Troubleshoot DMVPN Phase 2 Spoke-to-Spoke Tunnel

Contents

Introduction

This document describes how to troubleshoot a phase 2 spoke-to-spoke DMVPN tunnel when it does not establishes.

Prerequisites

Requirements

Cisco recommends that you have knowledge on the next topics:

- Dynamic Multipoint Virtual Private Network (DMVPN)
- IKE/IPSEC protocols
- Next Hop Resolution Protocol (NHRP)

Components Used

This document is based on this software version:

- Cisco CSR1000V (VXE) - Version 17.03.08

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

This document describes how to configure and use different troubleshooting tools on a common DMVPN issue. The issue is failed negotiation of a phase 2 DMVPN tunnel, where the source spoke, the DMVPN state shows UP with the correct Non-Broadcast Multi-Access (NBMA)/Tunnel mapping to the destination spoke. However, on the destination spoke an incorrect mapping is displayed.

Theoretical Background

It is important to understand how spoke-to-spoke tunnels are established when having a DMVPN Phase 2 set-up. This section provides a brief theoretical summary of the NHRP process during this phase.

In DMVPN Phase 2 you can build dynamic spoke-to-spoke tunnels on demand. This is possible because, on all devices within the DMVPN cloud (hub and spokes) the mode of the tunnel interface changes to Generic Routing Encapsulation (GRE) multipoint. One of the key features of this phase is that the hub is not perceived as the next-hop by the other devices. Instead, all spokes have the routing information of each other. When establishing a spoke-to-spoke tunnel in phase 2, an NHRP process gets triggered where the spokes learn the information about other spokes, and makes a mapping between the NBMA and tunnel IP addresses.

The next steps list how the NHRP resolution process is triggered:

1. When the source spoke tries to reach the LAN of the destination spoke, it does a route lookup triggering the resolution request message to get the NBMA address of the destination spoke. The source spoke send this initial message to the hub.

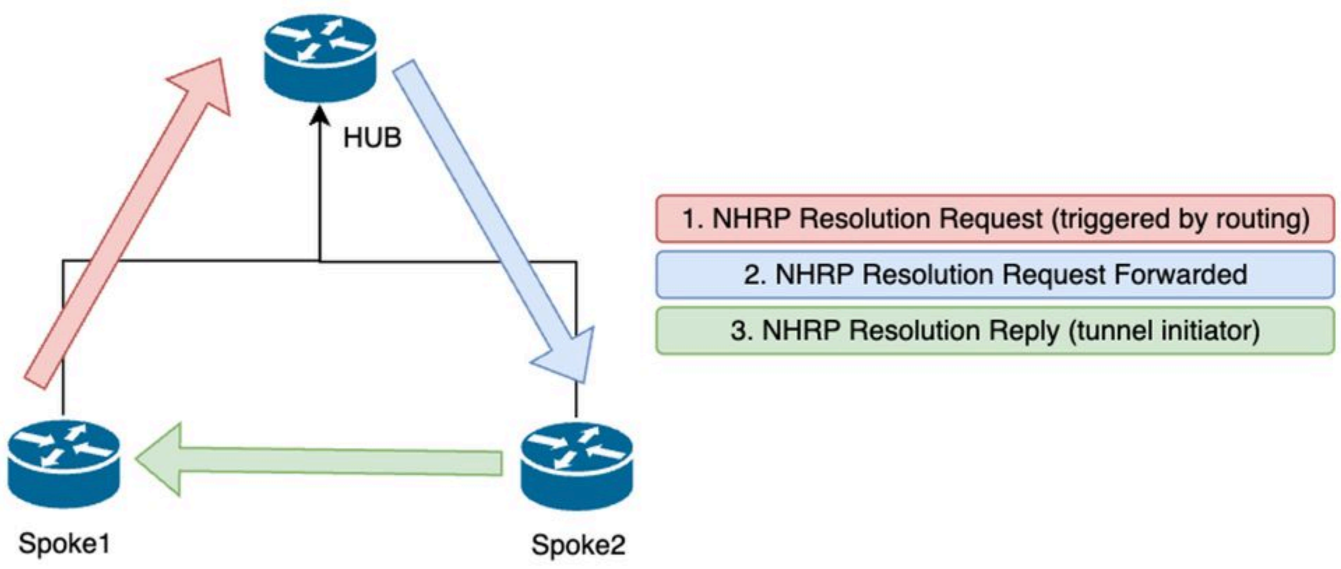
2. The hub receives the resolution request and forwards it to the destination spoke.

3. The destination spoke sends the resolution reply to the source spoke. If the tunnel configuration has an IPSEC profile linked:

   - The NHRP resolution process is delayed until IKE/IPSEC protocols can establish.

   - The destination spoke initiates and establish the IKE/IPSEC tunnels.

   - Then, the NHRP process is resumed and the destination spoke sends the resolution reply to the source spoke using the IPSEC tunnel as transport method.


HUB
1. NHRP Resolution Request (triggered by routing)
2. NHRP Resolution Request Forwarded
3. NHRP Resolution Reply (tunnel initiator)
Spoke1
Spoke2

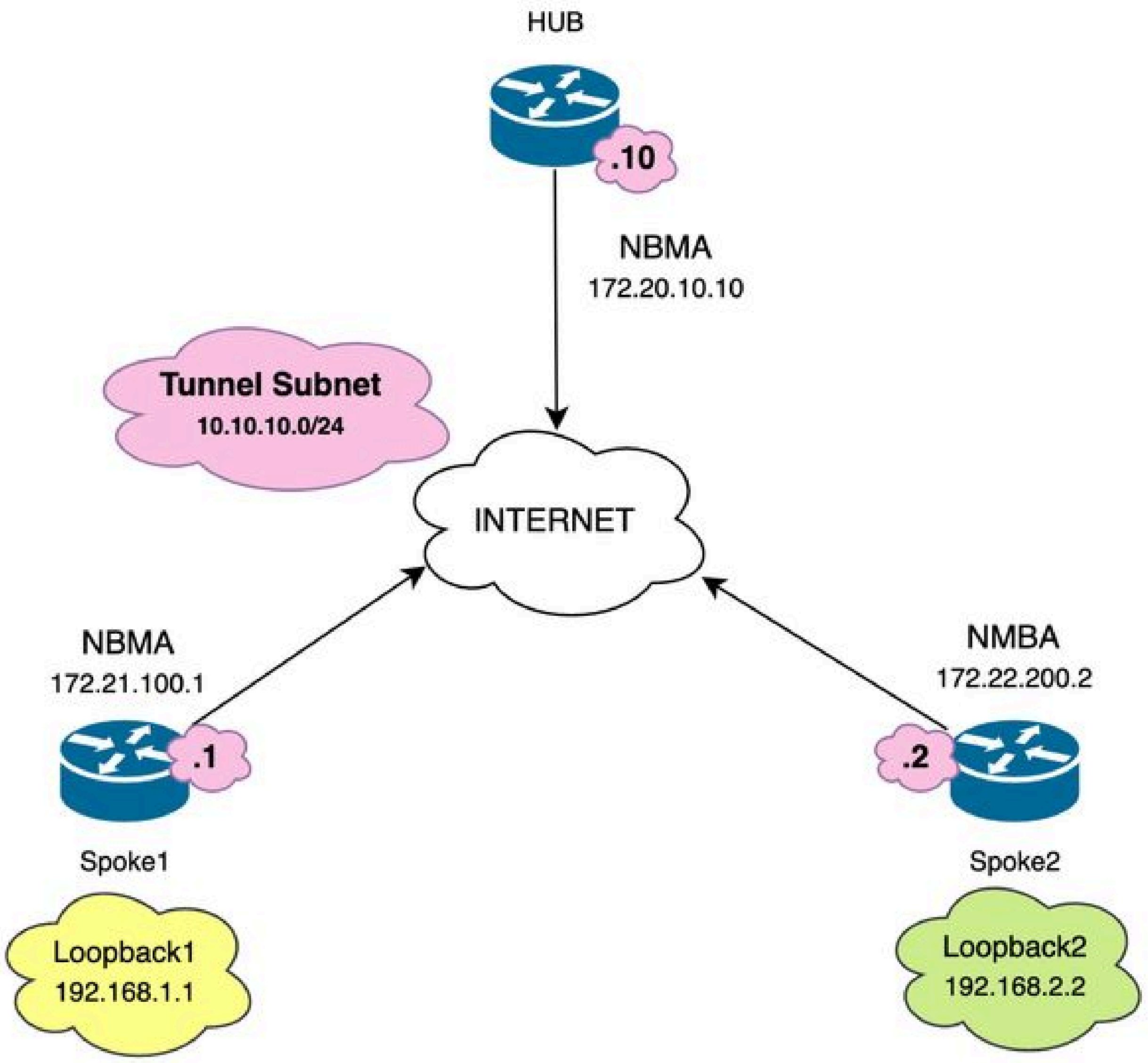
NHRP Message Flow Between the Spokes on Phase 2

Note: Before the resolution process can start, all the spokes must be already registered with the

HUB.

Topology

This diagram shows the topology used for the scenario:



Network Diagram and IP Subnets Used

Troubleshooting Steps

In this scenario the spoke-to-spoke tunnel between Spoke1 and Spoke2 is not established, affecting the communication between their local resources (represented by loopback interfaces) as they are not able to reach each other.

```
SPOKE1#ping 192.168.2.2 source loopback1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Initial Validation

When encountering such a scenario, it is important to begin by validating the tunnel configuration and ensure that both devices have the correct values within it. To review the tunnel configuration run the command **show running-config interface tunnel<ID>**.

Spoke 1 tunnel configuration:

<#root>

```
SPOKE1#show running-config interface tunnel10
Building configuration...

Current configuration : 341 bytes
!
interface Tunnel10
ip address 10.10.10.1 255.255.255.0
no ip redirects

ip nhrp authentication DMVPN



ip nhrp map 10.10.10.10 172.20.10.10



ip nhrp map multicast 172.20.10.10


ip nhrp network-id 10

ip nhrp nhs 10.10.10.10


tunnel source GigabitEthernet1

tunnel mode gre multipoint



tunnel protection IPSEC profile IPSEC_Profile_1


end
```

Spoke 2 tunnel configuration:

<#root>

```
SPOKE2#show running-config interface tunnel10
Building configuration...
```

```
Current configuration : 341 bytes
!
interface Tunnel10
ip address 10.10.10.2 255.255.255.0
no ip redirects
ip nhrp authentication DMVPN


ip nhrp map 10.10.10.10 172.20.10.10


ip nhrp map multicast 172.20.10.10

ip nhrp network-id 10
ip nhrp nhs 10.10.10.10

tunnel source GigabitEthernet1
tunnel mode gre multipoint


tunnel protection IPSEC profile IPSEC_Profile_1

end
```

On the configuration you need to validate that the mapping to the HUB is correct, the NHRP authentication string is matching between the devices, both spokes have the same DMVPN phase configured, and, if IPSEC protection is used, verify that the correct crypto configuration is applied.

If the configuration is correct and it includes IPSEC protection, it is necessary to verify that the IKE and IPSEC protocols are working correctly. This is because NHRP uses the IPSEC tunnel as transport method to fully negotiate. To verify the state of the IKE/IPSEC protocols run the command **show crypto IPSEC sa peer x.x.x.x** (where x.x.x.x is the NBMA IP address of the spoke you are trying to establish the tunnel with).

Note: To verify if the IPSEC tunnel is up, the inbound and outbound Encapsulation Security Payload (ESP) section must have the tunnel information (SPI, transform-set, and so on). All the values shown on this section must match on both ends.

Note: If any issues with IKE/IPSEC are identified, the troubleshooting must focus on those protocols.

IKE/IPSEC tunnel status on Spoke1:

<#root>

```
SPOKE1#
```

```
show crypto IPSEC sa peer 172.22.200.2
```

```
interface: Tunnel10
Crypto map tag: Tunnel10-head-0, local addr 172.21.100.1

protected vrf: (none)
local ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
remote ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
current_peer 172.22.200.2 port 500
PERMIT, flags={origin_is_acl,}
```

```
#pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0


#pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0


#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 172.21.100.1, remote crypto endpt.: 172.22.200.2
plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
current outbound spi: 0x6F6BF94A(1869347146)
PFS (Y/N): N, DH group: none


inbound esp sas:


spi: 0x84502A19(2219846169)


transform: esp-256-aes esp-sha256-hmac

 ,
in use settings ={Transport, }
conn id: 2049, flow_id: CSR:49, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28716)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:


outbound esp sas:


spi: 0x6F6BF94A(1869347146)


transform: esp-256-aes esp-sha256-hmac

 ,
in use settings ={Transport, }
conn id: 2050, flow_id: CSR:50, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28716)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:
```

IKE/IPSEC tunnel status on Spoke2:

<#root>

```
SPOKE2#

show crypto IPSEC sa peer 172.21.100.1



interface: Tunnel10
Crypto map tag: Tunnel10-head-0, local addr 172.22.200.2

protected vrf: (none)
local ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
remote ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
current_peer 172.21.100.1 port 500
PERMIT, flags={origin_is_acl,}

#pkts encaps: 16, #pkts encrypt: 16, #pkts digest: 16



#pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0


#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 172.22.200.2, remote crypto endpt.: 172.21.100.1
plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
current outbound spi: 0x84502A19(2219846169)
PFS (Y/N): N, DH group: none


inbound esp sas:



spi: 0x6F6BF94A(1869347146)



transform: esp-256-aes esp-sha256-hmac ,


in use settings ={Transport, }
conn id: 2045, flow_id: CSR:45, sibling_flags FFFFFFFF80004008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28523)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:


outbound esp sas:



spi: 0x84502A19(2219846169)
```

```
transform: esp-256-aes esp-sha256-hmac

 ,
in use settings ={Transport, }
conn id: 2046, flow_id: CSR:46, sibling_flags FFFFFFFF80004008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4607998/28523)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:
```

The outputs show that on both spokes the IPSEC tunnel is up, but, Spoke2 shows encrypted packets (encaps) but no decrypted packets (decaps). Meanwhile, Spoke1 does not show any packets flowing through the IPSEC tunnel. This indicates that the problem can be on the NHRP protocol.

Troubleshooting Tools

After doing the initial validation and corroborating the configuration and the IKE/IPSEC protocols (if needed) are not causing the communication problem, you can use the tools presented on this section to continue the troubleshooting.

Useful Commands

The command **show dmvpn interface tunnel<ID>** give you DMVPN-specific session information (NBMA/Tunnel IP addresses, state of the tunnel, up/down time and attribute). You can use the **detail** keyword to show details from the crypto session/socket. It is important to mention that the state of the tunnel must match on both ends.

Spoke 1 **show dmvpn interface tunnel<ID>** output:

<#root>

```
SPOKE1#

show dmvpn interface tunnel10


Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
N - NATed, L - Local, X - No Socket
T1 - Route Installed, T2 - Nexthop-override, B - BGP
C - CTS Capable, I2 - Temporary
# Ent --> Number of NHRP entries with same NBMA peer
NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:1,

# Ent Peer NBMA Addr Peer Tunnel Add  State UpDn Tm  Attrb
----- --------------- --------------- ----- -------- -----
   2

172.20.10.10     10.10.10.2      UP  00:00:51  I2
```

```
                        10.10.10.10     UP  02:53:27  S
```

Spoke 2 **show dmvpn interface tunnel<ID>** output:

<#root>

```
SPOKE2#

show dmvpn interface tunnel10


Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
N - NATed, L - Local, X - No Socket
T1 - Route Installed, T2 - Nexthop-override, B - BGP
C - CTS Capable, I2 - Temporary
# Ent --> Number of NHRP entries with same NBMA peer
NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

# Ent Peer NBMA Addr Peer Tunnel Add State UpDn Tm Attrb
----- --------------- --------------- ----- -------- -----


1    172.21.100.1     10.10.10.1     UP  00:03:53   D


  1    172.20.10.10     10.10.10.10    UP  02:59:14   S
```

The output on each device shows different information for each spoke. In the Spoke1 table, you can see that the entry for Spoke 2 does not include the correct NBMA IP address, and the attribute appears incomplete (I2). On the other hand, the Spoke2 table shows the correct mapping (NBMA/Tunnel IP addresses) and the state as **up** indicating that the tunnel is fully negotiated.

The next commands can be helpful during the troubleshooting process:

- **show ip nhrp**: Display NHRP mapping information
- **show ip nhrp traffic interface tunnel10**: Displays NHRP traffic statistics

Note: For command specifications (syntax, description, keywords, example), please consult the Command Reference: Cisco IOS Security Command Reference: Commands S to Z

Debugs

After verifying the previous information and confirmed that the tunnel is experiencing negotiation issues, it is necessary to enable debugs to observe how the NHRP packets are being exchanged. The next debugs must be enabled on all the involved devices:

1. **debug dmvpn condition peer NBMA** x.x.x.x (where x.x.x.x is the remote device IP address).
2. **debug dmvpn all all**: this command enables ISAKMP, IKEv2, IPSEC, DMVPN and NHRP debugging commands.

Tip: It is recommended to use the peer condition command every time you enable the debugs so you can see the negotiation of that specific tunnel.

To see the complete NHRP flow, the next debugs commands were used on each device:

Spoke1

```
debug dmvpn condition peer NBMA 172.22.200.2
debug dmvpn condition peer NBMA 172.20.10.10
debug dmvpn all all
```

HUB

```
debug dmvpn condition peer NBMA 172.21.100.1
debug dmvpn condition peer NBMA 172.22.200.2
debug dmvpn all all
```

Spoke2

```
debug dmvpn condition peer NBMA 172.21.100.1
debug dmvpn condition peer NBMA 172.20.10.10
debug dmvpn all all
```

Note: The debugs must be enabled and collected simultaneously on all the devices involved.

Debugs enabled on all the devices are displayed with the command **show debug**:

<#root>

```
ROUTER#

show debug


IOSXE Conditional Debug Configs:

Conditional Debug Global State: Stop


IOSXE Packet Tracing Configs:


Packet Infra debugs:

Ip Address Port
------------------------------------------------------|----------

NHRP:
NHRP protocol debugging is on
NHRP activity debugging is on
NHRP detail debugging is on
NHRP extension processing debugging is on
NHRP cache operations debugging is on
NHRP routing debugging is on
NHRP rate limiting debugging is on
NHRP errors debugging is on
NHRP events debugging is on

Cryptographic Subsystem:
Crypto ISAKMP debugging is on
Crypto ISAKMP Error debugging is on
Crypto IPSEC debugging is on
Crypto IPSEC Error debugging is on
Crypto secure socket events debugging is on
IKEV2:
IKEv2 error debugging is on
IKEv2 default debugging is on
IKEv2 packet debugging is on
IKEv2 packet hexdump debugging is on
IKEv2 internal debugging is on
Tunnel Protection Debugs:
Generic Tunnel Protection debugging is on

DMVPN:
DMVPN error debugging is on
DMVPN UP/DOWN event debugging is on
DMVPN detail debugging is on
DMVPN packet debugging is on
DMVPN all level debugging is on
```

After collecting all the debugs, you must start analyzing the debugs on the source spoke (Spoke1), this allows you to trace the negotiation from the beginning.

Spoke1 debug output:

<#root>

```
-------------------- [IKE/IPSEC DEBUG OUTPUTS OMITTED]--------------------

*Feb 1 01:31:34.657: ISAKMP: (1016):
```

Old State = IKE_QM_R_QM2 New State = IKE_QM_PHASE2_COMPLETE

```
*Feb 1 01:31:34.657: IPSEC(key_engine): got a queue event with 1 KMI message(s)
*Feb 1 01:31:34.657: IPSEC(key_engine_enable_outbound): rec'd enable notify from ISAKMP
*Feb 1 01:31:34.657: CRYPTO_SS(TUNNEL SEC): Sending MTU Changed message
*Feb 1 01:31:34.661: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): Got MTU message mtu 1458
*Feb 1 01:31:34.661: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): connection lookup returned 80007F2
*Feb 1 01:31:34.662: CRYPTO_SS(TUNNEL SEC): Sending Socket Up message
*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): connection lookup returned 80007F2
*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2):
```

tunnel_protection_socket_up

```
*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): Signalling NHRP
*Feb 1 01:31:36.428: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0
*Feb 1 01:31:36.429: NHRP: No delayed event node found.
*Feb 1 01:31:36.429: NHRP: There is no VPE Extension to construct for the request
*Feb 1 01:31:36.429: NHRP: Sending NHRP Resolution Request for dest: 10.10.10.2 to nexthop: 10.10.10.2
*Feb 1 01:31:36.429: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:36.429: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:36.429: NHRP:
```

Send Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85

```
*Feb 1 01:31:36.429: src: 10.10.10.1, dst: 10.10.10.2
*Feb 1 01:31:36.429: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:36.429: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:36.429: pktsz: 85 extoff: 52
*Feb 1 01:31:36.429: (M) flags: "router auth src-stable nat ",
```

reqid: 10

```
*Feb 1 01:31:36.429:
```

src NBMA: 172.21.100.1

```
*Feb 1 01:31:36.429:
```

src protocol: 10.10.10.1, dst protocol: 10.10.10.2

```
*Feb 1 01:31:36.429: (C-1) code: no error(0), flags: none
*Feb 1 01:31:36.429: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:36.429: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:36.429: Responder Address Extension(3):
*Feb 1 01:31:36.429: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:36.429: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:36.429: Authentication Extension(7):
*Feb 1 01:31:36.429: type:Cleartext(1),
```

data:DMVPN

```
*Feb 1 01:31:36.429: NAT address Extension(9):
*Feb 1 01:31:36.430: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.20.10
*Feb 1 01:31:36.430: NHRP: 109 bytes out Tunnel10
*Feb 1 01:31:36.430: NHRP-RATE:
```

```
Retransmitting Resolution Request for 10.10.10.2, reqid 10, (retrans ivl 4 sec)


*Feb 1 01:31:39.816: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0
*Feb 1 01:31:39.816: NHRP: No delayed event node found.
*Feb 1 01:31:39.816: NHRP: There is no VPE Extension to construct for the request
*Feb 1 01:31:39.817: NHRP: Sending NHRP Resolution Request for dest: 10.10.10.2 to nexthop: 10.10.10.2
*Feb 1 01:31:39.817: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:39.817: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:39.817: NHRP:

Send Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85


*Feb 1 01:31:39.817: src: 10.10.10.1, dst: 10.10.10.2
*Feb 1 01:31:39.817: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:39.817: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:39.817: pktsz: 85 extoff: 52
*Feb 1 01:31:39.817: (M) flags: "router auth src-stable nat ",

reqid: 10


*Feb 1 01:31:39.817:

src NBMA: 172.21.100.1


*Feb 1 01:31:39.817:

src protocol: 10.10.10.1, dst protocol: 10.10.10.2


*Feb 1 01:31:39.817: (C-1) code: no error(0), flags: none
*Feb 1 01:31:39.817: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:39.817: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:39.817: Responder Address Extension(3):
*Feb 1 01:31:39.817: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:39.817: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:39.817: Authentication Extension(7):
*Feb 1 01:31:39.817: type:Cleartext(1),

data:DMVPN


*Feb 1 01:31:39.817: NAT address Extension(9):
*Feb 1 01:31:39.817: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.20.10
*Feb 1 01:31:39.818: NHRP: 109 bytes out Tunnel10
*Feb 1 01:31:39.818: NHRP-RATE:

Retransmitting Resolution Request for 10.10.10.2, reqid 10, (retrans ivl 8 sec)


*Feb 1 01:31:46.039: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0
*Feb 1 01:31:46.040: NHRP: No delayed event node found.
*Feb 1 01:31:46.040: NHRP: There is no VPE Extension to construct for the request
```

Once the Spoke1 NHRP process begins, the logs show that the device is sending the NHRP resolution request. The packet has some important information like the **src NMBA** and **src protocol** that are the NBMA IP address and tunnel IP address of the source spoke (Spoke1). You can also see the **dst protocol** value that has the tunnel IP address of the destination spoke (Spoke2). This is indicating that Spoke1 is asking for the NBMA address of Spoke2 to complete the mapping. Also on the packet, you can find the

reqid value that can help you track the packet along the path. This value is going to remain the same through the whole process, and can be helpful to track an specific flow of the NHRP negotiation. The packet has another values that are important to the negotiation like the NHRP authentication string.

After the device sends the NHRP resolution request, the logs show that a retransmission is sent. This is because the device is not seeing the NHRP resolution response so it sends the packet again. As Spoke1 is not seeing the response, it is necessary to track that packet on the next device in the path, meaning the HUB.

HUB debug output:

<#root>

```
*Feb 1 01:31:34.262:

NHRP: Receive Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85


*Feb 1 01:31:34.262: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:34.262: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.263: pktsz: 85 extoff: 52
*Feb 1 01:31:34.263: (M) flags: "router auth src-stable nat ",

reqid: 10


*Feb 1 01:31:34.263:

src NBMA: 172.21.100.1


*Feb 1 01:31:34.263:

src protocol: 10.10.10.1, dst protocol: 10.10.10.2


*Feb 1 01:31:34.263: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.263: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.263: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:34.263: Responder Address Extension(3):
*Feb 1 01:31:34.263: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:34.263: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.263: Authentication Extension(7):
*Feb 1 01:31:34.263: type:Cleartext(1), data:DMVPN
*Feb 1 01:31:34.263: NAT address Extension(9):
*Feb 1 01:31:34.263: NHRP-DETAIL: netid_in = 10, to_us = 0
*Feb 1 01:31:34.263: NHRP-DETAIL:

Resolution request for afn 1 received on interface Tunnel10

 , for vrf: global(0x0) label: 0
*Feb 1 01:31:34.263: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.263: NHRP:

Route lookup for destination 10.10.10.2

 in vrf: global(0x0) yielded interface Tunnel10, prefixlen 24
*Feb 1 01:31:34.263: NHRP-DETAIL: netid_out 10, netid_in 10
*Feb 1 01:31:34.263: NHRP: Forwarding request due to authoritative request.
*Feb 1 01:31:34.263: NHRP-ATTR:

NHRP Resolution Request packet is forwarded to 10.10.10.2 using vrf: global(0x0)
```

```
*Feb 1 01:31:34.263: NHRP: Attempting to forward to destination: 10.10.10.2 vrf: global(0x0)
*Feb 1 01:31:34.264: NHRP: Forwarding: NHRP SAS picked source: 10.10.10.10 for destination: 10.10.10.2
*Feb 1 01:31:34.264: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:34.264: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:34.264: NHRP:
```

Forwarding Resolution Request via Tunnel10 vrf: global(0x0), packet size: 105

```
*Feb 1 01:31:34.264: src: 10.10.10.10, dst: 10.10.10.2
*Feb 1 01:31:34.264: (F) afn: AF_IP(1), type: IP(800), hop: 254, ver: 1
*Feb 1 01:31:34.264: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.264: pktsz: 105 extoff: 52
*Feb 1 01:31:34.264: (M) flags: "router auth src-stable nat ",
```

reqid: 10

```
*Feb 1 01:31:34.264:
```

src NBMA: 172.21.100.1

```
*Feb 1 01:31:34.264:
```

src protocol: 10.10.10.1, dst protocol: 10.10.10.2

```
*Feb 1 01:31:34.264: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.264: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.264: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:34.264: Responder Address Extension(3):
*Feb 1 01:31:34.264: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:34.264: (C-1)
```

code: no error(0)

```
, flags: none
*Feb 1 01:31:34.264: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.264: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.264:
```

client NBMA: 172.20.10.10

```
*Feb 1 01:31:34.264:
```

client protocol: 10.10.10.10

```
*Feb 1 01:31:34.264: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.264: Authentication Extension(7):
*Feb 1 01:31:34.264: type:Cleartext(1),
```

data:DMVPN

```
*Feb 1 01:31:34.265: NAT address Extension(9):
*Feb 1 01:31:34.265: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.22.200
*Feb 1 01:31:34.265: NHRP: 129 bytes out Tunnel10
```

Using the value of the **reqid**, you can observe that the HUB receives the resolution request sent by Spoke1. In the packet, the values of **src NBMA** and **src protocol** are the information from Spoke1, and the value of **dst protocol** is the tunnel IP of Spoke2, as it was seen on the debugs of Spoke1. When the HUB receives the resolution request, it performs a route lookup and forwards the packet to Spoke2. In the forwarded packet, the HUB adds an extension containing its own information (NBMA IP address and tunnel IP address).

The previous debugs show that the HUB is correctly forwarding the resolution request to spoke 2. Therefore, the next step is to confirm Spoke2 is receiving it, processing it correctly, and sending to Spoke1 the resolution reply.

Spoke2 debug output:

<#root>

```
-------------------- [IKE/IPSEC DEBUG OUTPUTS OMITTED]--------------------

*Feb 1 01:31:34.647: ISAKMP: (1015):

Old State = IKE_QM_IPSEC_INSTALL_AWAIT New State = IKE_QM_PHASE2_COMPLETE


*Feb 1 01:31:34.647: NHRP: Process delayed resolution request src:10.10.10.1 dst:10.10.10.2 vrf: global
*Feb 1 01:31:34.648: NHRP-DETAIL: Resolution request for afn 1 received on interface Tunnel10 , for vrf
*Feb 1 01:31:34.648: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.648: NHRP:

Route lookup for destination 10.10.10.2 in vrf: global(0x0) yielded interface Tunnel10, prefixlen 24


*Feb 1 01:31:34.648: NHRP-ATTR: smart spoke feature and attributes are not configured
*Feb 1 01:31:34.648:

NHRP:



Request was to us. Process the NHRP Resolution Request.


*Feb 1 01:31:34.648: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.648: NHRP: nhrp_rtlookup for 10.10.10.2 in vrf: global(0x0) yielded interface Tunnel10,
*Feb 1 01:31:34.648: NHRP: Request was to us, responding with ouraddress
*Feb 1 01:31:34.648: NHRP: Checking for delayed event 10.10.10.1/10.10.10.2 on list (Tunnel10 vrf: glob
*Feb 1 01:31:34.648: NHRP: No delayed event node found.
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10: Checking to see if we need to delay for src 172.22.200.2 dst
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10: crypto_ss_listen_start already listening
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Opening a socket with profile IPSE
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Socket is already open. Ignoring.
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): tunnel is already open!
*Feb 1 01:31:34.648: NHRP: No need to delay processing of resolution event NBMA src:172.22.200.2 NBMA d
*Feb 1 01:31:34.648: NHRP-MEF: No vendor private extension in NHRP packet
*Feb 1 01:31:34.649: NHRP-CACHE: Tunnel10: Cache update for target 10.10.10.1/32 vrf: global(0x0) label
*Feb 1 01:31:34.649: 172.21.100.1 (flags:0x2080)
*Feb 1 01:31:34.649: NHRP:

Adding Tunnel Endpoints (VPN: 10.10.10.1, NBMA: 172.21.100.1)


*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10: crypto_ss_listen_start already listening
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Opening a socket with profile IPSE
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Found an existing tunnel endpoint
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): tunnel_protection_stop_pending_tim
```

```
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Socket is already open. Ignoring.
*Feb 1 01:31:34.653:
```

NHRP: Successfully attached NHRP subblock for Tunnel Endpoints (VPN: 10.10.10.1, NBMA: 172.21.100.1)

```
*Feb 1 01:31:34.653: NHRP: Peer capability:0
*Feb 1 01:31:34.653: NHRP-CACHE: Inserted subblock node(1 now) for cache: Target 10.10.10.1/32 nhop 10.1
*Feb 1 01:31:34.653: NHRP-CACHE: Converted internal dynamic cache entry for 10.10.10.1/32 interface Tunn
*Feb 1 01:31:34.653: NHRP-EVE: NHP-UP: 10.10.10.1, NBMA: 172.21.100.1
*Feb 1 01:31:34.653: NHRP-MEF: No vendor private extension in NHRP packet
*Feb 1 01:31:34.653: NHRP-CACHE: Tunnel10: Internal Cache add for target 10.10.10.2/32 vrf: global(0x0)
*Feb 1 01:31:34.653: 172.22.200.2 (flags:0x20)
*Feb 1 01:31:34.653: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.1
*Feb 1 01:31:34.654: NHRP-DETAIL: First hop route lookup for 10.10.10.1 yielded 10.10.10.1, Tunnel10
*Feb 1 01:31:34.654:
```

NHRP: Send Resolution Reply via Tunnel10 vrf: global(0x0), packet size: 133

```
*Feb 1 01:31:34.654: src: 10.10.10.2, dst: 10.10.10.1
*Feb 1 01:31:34.654: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:34.654: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.654: pktsz: 133 extoff: 60
*Feb 1 01:31:34.654: (M) flags: "router auth dst-stable unique src-stable nat ",
```

reqid: 10

```
*Feb 1 01:31:34.654:
```

src NBMA: 172.21.100.1

```
*Feb 1 01:31:34.654:
```

src protocol: 10.10.10.1, dst protocol: 10.10.10.2

```
*Feb 1 01:31:34.654: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 32, mtu: 9976, hd_time: 599
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
```

client NBMA: 172.22.200.2

```
*Feb 1 01:31:34.654:
```

client protocol: 10.10.10.2

```
*Feb 1 01:31:34.654: Responder Address Extension(3):
*Feb 1 01:31:34.654: (C) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
```

client NBMA: 172.22.200.2

```
*Feb 1 01:31:34.654:
```

client protocol: 10.10.10.2

```
*Feb 1 01:31:34.654: Forward Transit NHS Record Extension(4):
```

```
*Feb 1 01:31:34.654: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
```

`client NBMA: 172.20.10.10`

```
*Feb 1 01:31:34.654:
```

`client protocol: 10.10.10.10`

```
*Feb 1 01:31:34.654: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.654: Authentication Extension(7):
*Feb 1 01:31:34.654: type:Cleartext(1),
```

`data:DMVPN`

```
*Feb 1 01:31:34.655: NAT address Extension(9):
*Feb 1 01:31:34.655: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.21.10
*Feb 1 01:31:34.655: NHRP: 157 bytes out Tunnel10
*Feb 1 01:31:34.655: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.655: NHRP-DETAIL: Deleted delayed event on interfaceTunnel10 dest: 172.21.100.1
```

The **reqid** matches the value seen in the previous outputs, with this, it is confirmed that the NHRP resolution request packet sent by Spoke1 reaches Spoke2. This packet triggers a route lookup on Spoke2, and realizes that the resolution request is for itself, therefore, Spoke2 adds the information from Spoke1 to its NHRP table. Before sending the resolution reply packet back to Spoke1, the device adds its own information (NBMA IP address and tunnel IP address) so that Spoke1 can use that packet to add that information to its database.

Based on all the debugs seen, the NHRP Resolution Reply send from Spoke2 is not arriving to Spoke1. The HUB can be discarded from the issue as it is receiving and forwarding the NHRP Resolution Request packet as expected. Therefore, the next step is to take captures between Spoke1 and Spoke2 to get more details about the issue.

Embedded Packet Capture

The embedded packet capture feature allows you to analyze the traffic passing through the device. The first step to configure it is to create an access-list including the traffic you want to capture on both traffic flows (inbound and outbound).

For this scenario, the NBMA IP addresses are used:

```
ip access-list extended filter
10 permit ip host 172.21.100.1 host 172.22.200.2
20 permit ip host 172.22.200.2 host 172.21.100.1
```

Then, configure the capture using the command **monitor capture <CAPTURE_NAME> access-list <ACL_NAME> buffer size 10 interface <WAN_INTERFACE> both** and start the capture with the command **monitor capture <CAPTURE_NAME> start**.

Capture configuration on Spoke1 and Spoke2:

```
monitor capture CAP access-list filter buffer size 10 interface GigabitEthernet1 both
monitor capture CAP start
```

To display the output of the capture use the command **show monitor capture <CAPTURE_NAME> buffer brief**.

Capture output Spoke1:

<#root>

```
SPOKE1#show monitor capture CAP buffer brief
-----------------------------------------------------------------------------
#   size   timestamp     source             destination      dscp    protocol
-----------------------------------------------------------------------------
   0  210    0.000000   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   1  150    0.014999   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   2  478    0.028990   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   3  498    0.049985   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   4  150    0.069988   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   5  134    0.072994   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   6  230    0.074993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   7  230    0.089992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   8  118    0.100993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
```

9 218   0.108988   172.22.200.2   -> 172.21.100.1   48 CS6   ESP

10   70   0.108988   172.21.100.1   -> 172.22.200.2   0 BE   ICMP

11 218   1.907994   172.22.200.2   -> 172.21.100.1   48 CS6   ESP

12   70   1.907994   172.21.100.1   -> 172.22.200.2   0 BE   ICMP

13 218   5.818003   172.22.200.2   -> 172.21.100.1   48 CS6   ESP

14   70   5.818003   172.21.100.1   -> 172.22.200.2   0 BE   ICMP

15 218   12.559969   172.22.200.2   -> 172.21.100.1   48 CS6   ESP

16   70   12.559969   172.21.100.1   -> 172.22.200.2   0 BE   ICMP

17 218   26.859001   172.22.200.2   -> 172.21.100.1   48 CS6   ESP

```
  18   70   26.859001   172.21.100.1    ->  172.22.200.2     0  BE   ICMP


  19  218   54.378978   172.22.200.2    ->  172.21.100.1     48 CS6  ESP


  20   70   54.378978   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
```

Capture output Spoke2:

<#root>

```
SPOKE2#show monitor capture CAP buffer brief
----------------------------------------------------------------------------
#   size   timestamp     source             destination      dscp    protocol
----------------------------------------------------------------------------
   0  210     0.000000   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   1  150     0.015990   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   2  478     0.027998   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   3  498     0.050992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   4  150     0.069988   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   5  134     0.072994   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   6  230     0.074993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   7  230     0.089992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   8  118     0.099986   172.22.200.2    ->  172.21.100.1     48 CS6  UDP



9  218    0.108988   172.22.200.2    ->  172.21.100.1     48 CS6  ESP


  10   70    0.108988   172.21.100.1    ->  172.22.200.2     0  BE   ICMP


  11  218    1.907994   172.22.200.2    ->  172.21.100.1     48 CS6  ESP


  12   70    1.909001   172.21.100.1    ->  172.22.200.2     0  BE   ICMP


  13  218    5.817011   172.22.200.2    ->  172.21.100.1     48 CS6  ESP


  14   70    5.818002   172.21.100.1    ->  172.22.200.2     0  BE   ICMP


  15  218   12.559968   172.22.200.2    ->  172.21.100.1     48 CS6  ESP


  16   70   12.560960   172.21.100.1    ->  172.22.200.2     0  BE   ICMP


  17  218   26.858009   172.22.200.2    ->  172.21.100.1     48 CS6  ESP
```

```
 18   70   26.859001   172.21.100.1    ->  172.22.200.2     0  BE   ICMP

 19  218   54.378978   172.22.200.2    ->  172.21.100.1     48 CS6  ESP

 20   70   54.379970   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
```

The output of the captures shows that the initial packets are UDP traffic, indicating the IKE/IPSEC negotiation. After that, Spoke2 sends the resolution reply to Spoke1, which is seen as ESP traffic (packet 9). After this, the expected traffic flow is ESP, however, the next packet seen is ICMP traffic coming from Spoke1 to Spoke2.

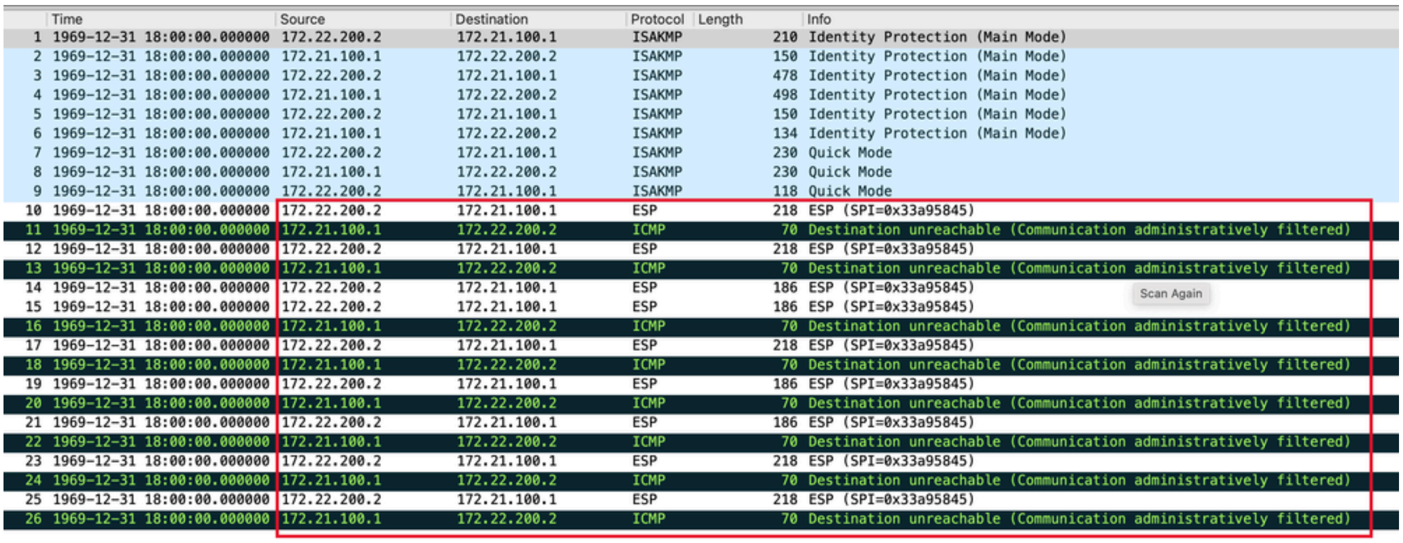
To analyze deeper the packet, you can export the pcap file from the device by running the command **show monitor capture <CAPTURE_NAME> buffer dump**. Then use a decoder tool to convert the dump output to a pcap file so you can open it with Wireshark.

Note: Cisco has a packet analyzer where you can find capture configuration, examples and a

---

decoder: Cisco TAC Tool - Packet Capture Config Generator and Analyzer

Wireshark output:



```
    Time                          Source          Destination     Protocol  Length  Info
 1  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ISAKMP       210  Identity Protection (Main Mode)
 2  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ISAKMP       150  Identity Protection (Main Mode)
 3  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ISAKMP       478  Identity Protection (Main Mode)
 4  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ISAKMP       498  Identity Protection (Main Mode)
 5  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ISAKMP       150  Identity Protection (Main Mode)
 6  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ISAKMP       134  Identity Protection (Main Mode)
 7  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ISAKMP       230  Quick Mode
 8  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ISAKMP       230  Quick Mode
 9  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ISAKMP       118  Quick Mode
10  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          218  ESP (SPI=0x33a95845)
11  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
12  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          218  ESP (SPI=0x33a95845)
13  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
14  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          186  ESP (SPI=0x33a95845)
15  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          186  ESP (SPI=0x33a95845)
16  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
17  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          218  ESP (SPI=0x33a95845)
18  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
19  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          186  ESP (SPI=0x33a95845)
20  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
21  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          186  ESP (SPI=0x33a95845)
22  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
23  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          218  ESP (SPI=0x33a95845)
24  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
25  1969-12-31 18:00:00.000000    172.22.200.2    172.21.100.1    ESP          218  ESP (SPI=0x33a95845)
26  1969-12-31 18:00:00.000000    172.21.100.1    172.22.200.2    ICMP          70  Destination unreachable (Communication administratively filtered)
```

Capture Output on Wireshark

The content of the ICMP packet has the error message **Destination unreachable (Communication administratively filtered)**. This indicates that there is some kind of filter, such as a router ACL or firewall affecting the traffic along the path. Most the of times, the filter is configured on the device sending the packet (in this case, Spoke1) but middle devices can send it as well.

Note: The Wireshark output is the same on both spokes.

Cisco IOS® XE Datapath Packet Trace Feature

The Cisco IOS XE datapath packet trace feature is used to analyze how the device is processing the traffic. To configure it, you need to create an access-list including the traffic you want to capture on both traffic flows (inbound and outbound).

For this scenario, the NBMA IP addresses are used.

```
ip access-list extended filter
10 permit ip host 172.21.100.1 host 172.22.200.2
20 permit ip host 172.22.200.2 host 172.21.100.1
```

Then, configure the fia-trace feature and set the debugging condition to use the access-list. Finally, start the condition.

```
debug platform packet-trace packet 1024 fia-trace
debug platform condition ipv4 access-list filter both
debug platform condition start
```

- **debug platform packet-trace packet <count> fia-trace:** enables detailed fia trace, stopping it once the amount of packets configured have been captured
- **debug platform condition ipv4 access-list <ACL-NAME> both:** sets a condition on the device using the access-list previously configured
- **debug platform condition start:** starts the condition

To review the output of the fia-trace use the next commands.

```
show platform packet-trace statistics
show platform packet-trace summary
show platform packet-trace packet <number>
```

Spoke1 **show platform packet-trace statistics** output:

<#root>

```
SPOKE1#show platform packet-trace statistics
Packets Summary
  Matched  18
  Traced   18
Packets Received
  Ingress  11
  Inject   7
    Count       Code  Cause
    4           2     QFP destination lookup
    3           9     QFP ICMP generated packet
Packets Processed
  Forward  7
  Punt     8
    Count       Code  Cause
    5           11    For-us data
    3           26    QFP ICMP generated packet

  Drop     3


    Count       Code  Cause


    3           8     Ipv4Acl


  Consume  0

          PKT_DIR_IN
             Dropped       Consumed       Forwarded
INFRA           0              0              0
TCP             0              0              0
UDP             0              0              5
```

```
IP                 0              0              5
IPV6               0              0              0
ARP                0              0              0

         PKT_DIR_OUT
             Dropped        Consumed        Forwarded
INFRA              0              0              0
TCP                0              0              0
UDP                0              0              0
IP                 0              0              0
IPV6               0              0              0
ARP                0              0              0
```

In the **show platform packet-trace statistics** output, you can see the counters for the packets processed by the device. This allows you to see the inbound and outbound packets, and check if the device is dropping any packets, along with the drop reason.

In the output shown, Spoke1 is dropping some packets with the description **Ipv4Acl**. To further analyze those packets, the command **show platform packet-trace summary** can be used.

Spoke1 **show platform packet-trace summary** output:

<#root>

```
SPOKE1#show platform packet-trace summary
Pkt   Input                      Output                     State  Reason
0     Gi1                        internal0/0/rp:0           PUNT   11  (For-us data)
1     INJ.2                      Gi1                        FWD
2     Gi1                        internal0/0/rp:0           PUNT   11  (For-us data)
3     INJ.2                      Gi1                        FWD
4     Gi1                        internal0/0/rp:0           PUNT   11  (For-us data)
5     INJ.2                      Gi1                        FWD
6     Gi1                        internal0/0/rp:0           PUNT   11  (For-us data)
7     INJ.2                      Gi1                        FWD
8     Gi1                        internal0/0/rp:0           PUNT   11  (For-us data)

9     Gi1                        Gi1                        DROP   8   (Ipv4Acl)


10    Gi1                        internal0/0/recycle:0      PUNT   26  (QFP ICMP generated packet)
11    INJ.9                      Gi1                        FWD

12    Gi1                        Gi1                        DROP   8   (Ipv4Acl)


13    Gi1                        internal0/0/recycle:0      PUNT   26  (QFP ICMP generated packet)
14    INJ.9                      Gi1                        FWD

15    Gi1                        Gi1                        DROP   8   (Ipv4Acl)


16    Gi1                        internal0/0/recycle:0      PUNT   26  (QFP ICMP generated packet)
17    INJ.9                      Gi1                        FWD

18    Gi1                        Gi1                        DROP   8   (Ipv4Acl)


19    Gi1                        internal0/0/recycle:0      PUNT   26  (QFP ICMP generated packet)
20    INJ.9                      Gi1                        FWD

21    Gi1                        Gi1                        DROP   8   (Ipv4Acl)
```

```
22    Gi1                        internal0/0/recycle:0     PUNT   26  (QFP ICMP generated packet)
23    INJ.9                      Gi1                       FWD

24    Gi1                        Gi1                       DROP   8   (Ipv4Acl)


25    Gi1                        internal0/0/recycle:0     PUNT   26  (QFP ICMP generated packet)
26    INJ.9                      Gi1                       FWD
```

With this output, you can see each packet arriving and leaving the device, as well as ingress and egress interfaces. The status of the packet is also shown, indicating whether it was forwarded, dropped, or processed internally (punt).

In this example, this output helped to identify the packets that are being dropped by the device. Using the command **show platform packet-trace packet <PACKET_NUMBER>**, you can see how the device processes that specific packet.

Spoke1 **show platform packet-trace packet <PACKET_NUMBER>** output:

<#root>

```
SPOKE1#show platform packet-trace packet 9
Packet: 9 CBUG ID: 9
Summary

Input : GigabitEthernet1



Output : GigabitEthernet1



State : DROP 8 (Ipv4Acl)


Timestamp
Start : 366032715676920 ns (02/01/2024 04:30:15.708990 UTC)
Stop : 366032715714128 ns (02/01/2024 04:30:15.709027 UTC)
Path Trace
  Feature: IPV4(Input)


Input : GigabitEthernet1


     Output : <unknown>


Source : 172.22.200.2



     Destination : 172.21.100.1



     Protocol : 50 (ESP)


  Feature: DEBUG_COND_INPUT_PKT
```

```
      Entry : Input - 0x812707d0

  Input : GigabitEthernet1


      Output : <unknown>

      Lapsed time : 194 ns
   Feature: IPV4_INPUT_DST_LOOKUP_ISSUE
      Entry : Input - 0x8129bf74

  Input : GigabitEthernet1


      Output : <unknown>

      Lapsed time : 769 ns
   Feature: IPV4_INPUT_ARL_SANITY
      Entry : Input - 0x812725cc

  Input : GigabitEthernet1


      Output : <unknown>

      Lapsed time : 307 ns
   Feature: EPC_INGRESS_FEATURE_ENABLE
      Entry : Input - 0x812782d0

 Input : GigabitEthernet1


      Output : <unknown>

      Lapsed time : 6613 ns
   Feature: IPV4_INPUT_DST_LOOKUP_CONSUME
      Entry : Input - 0x8129bf70

 Input : GigabitEthernet1


      Output : <unknown>

      Lapsed time : 272 ns
   Feature: STILE_LEGACY_DROP
      Entry : Input - 0x812a7650

  Input : GigabitEthernet1


      Output : <unknown>
      Lapsed time : 278 ns
   Feature: INGRESS_MMA_LOOKUP_DROP
      Entry : Input - 0x812a1278

Input : GigabitEthernet1

      Output : <unknown>

      Lapsed time : 697 ns
   Feature: INPUT_DROP_FNF_AOR
      Entry : Input - 0x81297278

Input : GigabitEthernet1

      Output : <unknown>

      Lapsed time : 676 ns
   Feature: INPUT_FNF_DROP
      Entry : Input - 0x81280f24

Input : GigabitEthernet1

      Output : <unknown>

      Lapsed time : 1018 ns
   Feature: INPUT_DROP_FNF_AOR_RELEASE
      Entry : Input - 0x81297274

Input : GigabitEthernet1

      Output : <unknown>

      Lapsed time : 174 ns
   Feature: INPUT_DROP

      Entry : Input - 0x8126e568

Input : GigabitEthernet1

      Output : <unknown>

      Lapsed time : 116 ns

Feature: IPV4_INPUT_ACL
```

```
     Entry : Input - 0x81271f70

Input : GigabitEthernet1


     Output : <unknown>

     Lapsed time : 12915 ns
```

In the first part, you can see the ingress and egress interface, and the state of the packet. This is followed by the second part of the output where you can find the source and destination IP addresses and the protocol.

Each subsequent phase shows how the device process this particular packet. This offers insights into any configurations like Network Address Translation (NAT) or access-list or other factors that could be impacting it.

In this case, it can be identified that the protocol of the packet is ESP, the source IP is the NBMA IP address of Spoke2, and the destination IP is the NBMA IP address of Spoke1. This indicates that this is the missing packet in the NHRP negotiation. Also, it is observed, that no egress interface is specified in any phase, suggesting that something affected the traffic before it could be forwarded. On the penultimate phase you can see that the device is dropping the inbound traffic on the interface specified (GigabitEthernet1). The last phase shows an input access-list, suggesting that there can be some configuration on the interface causing the drop.

Note: If after using all the troubleshooting tools listed in this document, the spokes involved in the negotiation are not showing any signs that they are dropping or affecting the traffic, that concludes the troubleshooting on those devices.

The next step must be to check on middle devices between them, such as firewalls, switches, and ISP.

Solution

If such a scenario is seen, the next step is to check the interface shown in the previous outputs. This involves checking the configuration to verify if there is anything affecting the traffic.

WAN interface configuration:

<#root>

```
SPOKE1#show running-configuration interface gigabitEthernet1
Building configuration...

Current configuration : 150 bytes
```

```
!
interface GigabitEthernet1
ip address 172.21.100.1 255.255.255.0
```

```
ip access-group ESP_TRAFFIC in
```

```
negotiation auto
no mop enabled
no mop sysid
end
```

As part of its configuration, the interface has an access-group applied. It is important to verify that the hosts configured on the access-list are not interfering with the traffic used for the NHRP negotiation.

<#root>

```
SPOKE1#show access-lists ESP_TRAFFIC
Extended IP access list ESP_TRAFFIC
10 deny esp host 172.21.100.1 host 172.22.200.2
```

```
20 deny esp host 172.22.200.2 host 172.21.100.1 (114 matches)
```

```
30 permit ip any any (22748 matches)
```

The second statement of the access-list is denying the communication between the NBMA IP address of Spoke2 and the NBMA IP address of Spoke1, causing the drop previously seen. After removing the access-group from the interface, the communication between the two spokes is successful:

```
SPOKE1#ping 192.168.2.2 source loopback1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/3 ms
```

The IPSEC tunnel is up and now it is showing encaps and decaps on both devices:

Spoke1:

<#root>

```
SPOKE1#show crypto IPSEC sa peer 172.22.200.2

interface: Tunnel10
    Crypto map tag: Tunnel10-head-0, local addr 172.21.100.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
   current_peer 172.22.200.2 port 500
     PERMIT, flags={origin_is_acl,}
```

```
#pkts encaps: 6, #pkts encrypt: 6, #pkts digest: 6
```

```
    #pkts decaps: 7, #pkts decrypt: 7, #pkts verify: 7
```

```
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 172.21.100.1, remote crypto endpt.: 172.22.200.2
     plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
     current outbound spi: 0x9392DA81(2475874945)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0xBF8F523D(3213840957)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2073, flow_id: CSR:73, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607998/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0x9392DA81(2475874945)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2074, flow_id: CSR:74, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607999/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```

Spoke2:

<#root>

```
SPOKE2#show crypto IPSEC sa peer 172.21.100.1

interface: Tunnel10
    Crypto map tag: Tunnel10-head-0, local addr 172.22.200.2

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
   current_peer 172.21.100.1 port 500
```

```
     PERMIT, flags={origin_is_acl,}
```

```
#pkts encaps: 7, #pkts encrypt: 7, #pkts digest: 7
```

```
    #pkts decaps: 6, #pkts decrypt: 6, #pkts verify: 6
```

```
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 172.22.200.2, remote crypto endpt.: 172.21.100.1
     plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
     current outbound spi: 0xBF8F523D(3213840957)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0x9392DA81(2475874945)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2073, flow_id: CSR:73, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607998/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0xBF8F523D(3213840957)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2074, flow_id: CSR:74, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607999/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```

Now the DMVPN table of Spoke1 is showing the correct mapping on both entries:

<#root>

```
SPOKE1#show dmvpn

Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        T1 - Route Installed, T2 - Nexthop-override, B - BGP
        C - CTS Capable, I2 - Temporary
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

# Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
----- --------------- --------------- ----- -------- -----


1 172.22.200.2     10.10.10.2     UP   00:01:31   D


     1 172.20.10.10     10.10.10.10    UP    1d05h     S
```