0] via 172.16.26.2, 00:28:34
B       25.25.25.5 [20/0] via 172.16.26.2, 00:12:06
B    192.1.1.0/24 [20/0] via 172.16.26.2, 00:00:36
```

All that remains now is to determine if we have reachability in the topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
!!!!!
```



```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

What if we source the ping from the connected interface.

```
R6#ping 192.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

eBGP PE-CE Common Issues

The primary issues that affect PE-CE eBGP routing configurations fall into just two operational domains. As a general rule, and to keep the failure domains associated to the individual topic of this chapter we will focus on the aspect of eBGP VRF aware routing as it applies to the PE – CE communication process.

The main issues that PE-CE BGP Routing brings us fall into two categories: BGP Configuration Errors and Filtering Issues. We will explore each of these categories in the sample topology illustrated in Figure 10-2.

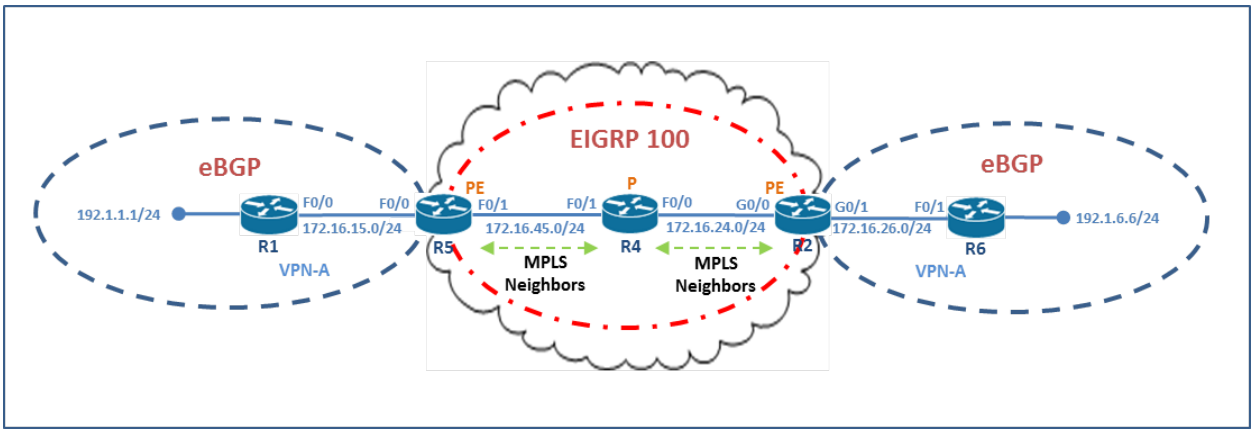


Figure 10-2: eBGP PE-CE Topology

We will explore each failure category in turn. Remember that for the purposes of our discussions we will consider the service provider cloud to be operational.

eBGP Configuration Errors

There are actually two areas where OSPF configuration issues can impact the performance of our MPLS domain. Primarily these two locations are at the CE and at the PE. We will look at the simplest of these locations first.

CE eBGP

At the CE we will use the standard BGP routing protocol that we have used throughout our studies. Due to the fact that the CE router has no knowledge of the concept of the virtual routing and forwarding instance we have no level of complication introduced at this phase. The typical issues associated with eBGP is to leave out the network commands.

PE eBPG

VRF aware routing is required at the PE in order to get eBGP to work as the routing protocol between the PE and CE devices while simultaneously supporting the Layer 3 VPN essential to our service provider cloud. In this case the most common problem is to forget to activate the ipv4 vrf neighborship.

Filtering Issues

As with any routing protocol, filtering whether it is in the form of access-lists, offset-lists, route-maps and distribute-lists can block prefixes from being allowed to enter the routing table of a device. VRF

aware routing protocols like eBGP are no exception. These tools can be very easily used to induce issues associated with our CE-PE routing.

Chapter Challenge: eBGP Sample Trouble Ticket

The following section includes one sample Trouble Ticket designed to challenge the troubleshooting skills that have been developed in all previous sections of this chapter. This Trouble Ticket was designed using the Routing & Switching rental racks at www.ProctorLabs.com with the initial configuration provided in the file ***MPLS-CH10-eBGP-TT-INITIAL.txt***. Keep in mind this sample Trouble Ticket was also tested against home practice racks and the most popular router emulators.

The network topology used in this section is shown in Figure 10-3 below:

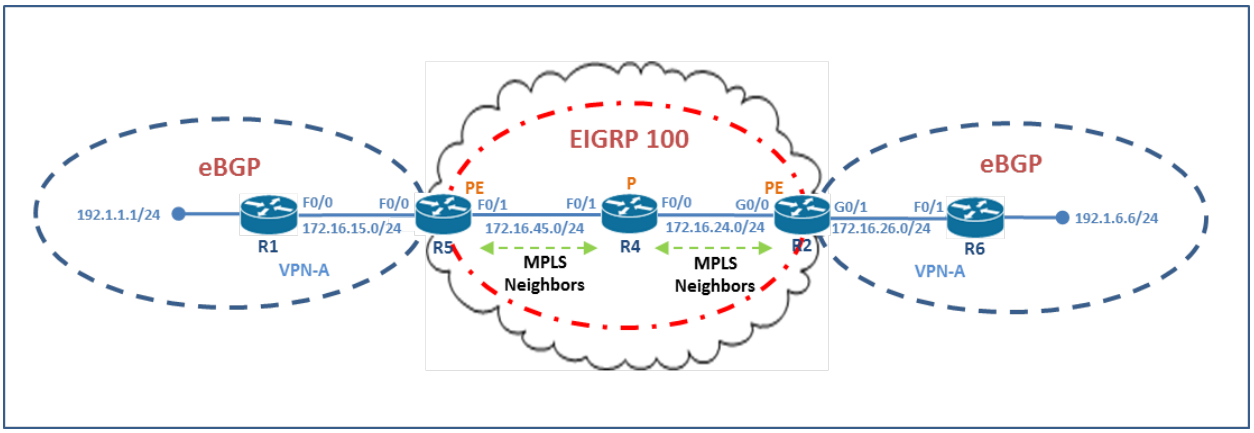


Figure 10-3: The Chapter Challenge Topology

Trouble Ticket #1

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Chapter Challenge: eBGP Sample Trouble Ticket Solution

The following section includes the solution to the one Trouble Ticket presented in the previous section.

Trouble Ticket #1 Solution

You supervisor has brought to your attention the fact that R6 cannot reach the address 192.1.1.1 on host R1. You are not allowed to remove any configuration in the remediation of this issue. Multiple issues exist in this topology.

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

Step 1 - Fault Verification:

Does the ping to 192.1.1.1 work or fail on R6?

```
R6#ping 192.1.1.1 source lo 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.1.6.6
.....
Success rate is 0 percent (0/5)
```

The pings are unsuccessful.

Step 2 - Fault Isolation:

<SOLUTION FORTHCOMING>