Cisco IOS NAT − Integration with MPLS VPN

Document ID: 112084

Contents

Introduction

Cisco IOS® Network Address Translation (NAT) software allows access to shared services from multiple MPLS VPNs, even when the devices in the VPNs use IP addresses that overlap. Cisco IOS NAT is VRF−aware and can be configured on provider edge routers within the MPLS network.

Note: MPLS in IOS is supported only with legacy NAT. At this time, there is no support in Cisco IOS for NAT NVI with MPLS.

The deployment of MPLS VPNs is projected to increase rapidly over the next several years. The benefits of a common network infrastructure that permits rapid expansion and flexible connectivity options will undoubtedly drive further growth in services that can be offered to the Internetwork community.

However, barriers to growth still remain. IPv6 and its promise of an IP address space that exceeds the connectivity needs for the foreseeable future is still in the early phases of deployment. Existing networks commonly use private IP addressing schemes as defined within RFC 1918 . Network address translation is often used to interconnect networks when address spaces overlap or duplication exists.

Service providers and enterprises that have network application services they want to offer or share with customers and partners will want to minimize any connectivity burden placed on the user of the service. It is desirable, even mandatory, to extend the offering to as many potential users as needed to achieve the desired goals or return. The IP addressing scheme in use must not be a barrier that excludes potential users.

By deploying Cisco IOS NAT within the common MPLS VPN infrastructure, communications service providers can relieve some of the connectivity burden on customers and accelerate their ability to link more shared application services to more consumers of those services.

Benefits from NAT MPLS Integration

NAT integration with MPLS has benefits for both service providers and their enterprise customers. It offers service providers more options to deploy shared services and to provide access to those services. Additional service offerings can be a differentiator over competitors.

For Service Provider	For VPN
More service offerings	Reduced costs
Increased access options	Simpler access
Increased revenue	Addressing flexibility

Enterprise customers seeking to outsource some of their current workload can also benefit from wider offerings by service providers. Shifting the burden of performing any necessary address translation to the service provider network relieves them of a complicated administrative task. Customers may continue to use private addressing, yet maintain access to shared services and the Internet. Consolidating the NAT function within the service provider network may also lower the total cost to enterprise customers since the customer edge routers do not have to perform the NAT function.

Design Considerations

When considering designs that will invoke NAT within the MPLS network, the first step is to determine the service needs from an application point of view. You will need to consider the protocols used and any special client/server communication imposed by the application. Make sure that the necessary support for the protocols employed are supported and handled by Cisco IOS NAT. A list of supported protocols is provided in the document Cisco IOS NAT Application Layer Gateways.

Next, it will be necessary to determine the expected usage of the shared service and the anticipated traffic rate in packets−per−second. NAT is a router CPU−intensive function. Therefore, performance requirements will be a factor in selecting a particular deployment option and to determine the number of NAT devices involved.

Also, consider any security issues and precautions that should be taken. Although MPLS VPNs, by definition, are private and effectively separate traffic, the shared service network is generally common among many VPNs.

Deployment Scenarios

There are two options for NAT deployment within the MPLS provider edge:

- Centralized with egress NAT PEs
- Distributed with ingress NAT PEs

Some advantages to configuring the NAT function at the egress point of the MPLS network nearest to the shared service network include:

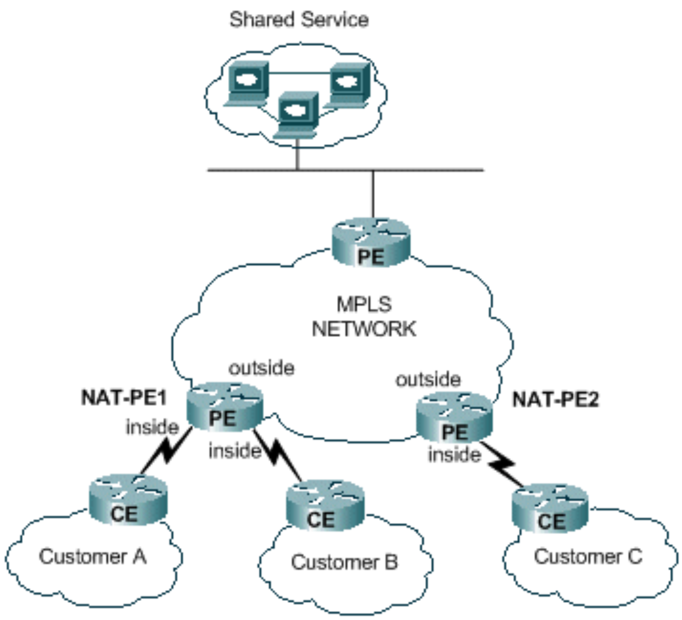
- A centralized configuration that promotes simpler service provisioning
- Simplified troubleshooting
- Enhanced operational scalability
- Decreased IP address allocation requirements

However, the advantages are offset by a reduction in scalability and performance. This is the main tradeoff that must be considered. Of course, the NAT function can also be performed within the customer networks if it is determined that integration of this feature with an MPLS network is not desirable.

Ingress PE NAT

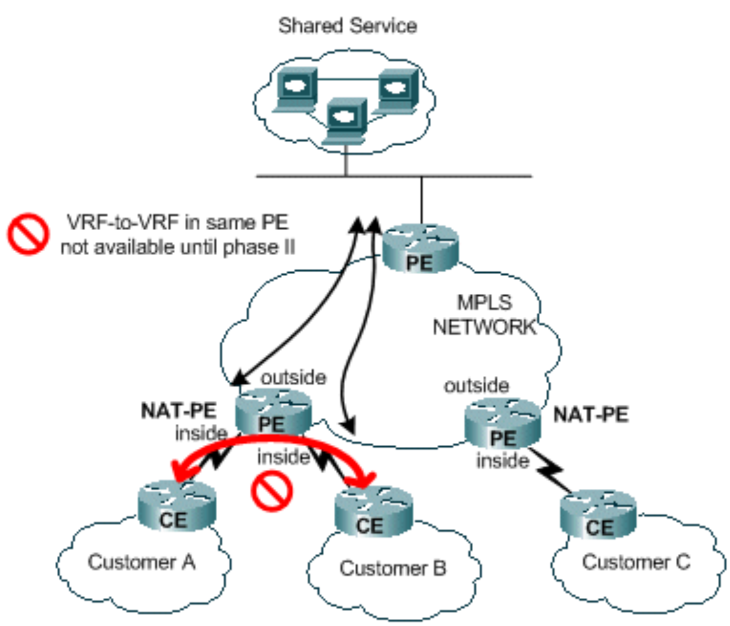
NAT can be configured at the MPLS network ingress PE router as shown in Figure 1. With this design, scalability is maintained to a large extent while performance is optimized by distributing the NAT function over many edge devices. Each NAT PE handles traffic for sites locally connected to that PE. NAT rules and access control lists or route maps control which packets require translation.

Figure 1: Ingress PE NAT



There is a restriction that prevents NAT between two VRFs while also providing NAT to a shared service as shown in Figure 2. This is due to the requirement to designate interfaces as NAT inside and outside interfaces. Support for connections between VRFs in a single PE is planned for a future Cisco IOS release.

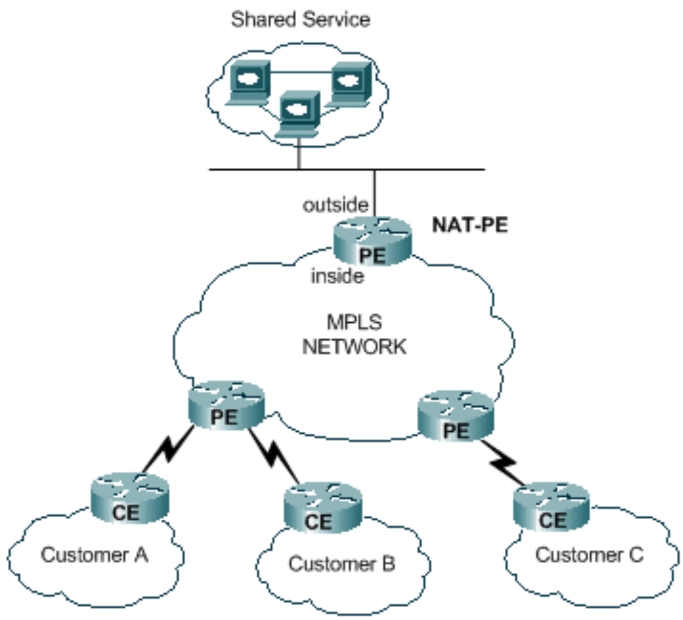
Figure 2: Business to Business



Egress PE NAT

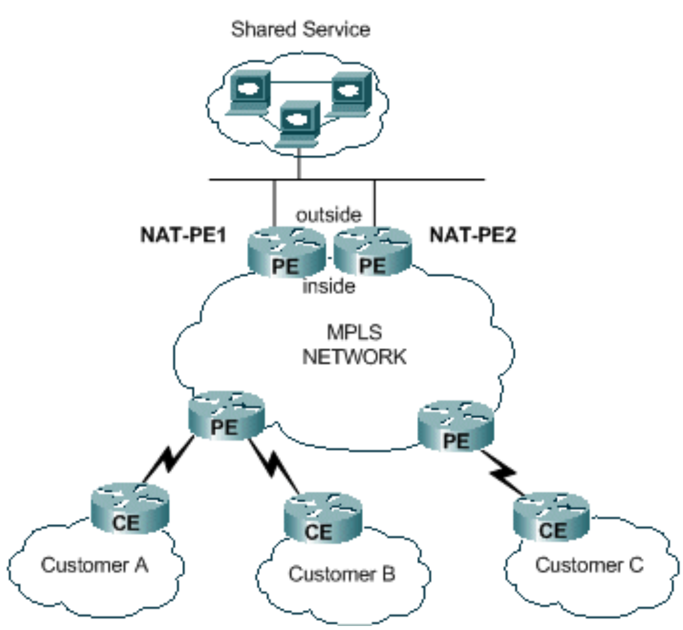
NAT can be configured at the MPLS network egress PE router as shown in Figure 3. With this design, scalability is reduced to some degree since the central PE must maintain routes for all customer networks that access the shared service. The application performance requirements must also be considered so that the traffic does not overburden the router that must translate the IP addresses of the packets. Because NAT occurs centrally for all customers using this path, IP address pools can be shared; thus, the total number of subnets required is reduced.

Figure 3: Egress PE NAT



Multiple routers could be deployed to increase the scalability of the egress PE NAT design as shown in Figure 4. In this scenario, customer VPNs could be provisioned on a specific NAT router. Network address translation would occur for the aggregate traffic to and from the shared service for that set of VPNs. For example, traffic from the VPNs for Customer A and B could use NAT−PE1, while traffic to and from the VPN for customer C uses NAT−PE2. Each NAT PE would carry traffic only for the specific VPNs defined and only maintain routes back to the sites in those VPNs. Separate NAT address pools could be defined within each of the NAT PE routers so that packets are routed from the shared service network to the proper NAT PE for translation and routing back to the customer VPN.

Figure 4: Multiple Egress PE NAT



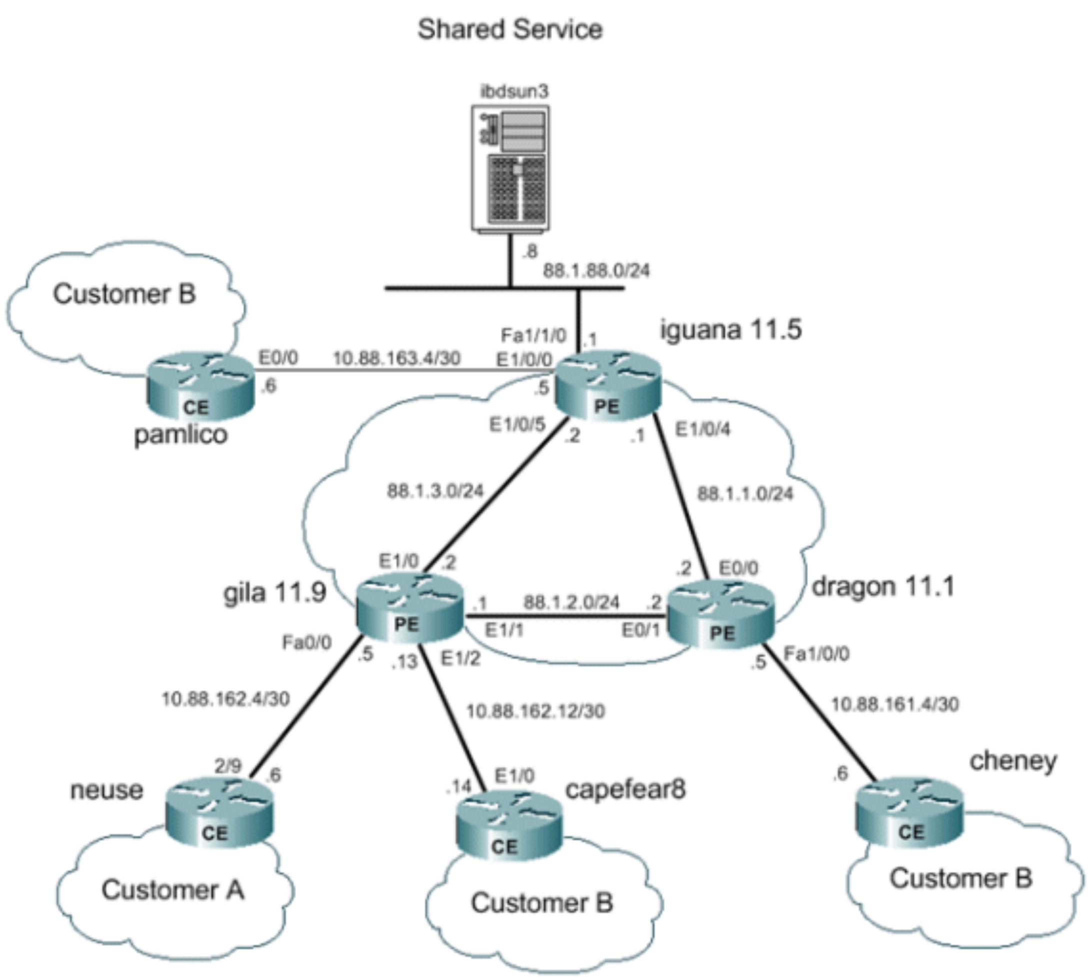
The centralized design does impose a restriction on how the shared service network must be configured. Specifically, use of import/export of MPLS VPN routes between a shared service VPN and customer VPNs is not possible. This is due to the nature of MPLS operation as specified by RFC 2547 . When routes are imported and exported using the extended communities and route descriptors, NAT cannot determine the source VPN from the packet coming into the central NAT PE. The usual case is to make the shared service network a generic interface rather than a VRF interface. A route to the shared service network is then added in the central NAT PE for each VRF table associated with a customer VPN needing access to the shared service as part of the provisioning process. This is described in more detail later.

Deployment Options and Configuration Details

This section includes some details related to each of the deployment options. The examples are all taken from the network shown in Figure 5. Refer to this diagram for the rest of this section.

Note: In the network used to illustrate the operation of VRF NAT for this paper, only PE routers are included. There are no core P routers. However, the essential mechanisms can still be seen.

Figure 5: VRF NAT Configuration Example



Egress PE NAT

In this example, the provider edge routers marked **gila** and **dragon** are configured as simple PE routers. The central PE near the shared service LAN (**iguana**) is configured for NAT. A single NAT pool is shared by each customer VPN that needs access to the shared service. The NAT is performed only on packets destined for the shared service host at 88.1.88.8.

Egress PE NAT Data Forwarding

With MPLS, each packet enters the network at an ingress PE and exits the MPLS network at an egress PE. The path of label switching routers traversed from ingress to egress is known as the label switched path (LSP). The LSP is unidirectional. A different LSP is used for return traffic.

When using egress PE NAT, a forwarding equivalence class (FEC) is effectively defined for all traffic from users of the shared service. In other words, all packets destined for the shared service LAN are members of a common FEC. A packet is assigned to a particular FEC just once at the ingress edge of the network and follows the LSP to the egress PE. The FEC is designated in the data packet by adding a particular label.

Packet Flow to Shared Service from VPN

In order for devices in multiple VPNs that have overlapping address schemes to access a shared service host, NAT is required. When NAT is configured at the egress PE, the network address translation table entries will include a VRF identifier to differentiate duplicate addresses and ensure proper routing.

Figure 6: Packets Transmitted to Egress PE NAT

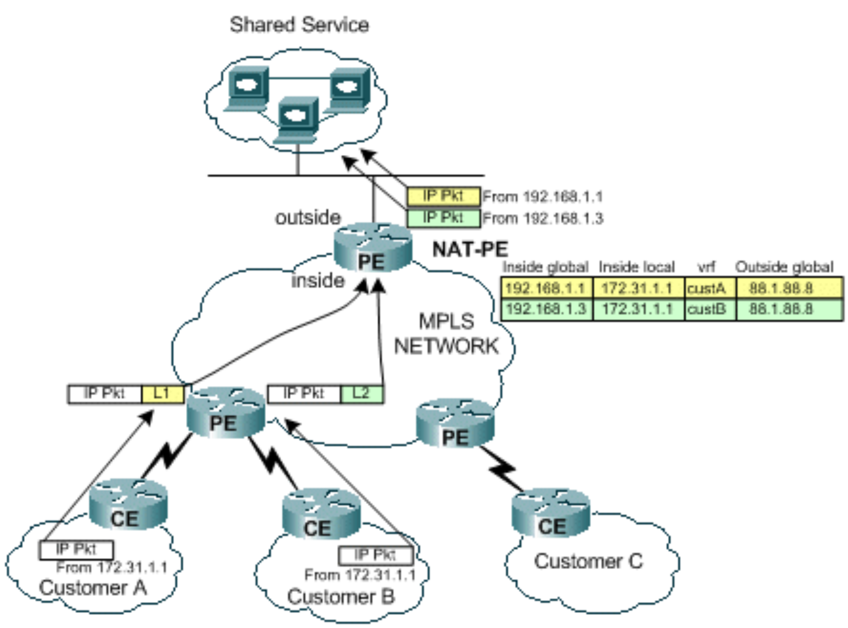


Figure 6 illustrates packets destined for a shared service host from two customer VPNs that have duplicate IP addressing schemes. The figure shows a packet originating at Customer A with a source address of 172.31.1.1 destined for a shared server at 88.1.88.8. Another packet from Customer B with the same source IP address is also sent to the same shared server. When the packets reach the PE router, a layer 3 lookup is done for the destination IP network in the forwarding information base (FIB).

The FIB entry tells the PE router to forward the traffic to the egress PE using a label stack. The bottom label in the stack is assigned by the destination PE router, in this case router **iguana**.

```
iguana#
show ip cef vrf custA 88.1.88.8

88.1.88.8/32, version 47, epoch 0, cached adjacency 88.1.3.2
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}
  via 88.1.11.5, 0 dependencies, recursive
```

```
    next hop 88.1.3.2, Ethernet1/0 via 88.1.11.5/32
    valid cached adjacency
    tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}

iguana# show ip cef vrf custB 88.1.88.8
88.1.88.8/32, version 77, epoch 0, cached adjacency 88.1.3.2
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0, 88.1.3.2, tags imposed: {28}
  via 88.1.11.5, 0 dependencies, recursive
    next hop 88.1.3.2, Ethernet1/0 via 88.1.11.5/32
    valid cached adjacency
    tag rewrite with Et1/0, 88.1.3.2, tags imposed: {28}
iguana#
```

We can see from the display that packets from VRF custA will have a tag value of 24 (0x18) and packets from VRF custB will have a tag value of 28 (0x1C).

In this case, because there are no P routers in our network, there is no additional tag imposed. Had there been core routers, an outside label would have been imposed and the normal process of label swapping would have taken place within the core network until the packet reached the egress PE.

Since the **gila** router is directly connected to the egress PE, we see that the tag is popped before it is ever added:

```
gila#
show tag-switching forwarding-table

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     Pop tag     88.1.1.0/24       0          Et1/1      88.1.2.2
       Pop tag     88.1.1.0/24       0          Et1/0      88.1.3.2
17     Pop tag     88.1.4.0/24       0          Et1/1      88.1.2.2
18     Pop tag     88.1.10.0/24      0          Et1/1      88.1.2.2
19     Pop tag     88.1.11.1/32      0          Et1/1      88.1.2.2
20     Pop tag     88.1.5.0/24       0          Et1/0      88.1.3.2
21     19          88.1.11.10/32     0          Et1/1      88.1.2.2
       22          88.1.11.10/32     0          Et1/0      88.1.3.2
22     20          172.18.60.176/32  0          Et1/1      88.1.2.2
       23          172.18.60.176/32  0          Et1/0      88.1.3.2
23     Untagged    172.31.1.0/24[V]  4980       Fa0/0      10.88.162.6
24     Aggregate   10.88.162.4/30[V] 1920
25     Aggregate   10.88.162.8/30[V] 137104
26     Untagged    172.31.1.0/24[V]  570        Et1/2      10.88.162.14
27     Aggregate   10.88.162.12/30[V]   \
                                     273480
30     Pop tag     88.1.11.5/32      0          Et1/0      88.1.3.2
31     Pop tag     88.1.88.0/24      0          Et1/0      88.1.3.2
32     16          88.1.97.0/24      0          Et1/0      88.1.3.2
33     Pop tag     88.1.99.0/24      0          Et1/0      88.1.3.2
gila#

gila# show tag-switching forwarding-table  88.1.88.0 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
31     Pop tag     88.1.88.0/24      0          Et1/0      88.1.3.2
        MAC/Encaps=14/14, MRU=1504, Tag Stack{}
        005054D92A250090BF9C6C1C8847
        No output feature configured
    Per-packet load-sharing
gila#
```

The next displays depict echo packets as received by the egress PE NAT router (at interface E1/0/5 on **iguana**).

```
From CustA:
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 1 arrived at  16:21:34.8415; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                   = 00018
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                   = 1 (Bottom of Stack)
        MPLS: Time to Live                  = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 175
        IP: Flags           = 0X
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 5EC0 (correct)
        IP: Source address      = [172.31.1.1]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = 4AF1 (correct)
        ICMP: Identifier = 4713
        ICMP: Sequence number = 6957
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]

From CustB:
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 11 arrived at  16:21:37.1558; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
```

```
        MPLS: Label Value                    = 0001C
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                    = 1 (Bottom of Stack)
        MPLS: Time to Live                   = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 165
        IP: Flags           = 0X
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 5ECA (correct)
        IP: Source address      = [172.31.1.1]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = AD5E (correct)
        ICMP: Identifier = 3365
        ICMP: Sequence number = 7935
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

These pings result in the following entries being created in the NAT table in the egress PE router **iguana**. The specific entries created for the packets shown above can be matched by their ICMP identifier.

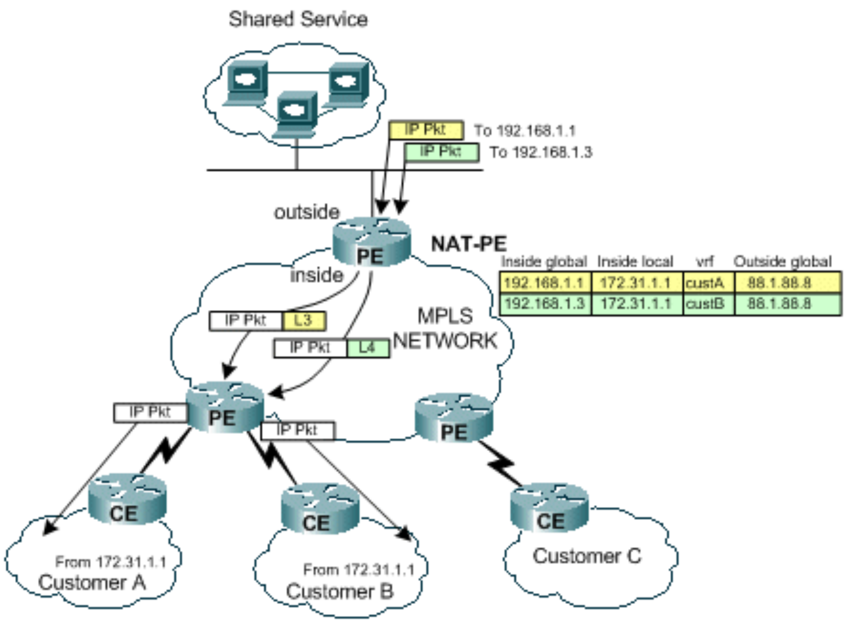
```
iguana#
show ip nat translations

Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.3:3365  172.31.1.1:3365    88.1.88.8:3365     88.1.88.8:3365
icmp 192.168.1.3:3366  172.31.1.1:3366    88.1.88.8:3366     88.1.88.8:3366
icmp 192.168.1.3:3367  172.31.1.1:3367    88.1.88.8:3367     88.1.88.8:3367
icmp 192.168.1.3:3368  172.31.1.1:3368    88.1.88.8:3368     88.1.88.8:3368
icmp 192.168.1.3:3369  172.31.1.1:3369    88.1.88.8:3369     88.1.88.8:3369
icmp 192.168.1.1:4713  172.31.1.1:4713    88.1.88.8:4713     88.1.88.8:4713
icmp 192.168.1.1:4714  172.31.1.1:4714    88.1.88.8:4714     88.1.88.8:4714
icmp 192.168.1.1:4715  172.31.1.1:4715    88.1.88.8:4715     88.1.88.8:4715
icmp 192.168.1.1:4716  172.31.1.1:4716    88.1.88.8:4716     88.1.88.8:4716
icmp 192.168.1.1:4717  172.31.1.1:4717    88.1.88.8:4717     88.1.88.8:4717

iguana#
show ip nat translations verbose


Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.3:3365  172.31.1.1:3365    88.1.88.8:3365     88.1.88.8:3365
    create 00:00:34, use 00:00:34, left 00:00:25, Map-Id(In): 2,
```

```
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:3366  172.31.1.1:3366    88.1.88.8:3366     88.1.88.8:3366
    create 00:00:34, use 00:00:34, left 00:00:25, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:3367  172.31.1.1:3367    88.1.88.8:3367     88.1.88.8:3367
    create 00:00:34, use 00:00:34, left 00:00:25, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:3368  172.31.1.1:3368    88.1.88.8:3368     88.1.88.8:3368
    create 00:00:34, use 00:00:34, left 00:00:25, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:3369  172.31.1.1:3369    88.1.88.8:3369     88.1.88.8:3369
    create 00:00:34, use 00:00:34, left 00:00:25, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.1:4713  172.31.1.1:4713    88.1.88.8:4713     88.1.88.8:4713
    create 00:00:37, use 00:00:37, left 00:00:22, Map-Id(In): 1,
Pro Inside global      Inside local       Outside local      Outside global
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:4714  172.31.1.1:4714    88.1.88.8:4714     88.1.88.8:4714
    create 00:00:37, use 00:00:37, left 00:00:22, Map-Id(In): 1,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:4715  172.31.1.1:4715    88.1.88.8:4715     88.1.88.8:4715
    create 00:00:37, use 00:00:37, left 00:00:22, Map-Id(In): 1,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:4716  172.31.1.1:4716    88.1.88.8:4716     88.1.88.8:4716
    create 00:00:37, use 00:00:37, left 00:00:22, Map-Id(In): 1,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:4717  172.31.1.1:4717    88.1.88.8:4717     88.1.88.8:4717
    create 00:00:37, use 00:00:37, left 00:00:22, Map-Id(In): 1,
    flags:
extended, use_count: 0, VRF : custA
iguana#
```

Packet Flow from Shared Service Back to Origin VPN

As packets flow back to devices that have accessed the shared service host, the NAT table is examined prior to routing (packets going from NAT outside interface to inside interface). Because each unique entry includes the corresponding VRF identifier, the packet can be translated and routed appropriately.

Figure 7: Packets Transmitted Back to Shared Service User



As shown in Figure 7, return traffic is first examined by NAT to find a matching translation entry. For example, a packet is sent to destination 192.168.1.1. The NAT table is searched. When the match is found, the appropriate translation is done to the inside local address (172.31.1.1) and then an adjacency lookup is performed using the associated VRF ID from the NAT entry.

```
iguana# show ip cef vrf custA 172.31.1.0
172.31.1.0/24, version 12, epoch 0, cached adjacency 88.1.3.1
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0/5, 88.1.3.1, tags imposed: {23}
  via 88.1.11.9, 0 dependencies, recursive
    next hop 88.1.3.1, Ethernet1/0/5 via 88.1.11.9/32
    valid cached adjacency
    tag rewrite with Et1/0/5, 88.1.3.1, tags imposed: {23}

iguana# show ip cef vrf custB 172.31.1.0
172.31.1.0/24, version 18, epoch 0, cached adjacency 88.1.3.1
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0/5, 88.1.3.1, tags imposed: {26}
  via 88.1.11.9, 0 dependencies, recursive
    next hop 88.1.3.1, Ethernet1/0/5 via 88.1.11.9/32
    valid cached adjacency
    tag rewrite with Et1/0/5, 88.1.3.1, tags imposed: {26}
iguana#
```

Label 23 (0x17) is used for traffic destined for 172.31.1.0/24 in VRF custA and label 26 (0x1A) is used for packets destined for 172.31.1.0/24 in VRF custB.

This is seen in the echo reply packets sent from router **iguana**:

```
To custA:
  DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 2 arrived at  16:21:34.8436; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 0090BF9C6C1C
        DLC:  Source      = Station 005054D92A25
        DLC:  Ethertype   = 8847 (MPLS)
```

```
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                  = 00017
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                  = 1 (Bottom of Stack)
        MPLS: Time to Live                 = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 56893
        IP: Flags           = 4X
        IP:       .1.. .... = don't fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 4131 (correct)
        IP: Source address      = [88.1.88.8]
        IP: Destination address = [172.31.1.1]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 0 (Echo reply)
        ICMP: Code = 0
        ICMP: Checksum = 52F1 (correct)
        ICMP: Identifier = 4713
        ICMP: Sequence number = 6957
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

When the packet reaches the destination PE router, the label is used to determine the appropriate VRF and interface to send the packet over.

```
gila#
show mpls forwarding-table

Local  Outgoing    Prefix            Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     Pop tag     88.1.1.0/24       0          Et1/1      88.1.2.2
       Pop tag     88.1.1.0/24       0          Et1/0      88.1.3.2
17     Pop tag     88.1.4.0/24       0          Et1/1      88.1.2.2
18     Pop tag     88.1.10.0/24      0          Et1/1      88.1.2.2
19     Pop tag     88.1.11.1/32      0          Et1/1      88.1.2.2
20     Pop tag     88.1.5.0/24       0          Et1/0      88.1.3.2
21     19          88.1.11.10/32     0          Et1/1      88.1.2.2
       22          88.1.11.10/32     0          Et1/0      88.1.3.2
22     20          172.18.60.176/32  0          Et1/1      88.1.2.2
       23          172.18.60.176/32  0          Et1/0      88.1.3.2
23     Untagged    172.31.1.0/24[V]  6306       Fa0/0      10.88.162.6
24     Aggregate   10.88.162.4/30[V] 1920
25     Aggregate   10.88.162.8/30[V] 487120
26     Untagged    172.31.1.0/24[V]  1896       Et1/2      10.88.162.14
```

```
27      Aggregate   10.88.162.12/30[V]   \
                                    972200
30      Pop tag     88.1.11.5/32      0          Et1/0      88.1.3.2
31      Pop tag     88.1.88.0/24      0          Et1/0      88.1.3.2
32      16          88.1.97.0/24      0          Et1/0      88.1.3.2
33      Pop tag     88.1.99.0/24      0          Et1/0      88.1.3.2
gila#
```

Configurations

Some extraneous information has been removed from the configurations for brevity.

```
IGUANA:
!
ip vrf custA
 rd 65002:100
 route-target export 65002:100
 route-target import 65002:100
!
ip vrf custB
 rd 65002:200
 route-target export 65002:200
 route-target import 65002:200
!
ip cef
mpls label protocol ldp
tag-switching tdp router-id Loopback0
!
interface Loopback0
 ip address 88.1.11.5 255.255.255.255
 no ip route-cache
 no ip mroute-cache
!
interface Loopback11
 ip vrf forwarding custA
 ip address 172.16.1.1 255.255.255.255
!
interface Ethernet1/0/0
 ip vrf forwarding custB
 ip address 10.88.163.5 255.255.255.252
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet1/0/4
 ip address 88.1.1.1 255.255.255.0
 ip nat inside
 no ip mroute-cache
 tag-switching ip
!
interface Ethernet1/0/5
 ip address 88.1.3.2 255.255.255.0
 ip nat inside
 no ip mroute-cache
 tag-switching ip
!
!
interface FastEthernet1/1/0
 ip address 88.1.88.1 255.255.255.0
 ip nat outside
 full-duplex
!
interface FastEthernet5/0/0
 ip address 88.1.99.1 255.255.255.0
 speed 100
 full-duplex
```

```
!
router ospf 881
 log-adjacency-changes
 redistribute static subnets
 network 88.1.0.0 0.0.255.255 area 0
!
router bgp 65002
 no synchronization
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 88.1.11.1 remote-as 65002
 neighbor 88.1.11.1 update-source Loopback0
 neighbor 88.1.11.9 remote-as 65002
 neighbor 88.1.11.9 update-source Loopback0
 neighbor 88.1.11.10 remote-as 65002
 neighbor 88.1.11.10 update-source Loopback0
 no auto-summary
 !
 address-family ipv4 multicast
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family vpnv4
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.1 send-community extended
 neighbor 88.1.11.9 activate
 neighbor 88.1.11.9 send-community extended
 no auto-summary
 exit-address-family
 !
 address-family ipv4
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.9 activate
 neighbor 88.1.11.10 activate
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf custB
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf custA
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
!
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload
ip nat inside source list 181 pool SSPOOL1 vrf custB overload
ip classless
ip route 88.1.88.0 255.255.255.0 FastEthernet1/1/0
ip route 88.1.97.0 255.255.255.0 FastEthernet5/0/0 88.1.99.2
ip route 88.1.99.0 255.255.255.0 FastEthernet5/0/0 88.1.99.2
ip route 192.168.1.0 255.255.255.0 Null0
ip route vrf custA 88.1.88.8 255.255.255.255 FastEthernet1/1/0 88.1.88.8 global
ip route vrf custB 10.88.208.0 255.255.240.0 10.88.163.6
ip route vrf custB 64.102.0.0 255.255.0.0 10.88.163.6
ip route vrf custB 88.1.88.8 255.255.255.255 FastEthernet1/1/0 88.1.88.8 global
ip route vrf custB 128.0.0.0 255.0.0.0 10.88.163.6
no ip http server
```

```
!
access-list 181 permit ip any host 88.1.88.8
!
```

GILA:

```
!
ip vrf custA
 rd 65002:100
 route-target export 65002:100
 route-target import 65002:100
!
ip vrf custB
 rd 65002:200
 route-target export 65002:200
 route-target import 65002:200
!
ip cef
mpls label protocol ldp
tag-switching tdp router-id Loopback0
!
interface Loopback0
 ip address 88.1.11.9 255.255.255.255
!
interface FastEthernet0/0
 ip vrf forwarding custA
 ip address 10.88.162.5 255.255.255.252
 duplex full
!
interface Ethernet1/0
 ip address 88.1.3.1 255.255.255.0
 no ip mroute-cache
 duplex half
 tag-switching ip
!
interface Ethernet1/1
 ip address 88.1.2.1 255.255.255.0
 no ip mroute-cache
 duplex half
 tag-switching ip
!
interface Ethernet1/2
 ip vrf forwarding custB
 ip address 10.88.162.13 255.255.255.252
 ip ospf cost 100
 duplex half
!
interface FastEthernet2/0
 ip vrf forwarding custA
 ip address 10.88.162.9 255.255.255.252
 duplex full
!
router ospf 881
 log-adjacency-changes
 redistribute static subnets
 network 88.1.0.0 0.0.255.255 area 0
 default-metric 30
!
router bgp 65002
 no synchronization
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 88.1.11.1 remote-as 65002
 neighbor 88.1.11.1 update-source Loopback0
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.5 remote-as 65002
```

```
 neighbor 88.1.11.5 update-source Loopback0
 neighbor 88.1.11.5 activate
 no auto-summary
 !
 address-family ipv4 vrf custB
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf custA
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family vpnv4
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.1 send-community extended
 neighbor 88.1.11.5 activate
 neighbor 88.1.11.5 send-community extended
 no auto-summary
 exit-address-family
!
ip classless
ip route vrf custA 172.31.1.0 255.255.255.0 FastEthernet0/0 10.88.162.6
ip route vrf custB 172.31.1.0 255.255.255.0 Ethernet1/2 10.88.162.14
!
```

Router **dragon** would have a configuration very similar to **gila**.

Import/Export of Route Targets Not Permitted

When the shared service network is configured as a VRF instance itself, central NAT at the egress PE is not possible. This is because the incoming packets cannot be distinguished and only one route back to the originating subnet is present at the egress PE NAT.

Note: The displays that follow are meant to illustrate the result of an invalid configuration.

The sample network was configured so that the shared service network was defined as a VRF instance (VRF name = sserver). Now, a display of the CEF table on the ingress PE shows this:

```
gila# show ip cef vrf custA 88.1.88.0
88.1.88.0/24, version 45, epoch 0, cached adjacency 88.1.3.2
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}
  via 88.1.11.5, 0 dependencies, recursive
    next hop 88.1.3.2, Ethernet1/0 via 88.1.11.5/32
    valid cached adjacency
    tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}
gila#

gila# show ip cef vrf custB 88.1.88.0
88.1.88.0/24, version 71, epoch 0, cached adjacency 88.1.3.2
0 packets, 0 bytes
  tag information set
    local tag: VPN-route-head
    fast tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}
  via 88.1.11.5, 0 dependencies, recursive
```

```
    next hop 88.1.3.2, Ethernet1/0 via 88.1.11.5/32
    valid cached adjacency
    tag rewrite with Et1/0, 88.1.3.2, tags imposed: {24}
gila#

iguana#
show tag-switching forwarding vrftags 24
Local  Outgoing    Prefix             Bytes tag  Outgoing   Next Hop
tag    tag or VC   or Tunnel Id       switched   interface
24     Aggregate   88.1.88.0/24[V]    10988
iguana#
```

Note: Notice how the tag value *24* is imposed for both VRF custA and VRF custB.

This display shows the routing table for the shared service VRF instance sserver :

```
iguana#
show ip route vrf sserver 172.31.1.1
Routing entry for 172.31.1.0/24
  Known via "bgp 65002", distance 200, metric 0, type internal
  Last update from 88.1.11.9 1d01h ago
  Routing Descriptor Blocks:
  * 88.1.11.9 (Default-IP-Routing-Table), from 88.1.11.9, 1d01h ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
```

Note: Only one route is present for the destination network from the egress PE router s (**iguana**) perspective.

Therefore, traffic from multiple customer VPNs could not be distinguished and return traffic could not reach the appropriate VPN. **In the case where the shared service must be defined as a VRF instance, the NAT function must be moved to the ingress PE.**

Ingress PE NAT

In this example, the provider edge routers marked **gila** and **dragon** are configured for NAT. A NAT pool is defined for each attached customer VPN that needs access to the shared service. The appropriate pool is used for each of the customer network addresses that are NATed. The NAT is performed only on packets destined for the shared service host at 88.1.88.8.

```
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat pool SSPOOL2 192.168.2.1 192.168.2.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload
ip nat inside source list 181 pool SSPOOL2 vrf custB overload
```

Note: In this scenario, shared pools are not supported. If the shared service LAN (at the egress PE) is connected through a generic interface, then the NAT pool may be shared.

A ping sourced from a duplicate address (172.31.1.1) within each of the networks attached to **neuse** and **capefear8** results in these NAT entries:

From **gila**:

```
gila#
show ip nat translations

Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.1:2139  172.31.1.1:2139    88.1.88.8:2139     88.1.88.8:2139
icmp 192.168.1.1:2140  172.31.1.1:2140    88.1.88.8:2140     88.1.88.8:2140
icmp 192.168.1.1:2141  172.31.1.1:2141    88.1.88.8:2141     88.1.88.8:2141
icmp 192.168.1.1:2142  172.31.1.1:2142    88.1.88.8:2142     88.1.88.8:2142
```

```
icmp 192.168.1.1:2143  172.31.1.1:2143    88.1.88.8:2143     88.1.88.8:2143
icmp 192.168.2.2:676   172.31.1.1:676     88.1.88.8:676      88.1.88.8:676
icmp 192.168.2.2:677   172.31.1.1:677     88.1.88.8:677      88.1.88.8:677
icmp 192.168.2.2:678   172.31.1.1:678     88.1.88.8:678      88.1.88.8:678
icmp 192.168.2.2:679   172.31.1.1:679     88.1.88.8:679      88.1.88.8:679
icmp 192.168.2.2:680   172.31.1.1:680     88.1.88.8:680      88.1.88.8:680
```

Note: The same inside local address (172.31.1.1) is translated to each of the defined pools according to the source VRF. The VRF can be seen in the **show ip nat translation verbose** command:

```
gila# show ip nat translations verbose
Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.1:2139  172.31.1.1:2139    88.1.88.8:2139     88.1.88.8:2139
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:2140  172.31.1.1:2140    88.1.88.8:2140     88.1.88.8:2140
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:2141  172.31.1.1:2141    88.1.88.8:2141     88.1.88.8:2141
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:2142  172.31.1.1:2142    88.1.88.8:2142     88.1.88.8:2142
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:2143  172.31.1.1:2143    88.1.88.8:2143     88.1.88.8:2143
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.2.2:676   172.31.1.1:676     88.1.88.8:676      88.1.88.8:676
    create 00:00:10, use 00:00:10, left 00:00:49, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.2.2:677   172.31.1.1:677     88.1.88.8:677      88.1.88.8:677
    create 00:00:10, use 00:00:10, left 00:00:49, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.2.2:678   172.31.1.1:678     88.1.88.8:678      88.1.88.8:678
    create 00:00:10, use 00:00:10, left 00:00:49, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.2.2:679   172.31.1.1:679     88.1.88.8:679      88.1.88.8:679
    create 00:00:10, use 00:00:10, left 00:00:49, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.2.2:680   172.31.1.1:680     88.1.88.8:680      88.1.88.8:680
    create 00:00:10, use 00:00:10, left 00:00:49, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
```

These displays show the routing information for each of the locally attached VPNs for customer A and customer B:

```
gila# show ip route vrf custA
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, 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 88.1.11.1 to network 0.0.0.0

     172.18.0.0/32 is subnetted, 2 subnets
B       172.18.60.179 [200/0] via 88.1.11.1, 00:03:59
B       172.18.60.176 [200/0] via 88.1.11.1, 00:03:59
     172.31.0.0/24 is subnetted, 1 subnets
S       172.31.1.0 [1/0] via 10.88.162.6, FastEthernet0/0
     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
B       10.88.0.0/20 [200/0] via 88.1.11.1, 00:03:59
B       10.88.32.0/20 [200/0] via 88.1.11.1, 00:03:59
C       10.88.162.4/30 is directly connected, FastEthernet0/0
C       10.88.162.8/30 is directly connected, FastEthernet2/0
B       10.88.161.8/30 [200/0] via 88.1.11.1, 00:04:00
     88.0.0.0/24 is subnetted, 2 subnets
B       88.1.88.0 [200/0] via 88.1.11.5, 00:04:00
B       88.1.99.0 [200/0] via 88.1.11.5, 00:04:00
S    192.168.1.0/24 is directly connected, Null0
B*   0.0.0.0/0 [200/0] via 88.1.11.1, 00:04:00

gila# show ip route vrf custB
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, 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

     64.0.0.0/16 is subnetted, 1 subnets
B       64.102.0.0 [200/0] via 88.1.11.5, 1d21h
     172.18.0.0/32 is subnetted, 2 subnets
B       172.18.60.179 [200/0] via 88.1.11.1, 1d21h
B       172.18.60.176 [200/0] via 88.1.11.1, 1d21h
     172.31.0.0/24 is subnetted, 1 subnets
S       172.31.1.0 [1/0] via 10.88.162.14, Ethernet1/2
     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
B       10.88.194.16/28 [200/100] via 88.1.11.1, 1d20h
B       10.88.208.0/20 [200/0] via 88.1.11.5, 1d21h
B       10.88.194.4/30 [200/100] via 88.1.11.1, 1d20h
B       10.88.163.4/30 [200/0] via 88.1.11.5, 1d21h
B       10.88.161.4/30 [200/0] via 88.1.11.1, 1d21h
C       10.88.162.12/30 is directly connected, Ethernet1/2
     11.0.0.0/24 is subnetted, 1 subnets
B       11.1.1.0 [200/100] via 88.1.11.1, 1d20h
     88.0.0.0/24 is subnetted, 2 subnets
B       88.1.88.0 [200/0] via 88.1.11.5, 1d21h
B       88.1.99.0 [200/0] via 88.1.11.5, 1d21h
S    192.168.2.0/24 is directly connected, Null0
B    128.0.0.0/8 [200/0] via 88.1.11.5, 1d21h
```

Note: A route for each of the NAT pools has been added from the static configuration. These subnets are subsequently imported into the shared server VRF at the egress PE router **iguana**:

```
iguana# show ip route vrf sserver

Routing Table: sserver
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, 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

     64.0.0.0/16 is subnetted, 1 subnets
B       64.102.0.0 [20/0] via 10.88.163.6 (custB), 1d20h
     172.18.0.0/32 is subnetted, 2 subnets
B       172.18.60.179 [200/0] via 88.1.11.1, 1d20h
B       172.18.60.176 [200/0] via 88.1.11.1, 1d20h
     172.31.0.0/24 is subnetted, 1 subnets
B       172.31.1.0 [200/0] via 88.1.11.9, 1d05h
     10.0.0.0/8 is variably subnetted, 8 subnets, 3 masks
B       10.88.194.16/28 [200/100] via 88.1.11.1, 1d20h
B       10.88.208.0/20 [20/0] via 10.88.163.6 (custB), 1d20h
B       10.88.194.4/30 [200/100] via 88.1.11.1, 1d20h
B       10.88.162.4/30 [200/0] via 88.1.11.9, 1d20h
B       10.88.163.4/30 is directly connected, 1d20h, Ethernet1/0/0
B       10.88.161.4/30 [200/0] via 88.1.11.1, 1d20h
B       10.88.162.8/30 [200/0] via 88.1.11.9, 1d20h
B       10.88.162.12/30 [200/0] via 88.1.11.9, 1d20h
     11.0.0.0/24 is subnetted, 1 subnets
B       11.1.1.0 [200/100] via 88.1.11.1, 1d20h
     12.0.0.0/24 is subnetted, 1 subnets
S       12.12.12.0 [1/0] via 88.1.99.10
     88.0.0.0/24 is subnetted, 3 subnets
C       88.1.88.0 is directly connected, FastEthernet1/1/0
S       88.1.97.0 [1/0] via 88.1.99.10
C       88.1.99.0 is directly connected, FastEthernet5/0/0
B    192.168.1.0/24 [200/0] via 88.1.11.9, 1d20h
B    192.168.2.0/24 [200/0] via 88.1.11.9, 01:59:23
B    128.0.0.0/8 [20/0] via 10.88.163.6 (custB), 1d20h
```

Configurations

Some extraneous information has been removed from the configurations for brevity.

```
GILA:
ip vrf custA
 rd 65002:100
 route-target export 65002:100
 route-target export 65002:1001
 route-target import 65002:100
!
ip vrf custB
 rd 65002:200
 route-target export 65002:200
 route-target export 65002:2001
 route-target import 65002:200
 route-target import 65002:10
!
ip cef
mpls label protocol ldp
!

interface Loopback0
 ip address 88.1.11.9 255.255.255.255
!
interface FastEthernet0/0
 ip vrf forwarding custA
 ip address 10.88.162.5 255.255.255.252
 ip nat inside
 duplex full
!
interface Ethernet1/0
 ip address 88.1.3.1 255.255.255.0
```

```
 ip nat outside
 no ip mroute-cache
 duplex half
 tag-switching ip
!
interface Ethernet1/1
 ip address 88.1.2.1 255.255.255.0
 ip nat outside
 no ip mroute-cache
 duplex half
 tag-switching ip
!
interface Ethernet1/2
 ip vrf forwarding custB
 ip address 10.88.162.13 255.255.255.252
 ip nat inside
 duplex half
!
router ospf 881
 log-adjacency-changes
 redistribute static subnets
 network 88.1.0.0 0.0.255.255 area 0
 default-metric 30
!
router bgp 65002
 no synchronization
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 88.1.11.1 remote-as 65002
 neighbor 88.1.11.1 update-source Loopback0
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.5 remote-as 65002
 neighbor 88.1.11.5 update-source Loopback0
 neighbor 88.1.11.5 activate
 no auto-summary
 !
 address-family ipv4 vrf custB
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf custA
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family vpnv4
 neighbor 88.1.11.1 activate
 neighbor 88.1.11.1 send-community extended
 neighbor 88.1.11.5 activate
 neighbor 88.1.11.5 send-community extended
 no auto-summary
 exit-address-family
!
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat pool SSPOOL2 192.168.2.1 192.168.2.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload
ip nat inside source list 181 pool SSPOOL2 vrf custB overload
ip classless
ip route vrf custA 172.31.1.0 255.255.255.0 FastEthernet0/0 10.88.162.6
ip route vrf custA 192.168.1.0 255.255.255.0 Null0
ip route vrf custB 172.31.1.0 255.255.255.0 Ethernet1/2 10.88.162.14
```

```
ip route vrf custB 192.168.2.0 255.255.255.0 Null0
!
access-list 181 permit ip any host 88.1.88.8
!
```

Note: The interfaces that face the customer networks are designated as NAT inside interfaces and the MPLS interfaces are designated as NAT outside interfaces.

```
iguana:
ip vrf custB
 rd 65002:200
 route-target export 65002:200
 route-target export 65002:2001
 route-target import 65002:200
 route-target import 65002:10
!
ip vrf sserver
 rd 65002:10
 route-target export 65002:10
 route-target import 65002:2001
 route-target import 65002:1001
!
ip cef distributed
mpls label protocol ldp
!

interface Loopback0
 ip address 88.1.11.5 255.255.255.255
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet1/0/0
 ip vrf forwarding custB
 ip address 10.88.163.5 255.255.255.252
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet1/0/4
 ip address 88.1.1.1 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 tag-switching ip
!
interface Ethernet1/0/5
 ip address 88.1.3.2 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 tag-switching ip
!
interface FastEthernet1/1/0
 ip vrf forwarding sserver
 ip address 88.1.88.1 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 full-duplex
!
router ospf 881
 log-adjacency-changes
 redistribute static subnets
 network 88.1.0.0 0.0.255.255 area 0
!
router bgp 65002
 no synchronization
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
```

```
neighbor 88.1.11.1 remote-as 65002
neighbor 88.1.11.1 update-source Loopback0
neighbor 88.1.11.9 remote-as 65002
neighbor 88.1.11.9 update-source Loopback0
neighbor 88.1.11.10 remote-as 65002
neighbor 88.1.11.10 update-source Loopback0
no auto-summary
!
address-family ipv4 multicast
no auto-summary
no synchronization
exit-address-family
!
address-family vpnv4
neighbor 88.1.11.1 activate
neighbor 88.1.11.1 send-community extended
neighbor 88.1.11.9 activate
neighbor 88.1.11.9 send-community extended
no auto-summary
exit-address-family
!
address-family ipv4
neighbor 88.1.11.1 activate
neighbor 88.1.11.9 activate
neighbor 88.1.11.10 activate
no auto-summary
no synchronization
exit-address-family
!
address-family ipv4 vrf sserver
redistribute connected
no auto-summary
no synchronization
exit-address-family
!
address-family ipv4 vrf custB
redistribute connected
redistribute static
no auto-summary
no synchronization
exit-address-family
```

Router **dragon** would have a configuration very similar to **gila**.

Packets Arriving at Central PE after Ingress PE NAT

The traces below illustrate the requirement for unique NAT pools when the destination shared service network is configured as a VRF instance. Again, refer to the diagram in Figure 5. The packets shown below were captured as they entered the MPLS IP interface e1/0/5 at router **iguana**.

Echo from Customer A VPN

Here, we see an echo request coming from source IP address 172.31.1.1 in VRF custA. The source address has been translated to 192.168.1.1 as specified by the NAT configuration:

```
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload

DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 1 arrived at  09:15:29.8157; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
```

```
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                    = 00019
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                    = 1 (Bottom of Stack)
        MPLS: Time to Live                   = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 0
        IP: Flags           = 0X
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 4AE6 (correct)
        IP: Source address      = [192.168.1.1]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = 932D (correct)
        ICMP: Identifier = 3046
        ICMP: Sequence number = 3245
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
        ICMP:
```

Echo from Customer B VPN

Here, we see an echo request coming from source IP address 172.31.1.1 in VRF custB. The source address has been translated to 192.168.2.1 as specified by the NAT configuration:

```
ip nat pool SSPOOL2 192.168.2.1 192.168.2.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL2 vrf custB overload

DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 11 arrived at  09:15:49.6623; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                    = 00019
```

```
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                   = 1 (Bottom of Stack)
        MPLS: Time to Live                  = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 15
        IP: Flags           = 0X
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 49D6 (correct)
        IP: Source address      = [192.168.2.2]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = AB9A (correct)
        ICMP: Identifier = 4173
        ICMP: Sequence number = 4212
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

Note: The MPLS label value is *0019* in both of the packets shown above.

Echo Reply to Customer A VPN

Next, we see an echo reply going back to the destination IP address 192.168.1.1 in VRF custA. The destination address is translated to 172.31.1.1 by the ingress PE NAT function.

```
To VRF custA:
  DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 2 arrived at  09:15:29.8198; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 0090BF9C6C1C
        DLC:  Source      = Station 005054D92A25
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                   = 0001A
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                   = 1 (Bottom of Stack)
        MPLS: Time to Live                  = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
```

```
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 18075
        IP: Flags           = 4X
        IP:       .1.. .... = don't fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = C44A (correct)
        IP: Source address      = [88.1.88.8]
        IP: Destination address = [192.168.1.1]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 0 (Echo reply)
        ICMP: Code = 0
        ICMP: Checksum = 9B2D (correct)
        ICMP: Identifier = 3046
        ICMP: Sequence number = 3245
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
        ICMP:
```

Echo Reply to Customer B VPN

Here, we see an echo reply going back to the destination IP address 192.168.1.1 in VRF custB. The destination address is translated to 172.31.1.1 by the ingress PE NAT function.

```
To VRF custB:
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 12 arrived at  09:15:49.6635; frame size is 118 (0076 hex) bytes.
        DLC:  Destination = Station 0090BF9C6C1C
        DLC:  Source      = Station 005054D92A25
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                    = 0001D
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                    = 1 (Bottom of Stack)
        MPLS: Time to Live                   = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 37925
```

```
        IP: Flags           = 4X
        IP:       .1.. .... = don't fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 75BF (correct)
        IP: Source address      = [88.1.88.8]
        IP: Destination address = [192.168.2.2]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 0 (Echo reply)
        ICMP: Code = 0
        ICMP: Checksum = B39A (correct)
        ICMP: Identifier = 4173
        ICMP: Sequence number = 4212
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

Note: In the return packets, the MPLS label values are included and differ: *001A* for VRF custA and *001D* for VRF custB.

Echo from Customer A VPN   Destination Is a Generic Interface

This next set of packets show the difference when the interface to the shared service LAN is a generic interface and not part of a VRF instance. Here, the configuration has been changed to use a common pool for both local VPNs with overlapping IP addresses.

```
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload
ip nat inside source list 181 pool SSPOOL1 vrf custB overload
```

```
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 1 arrived at  09:39:19.6580; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                        = 00019
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                        = 1 (Bottom of Stack)
        MPLS: Time to Live                       = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 55
        IP: Flags           = 0X
```

```
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 4AAF (correct)
        IP: Source address      = [192.168.1.1]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = 0905 (correct)
        ICMP: Identifier = 874
        ICMP: Sequence number = 3727
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

Echo from Customer B VPN   Destination Is a Generic Interface

Here, we see an echo request coming from source IP address 172.31.1.1 in VRF custB. The source address was translated to 192.168.1.3 (from common pool SSPOOL1) as specified by the NAT configuration:

```
ip nat pool SSPOOL1 192.168.1.1 192.168.1.254 prefix-length 24
ip nat inside source list 181 pool SSPOOL1 vrf custA overload
ip nat inside source list 181 pool SSPOOL1 vrf custB overload

DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 11 arrived at  09:39:26.4971; frame size is 118 (0076 hex)
            bytes.
        DLC:  Destination = Station 005054D92A25
        DLC:  Source      = Station 0090BF9C6C1C
        DLC:  Ethertype   = 8847 (MPLS)
        DLC:
  MPLS: ----- MPLS Label Stack  -----
        MPLS:
        MPLS: Label Value                   = 0001F
        MPLS: Reserved For Experimental Use = 0
        MPLS: Stack Value                   = 1 (Bottom of Stack)
        MPLS: Time to Live                  = 254 (hops)
        MPLS:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 75
        IP: Flags           = 0X
        IP:       .0.. .... = may fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 4A99 (correct)
```

```
        IP: Source address      = [192.168.1.3]
        IP: Destination address = [88.1.88.8]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 8 (Echo)
        ICMP: Code = 0
        ICMP: Checksum = 5783 (correct)
        ICMP: Identifier = 4237
        ICMP: Sequence number = 977
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

Note: When the interface at the egress PE is a generic interface (not a VRF instance), the labels imposed are different. In this case, *0x19* and *0x1F*.

Echo Reply to Customer A VPN   Destination Is a Generic Interface

Next, we see an echo reply going back to the destination IP address 192.168.1.1 in VRF custA. The destination address is translated to 172.31.1.1 by the ingress PE NAT function.

```
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 2 arrived at  09:39:19.6621; frame size is 114 (0072 hex)
            bytes.
        DLC:  Destination = Station 0090BF9C6C1C
        DLC:  Source      = Station 005054D92A25
        DLC:  Ethertype   = 0800 (IP)
        DLC:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit - transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit - no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 54387
        IP: Flags           = 4X
        IP:       .1.. .... = don't fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 3672 (correct)
        IP: Source address      = [88.1.88.8]
        IP: Destination address = [192.168.1.1]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 0 (Echo reply)
        ICMP: Code = 0
        ICMP: Checksum = 1105 (correct)
        ICMP: Identifier = 874
        ICMP: Sequence number = 3727
        ICMP: [72 bytes of data]
        ICMP:
```

```
        ICMP: [Normal end of "ICMP header".]
```

Echo Reply to Customer B VPN   Destination Is a Generic Interface

Here, we see an echo reply going back to the destination IP address 192.168.1.3 in VRF custB. The destination address is translated to 172.31.1.1 by the ingress PE NAT function.

```
DLC:  ----- DLC Header -----
        DLC:
        DLC:  Frame 12 arrived at  09:39:26.4978; frame size is 114 (0072 hex)
            bytes.
        DLC:  Destination = Station 0090BF9C6C1C
        DLC:  Source      = Station 005054D92A25
        DLC:  Ethertype   = 0800 (IP)
        DLC:
  IP: ----- IP Header -----
        IP:
        IP: Version = 4, header length = 20 bytes
        IP: Type of service = 00
        IP:       000. ....   = routine
        IP:       ...0 .... = normal delay
        IP:       .... 0... = normal throughput
        IP:       .... .0.. = normal reliability
        IP:       .... ..0. = ECT bit − transport protocol will ignore the CE
            bit
        IP:       .... ...0 = CE bit − no congestion
        IP: Total length    = 100 bytes
        IP: Identification  = 61227
        IP: Flags           = 4X
        IP:       .1.. .... = don't fragment
        IP:       ..0. .... = last fragment
        IP: Fragment offset = 0 bytes
        IP: Time to live    = 254 seconds/hops
        IP: Protocol        = 1 (ICMP)
        IP: Header checksum = 1BB8 (correct)
        IP: Source address      = [88.1.88.8]
        IP: Destination address = [192.168.1.3]
        IP: No options
        IP:
  ICMP: ----- ICMP header -----
        ICMP:
        ICMP: Type = 0 (Echo reply)
        ICMP: Code = 0
        ICMP: Checksum = 5F83 (correct)
        ICMP: Identifier = 4237
        ICMP: Sequence number = 977
        ICMP: [72 bytes of data]
        ICMP:
        ICMP: [Normal end of "ICMP header".]
```

Note: Since the replies are destined to a global address, no VRF labels are imposed.

With the exit interface to the shared service LAN segment defined as a generic interface, a common pool is permitted. The pings result in these NAT entries in router **gila**:

```
gila# show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.3:4237  172.31.1.1:4237    88.1.88.8:4237     88.1.88.8:4237
icmp 192.168.1.3:4238  172.31.1.1:4238    88.1.88.8:4238     88.1.88.8:4238
icmp 192.168.1.3:4239  172.31.1.1:4239    88.1.88.8:4239     88.1.88.8:4239
icmp 192.168.1.3:4240  172.31.1.1:4240    88.1.88.8:4240     88.1.88.8:4240
icmp 192.168.1.3:4241  172.31.1.1:4241    88.1.88.8:4241     88.1.88.8:4241
icmp 192.168.1.1:874   172.31.1.1:874     88.1.88.8:874      88.1.88.8:874
icmp 192.168.1.1:875   172.31.1.1:875     88.1.88.8:875      88.1.88.8:875
```

```
icmp 192.168.1.1:876   172.31.1.1:876     88.1.88.8:876      88.1.88.8:876
icmp 192.168.1.1:877   172.31.1.1:877     88.1.88.8:877      88.1.88.8:877
icmp 192.168.1.1:878   172.31.1.1:878     88.1.88.8:878      88.1.88.8:878
gila#

gila# show ip nat tr ver
Pro Inside global      Inside local       Outside local      Outside global
icmp 192.168.1.3:4237  172.31.1.1:4237    88.1.88.8:4237     88.1.88.8:4237
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:4238  172.31.1.1:4238    88.1.88.8:4238     88.1.88.8:4238
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:4239  172.31.1.1:4239    88.1.88.8:4239     88.1.88.8:4239
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:4240  172.31.1.1:4240    88.1.88.8:4240     88.1.88.8:4240
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.3:4241  172.31.1.1:4241    88.1.88.8:4241     88.1.88.8:4241
    create 00:00:08, use 00:00:08, left 00:00:51, Map-Id(In): 2,
    flags:
extended, use_count: 0, VRF : custB
icmp 192.168.1.1:874   172.31.1.1:874     88.1.88.8:874      88.1.88.8:874
    create 00:00:16, use 00:00:16, left 00:00:43, Map-Id(In): 3,
Pro Inside global      Inside local       Outside local      Outside global
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:875   172.31.1.1:875     88.1.88.8:875      88.1.88.8:875
    create 00:00:18, use 00:00:18, left 00:00:41, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:876   172.31.1.1:876     88.1.88.8:876      88.1.88.8:876
    create 00:00:18, use 00:00:18, left 00:00:41, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:877   172.31.1.1:877     88.1.88.8:877      88.1.88.8:877
    create 00:00:18, use 00:00:18, left 00:00:41, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA
icmp 192.168.1.1:878   172.31.1.1:878     88.1.88.8:878      88.1.88.8:878
    create 00:00:18, use 00:00:18, left 00:00:41, Map-Id(In): 3,
    flags:
extended, use_count: 0, VRF : custA

gila#
debug ip nat vrf

IP NAT VRF debugging is on
gila#
.Jan  2 09:34:54 EST: NAT-TAGSW(p) : Tag Pkt s=172.18.60.179, d=10.88.162.9, vrf=custA
.Jan  2 09:35:02 EST: NAT-TAGSW(p) : Tag Pkt s=172.18.60.179, d=10.88.162.13, vrf=custB
.Jan  2 09:35:12 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custA
.Jan  2 09:35:12 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:12 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custA
.Jan  2 09:35:12 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:12 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custA
.Jan  2 09:35:12 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:12 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custA
.Jan  2 09:35:12 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:12 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custA
.Jan  2 09:35:12 EST: NAT-ip2tag: Punting to process
```

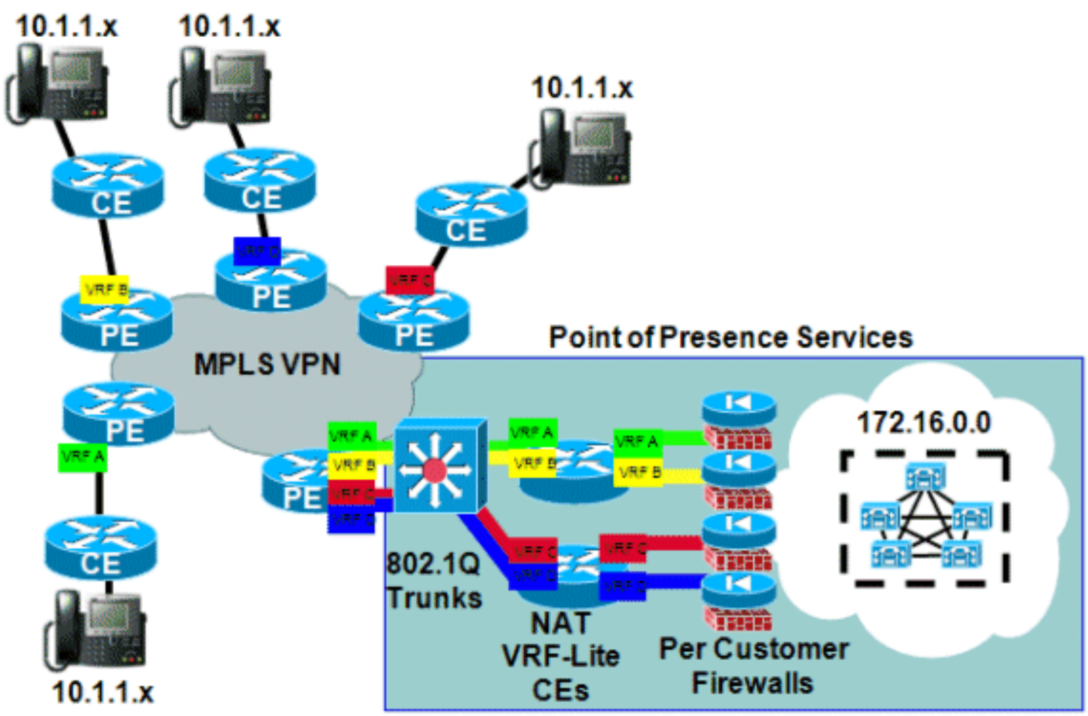
```
.Jan  2 09:35:19 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custB
.Jan  2 09:35:19 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:19 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custB
.Jan  2 09:35:19 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:19 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custB
.Jan  2 09:35:19 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:19 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custB
.Jan  2 09:35:19 EST: NAT-ip2tag: Punting to process
.Jan  2 09:35:19 EST: NAT-ip2tag : Tag Pkt s=172.31.1.1, d=88.1.88.8, vrf=custB
.Jan  2 09:35:19 EST: NAT-ip2tag: Punting to process
gila#
```

Service Example

An example of a shared virtual IP PBX service is shown in Figure 8. This illustrates a variant to the ingress and egress examples described earlier.

In this design, the shared VoIP service is front−ended by a set of routers that perform the NAT function. These routers have multiple VRF interfaces using a feature known as VRF−Lite. The traffic then flows to the shared Cisco CallManager cluster. Firewall services are also provided on a per−company basis. Inter−company calls must pass through the firewall, while intra−company calls are handled across the customer VPN using the company s internal addressing scheme.

Figure 8: Managed Virtual PBX Service Example



Availability

Cisco IOS NAT support for MPLS VPNs is available in Cisco IOS release 12.2(13)T and is available for all platforms that support MPLS and can run this early deployment release train.

Conclusion

Cisco IOS NAT has features to permit scalable deployment of shared services today. Cisco continues to develop NAT application level gateway (ALG) support for protocols important to customers. Performance improvements and hardware acceleration for translation functions will ensure that NAT and ALGs provide acceptable solutions for some time to come. All relevant standards activities and community actions are being monitored by Cisco. As other standards are developed, their use will be evaluated based on customer desires, requirements, and application.

Related Information

- **Cisco IOS NAT Application Layer Gateways**
- **MPLS and VPN Architectures**
- **Advanced MPLS Design and Implementation**
- **Technical Support & Documentation − Cisco Systems**

Updated: Aug 17, 2010 Document ID: 112084