Authors: S. Dikshit   V. Joshi   A. Sujeet Nayak
         Aruba, HPE   Aruba, HPE   Cisco

MTU propagation over EVPN Overlays

Abstract

Path MTU Discovery between end-host-devices/Virtual-Machines/
servers/workloads connected over an EVPN-Overlay Network in an
Datacenter/Campus/enterprise deployment, is a problem, yet to be
resolved in the standards forums. It needs a converged solution to
ensure optimal usage of network and computational resources of the
underlay routers/switches forming the basis of the overlay network.
This documents takes leads from the guidelines presented in
[RFC4459].

The overlay connectivity can pan across various sites
(geographically seperated or collocated) for realizing a Datacenter
Interconnect or intersite VPNs between campus sites (buildings,
branch offices etc).

This literature intends to solve problem of icmp error propagation
from an underlay routing/switching device to an end-host (hooked to
EVPN overlay), thus facilitating "accurate MTU" learnings.

This document also leverages the icmp multipart message extension,
mentioned in [RFC4884] to carry the original packet in the icmp PDU.

Status of This Memo

This Internet-Draft will expire on 5 February 2022.

Copyright Notice

Table of Contents

1. Introduction

There is an operational disconnect between underlay network provisioned as the underlay network, and the overlay network which intends to connect islands of customer deployments. The deployments can range from cloud based services to storage applications or

web(over the top) servers hosted over virtual machines or any other
end devices like blade servers. Overlay network are provisioned as
tunnels leveraging Vxlan (and associated ones like gpe, geneve,
gue), NVGRE, MPLS and other overlay encapsulations.

The end hosts (VMs, workloads, user-devices) in a datacenter/campus
deployment are connected to gateway. In case the core network is
laid out with EVPN-overlays, the gateways are Vteps (Vxlan-fabric
gateways). These are the networking devices which encapsulate the
packet in an Overlay construct and relays it over the underlay
network.

For campus deployments, various branch offices can be provisioned
with EVPN-overlays and the vpn connectivity between them can be
realized via EVPN-overlays (VXLAN, MPLS, NVGRE fabrics) itself. Thus
it involves interworking/stitching of same/different overlays at
DCI/VPN transit routing/switching devices. The transit devices can
be on-premise WAN gateways (SDWAN or otherwize) or service provider
network entry points.

IPv6/IPv4 enabled hosts/end-points, triggering PMTUD, may not get
the right/inconsistent (or none) information from (over) the
underlay network in case MTU errors are encountered in the packet
path (encapsulated in the overlay). This document validates the
detailed solution for Vxlan-fabric (though equally applicable to
other EVPN-overlays like Geneve, GUE, GPE, NVGRE) faciliated by an
underlay network (via any routing protocol like BGP, OSPF, ISIS,
EIGRP etc). This solution is equally applicable to other tunnel/
overlay specifications falling into EVPN-overlay category.

The proposal in this document, formulates an integrated approach
which falls inline with OAM modelling discussed in NVO3.

2. Requirements

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

When used in lowercase, these words convey their typical use in
common language, and they are not to be interpreted as described in
[RFC2119].

2.2. Solution Requirements

This section describes the advantages of the proposed solution,
considering deployment in a typical EVPN-overlay underlay network:

(a)   Optimal use of bandwidth in underlay and end-host network.

(b)   In case Vxlan Gateway nodes complies to this solution, it MAY avoid black holing of icmp errors generated by underlay network devices.

(c)   All end host applications (like web servers) can tailor the MSS accordingly against their respective transports.

(d)   Facilitates seamless integration of IPv6 or dual stack applications over IPv4 based overlays and vice versa.

(e)   The proposed solution is applicable to all encapsulations [RFC7348], [I-D.draft-ietf-nvo3-vxlan-gpe], [I-D.draft-ietf-nvo3-gue], [I-D.draft-gross-geneve] and [RFC7637]. Although the problem and solution refers to VXLAN [RFC7348] as a use-case in this document.

3. Problem Description

In current vendor implementation(s) of Vxlan-Gateway or other network devices, which form part of the underlay network and is configured with an overlay(tunnel) mechanism to transport packets from one customer end point to another, are incapable of relaying the errors encountered in routing/switching path in their networks (underlay network) to the customer end points (hosts/vm/blade-servers). This deems right, as the underlay network should be transparent and water-tight with respect to leaking any public (underlay) network information to customer devices (and vice versa), thus ensuring seclusion between different customers provisioning tunneled over the common underlay network.

For example, the information carried in the IP header of a Vxlan encapsulated packet is transparent to the payload (end-point generated packet). Hence, any network-specific information related to IPv6/IPv4 native functionality is carried to the end-point devices, as is the case with an end-to-end private network. The information generated in the underlay network devices while processing packets destined-to/sourced-from end-point devices, need to be percolated from underlay encapsulation to end customer specific payload. This is something which is NOT directed by any standards, and also NOT implemented by current deployment(s) of routers and switches.

Thus end-host sending out packets may never know about a lingering problem, impacting it's traffic in the underlay network.

Note that terms "icmpv6" or "icmpv4" are used in the document with
an intention to refer to both icmp and icmpv6, in case same context
applies to both.

3.1. Issues in MTU propagation in an underlay

As mentioned in the [RFC1981], IPV6 PMTUD is based on the "Packet
too big" icmpv6 error code, generated by the networking device which
is capable of generating such messages on encountering packet paths
which go over link with MTU size smaller than packet size.

There are problems getting this working when end-point device
initiates a "Path MTU Discovery" to remote end-point device. It may
lead to black-holing as per the current implementations.

The following bullets provides pointers to potential black holing of
PMTUD packets,

(1)   Vxlan Gateway MAY not set the DF bit in the outer IP header
        encapsulation.

(2)   Vxlan Gateway is incapable of relaying icmp error
        "Fragmentation Needed and Don't Fragment was Set", generated
        by IPv4 enabled underlay network device, to IPv6 enabled end-
        point host/vm/server(source of the original packet).

The problems are discussed in detail in the following sub-sections.

3.1.1. Inaccurate MTU relayed to end hosts

Figure 1 depicts the topology referenced in the document for
explaining the problem statement and the solution.

```
+----------+                      +----------+
|    H1    |                      |    H2    |
|          |                      |          |
|(H1_IPv6) |                      |(H2_IPv6) |
+----------+                      +----------+
     |                                 |
     |                                 |
+------------+   +----------+   +------------+
|(VtepA_IPv6)|   |          |   |(VtepB_IPv6)|
|   VtepA    |   |    R1    |   |   VtepB    |
|(VtepA_IPv4)|---| (R1_IPv4)|---|(VtepB_IPv4)|
+------------+   +----------+   +------------+
```
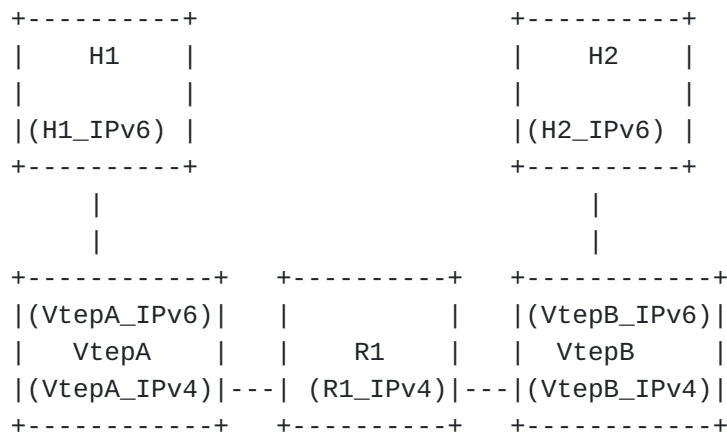
Figure 1. L3 Overlay

```
      LEGEND:
      MAC address : <Node_name>_MAC
      IP address  : <Node_name>_IPv4
      IPv6 address: <Node_name>_IPv6
      <Node_name> : node names in the above topology are
                    H1, VtepA, R1, VtepB, H2.
      VtepA, VtepB: Vxlan gateways
               R1: Intermediate router in underlay network
            H1,H2: End-point devices communicating withe each other
```

H1 and H2 are the end point hosts in different subnet connected over
Vxlan Overlays in the underlay network. The Vtep tunnel end points,
christened as VtepA and VtepB, are reachabile over an underlay IPv4
network. In this example, VtepA and VtepB are dual stack enabled and
act as Vxlan gateways to connected hosts. Link mtu between VtepA, R1
and VtepB is configured as 1300 bytes; where as for the links
between H1 and VtepA, H2 and VtepB, it is configured as 1500 bytes.

H1 sends out a packet obliging to 1500 bytes MTU packet size
containment over the H1 and VtepA link. VtepA encapsulates the
packet with (Vxlan + UDP) header and outer IP header corresponding
to underlay reachability to destination tunnel end-point, that is
VtepB, to reach out to H2.

If size of encapsulated packet to be send over the link VtepA-R1
exceeds the MTU (1300 bytes). IPv4 packet with (IP header + UDP
header + Vxlan header + Original L2 Packet from H1 containing the
IPv6 Payload) SHOULD be fragmented. In case Vxlan gateway, VtepA,
does not sets the DF-bit in the outer IP header, the packet gets
fragmented, with the reassembly done at the egress gateway (VtepB).

The re-assembled packet is routed by VtepB to H2. This can
potentially lead to inaccurate Path MTU calculation at H1. H1
assumes it to be 1500 bytes as no icmp error is received. This opens
the door for fragment/reassembly and more cpu cycles on networking
devices in the underlay network.

3.1.2. Packet_Too_Big not-relayed to host

In figure 1, assume that link between VtepA and R1 is 1500 as the
only change from the figure 1 topology. Hence the packet send by H1,
leads to VtepA setting the DF-bit in the outer IP header(as part of
Vxlan Encapsulation). When R1 receives the packet and the routing
table lookup points to the outgoing link with mtu size R1_VtepB_MTU
bytes, less than the packet size (1500 bytes). As DF-bit is set, R1
generates icmpv4 error directed towards the src-ip (VtepA_IPv4). It
encapsulates the inner PDU of the original packet. However, VtepA
drops the icmp error packet and fails to relay it to H1. This leads
to black-holing.

The above two sub-sections lay down potential problems for IPv6 Path
MTU Discovery mechanism in an Overlay network. Although these
problem are generic to any combination of underlay and overlay
network types (IPv4 or IPv6), the use-case topology in this document
is specific to IPv6 end-point devices connected over Vxlan network,
wherein, the underlay is connected over IPv4 network, unless
mentioned specifically.

4. Solution(s)

4.1. Discovery of end-to-end Path MTU

Since Vxlan Gateway is the one, which encapsulates the Vxlan (or any
other overlay) header onto the packet traversing through the overlay
network and also decapsulates the overlay header for packets
egressing out of same and heading towards the end devices, the
solution becomes more apt to be installed on devices playing such
role.

Firstly, It is a MUST that Vxlan gateways (VtepA and VtepB) SHOULD
set the DF-bit in Outer header encapsulation for client packets that
are wrapped with vxlan, related encapsulation, for Path MTU
Discovery. Thus ensuring that icmp error packet is generated for
packet size exceeding the link MTU in underlay network.

Secondly, it is MUST that Vxlan gateway devices translates the icmp
error "Destination Unreachable" with code 'Fragmentation Needed and
Don't Fragment was Set', into a icmpv6 error 'Packet too big'
packet. This mandates that original packet carried in the icmp error
message MUST carry information about the inner payload(original
packet), and it is an IPv6 Packet, originated from the end-point
device (H1 for VtepA in figure 1), connected to the Vxlan gateway
over L3/L2 network.

Thirdly, it is MUST that Vxlan gateway devices translates the icmpv6
error 'Packet too big' into a icmp error 'Destination Unreachable'
with code 'Fragmentation Needed and Don't Fragment was Set' packet.
Successfully translation mandates that, original packet carried in
the icmp error message gives information about the inner payload
(original packet), and it is an IPv4 packet, which originated from
the end-point device connected to gateway over L3/L2 network.

Fourthly, incase both, the client side network connected to Vxlan
Gateway and the underlay network are same, that is, either both are
ipv4 or both are ipv6, then icmp error code error translation is NOT
required. Rest of the process to retrieve original packet is
identical.

4.1.1. ICMP extensions leveraged for MTU propagation

This solution leverages extensions in icmp and icmpv6 standards,
[RFC4884], for the maximum size of the original packet that can be
encapsulated in icmp error message with code as "Fragmentation
Required(icmp)" or "Packet too big(icmpv6)" respectively. As the
host info is encapsulated in the inner payload, this requires
additional bytes of data in icmp packet: (Outer IP Header + UDP
Header + Vxlan + Inner L2 Header + Inner IPv6 SRC/DST IPs).

In case Vxlan underlay network is provisioned over IPv6 underlay,
then similar extensions are applicable to icmpv6.

The processing of icmpv6 packet is extended from the current
standards of 'non-delivery of icmpv6 packets to upper-layers on
Vxlan gateways' to 'relaying it to the end-point devices'.

4.1.2. Packet Path Processing

Packet Path handling and processing is explained in this section.
The assumptions are made with respect to network topology mentioned
in Section 3.1.1. The packet format in each flow captures packet
fields which are significant with respect to this solution. To
understand the solution, the packet flow is explained which leads to
generation of icmp or icmpv6 error by intermediate node in underlay
network.

IPv6 packet is sent by host H1 destined to host H2, both are in
different IPv6 subnets.This packet is referred to as P1 in the
document.

```
      +----------------------------------------------------+
H1--|L2_Hdr(14 bytes): src-mac:H1_MAC, dest-mac:VtepA_MAC|-->VtepA
      +----------------------------------------------------+
      |IPv6_Hdr(40 bytes): src-ip:H1_IPV6, dest-ip:H2_IPv6 |
      +----------------------------------------------------+
      |Host/App specific Payload                           |
      +----------------------------------------------------+
      Figure 2a. Packet P1 sent by host H1 to host H2
```

VtepA re-writes the mac addresses in 'P1' as part of Vxlan
encapsulation. This encapsulation is referred as 'P2' in the
document.

```
       +---------------------------------------------------------+
H1--|L2_Hdr(14 bytes):src-mac:VtepA_MAC, dest-mac:VtepB_MAC|-->VtepA
       +---------------------------------------------------------+
       |IPv6_Hdr(40 bytes): src-ip:H1_IPV6, dest-ip:H2_IPv6   |
       +---------------------------------------------------------+
       |Host/App specific Payload                             |
       +---------------------------------------------------------+
       Figure 2b. Packet P1 re-written by VtepA
```

4.1.2.1. Packet Processing at Vxlan Gateway

Processing at VtepA, in packet path from H1 to H2.

(1)   VtepA(Vxlan gateway) performs the Vxlan encapsulation over the
          packet received from H1, based on route lookup. The detail for
          encap are mentioned in [RFC7348].

(2)   VtepA MUST set the DF-bit in the Outer IP header.

(3)   Since the MTU of outgoing link is more than the packet, packet
          is sent out towards the underlay next hop, R1.

(4)   P3 packets encapsulation is shown in figure 3. P3 may find a
          reference without outer header encapsulation [RFC7348]
          provides details of the vxlan encapsulation.

```
       +-------------------------------------------------------------+
VtepA-|L2_Hdr(14bytes):src-mac:VtepA_Mac, dest-mac:R1_MAC        |-->R1
       +-------------------------------------------------------------+
       |IPv4_Hdr(20 bytes):src-ip:VtepA_IPv4,dest-ip:VtepB_IPv4,DF|
       +-------------------------------------------------------------+
       |UDP(8 bytes): src-port: ephemeral-port, dest-port: 4789   |
       +-------------------------------------------------------------+
       |Vxlan(8 bytes): Vxlan network identifier                  |
       +-------------------------------------------------------------+
       |P2 packet (refer to H1 to VtepA flow for details of P1)   |
       +-------------------------------------------------------------+
       Figure 3. Vxlan Encap packet sent by Vxlan Gateway to underlay
```

4.1.2.2. Underlay Generates ICMP error

In case the underlay is ipv6 and not ipv4, icmpv6 error is
generated.

Processing at R1:

(1)   Packet Size (1500 bytes) is more than the outgoing link's mtu
          (1300 bytes) and DF-bit is set in the Outer IPv4 header added
          as part of Vxlan encapsulation at VtepA.

(2)   R1 MUST generate icmp error message (Destination Unreachable)
with error code (Fragmentation Needed and Don't Fragment was
Set). For ease of solution description, mtu is assumed to be
symmetric over the reverse path, hence reverse path mtu from
R1 to VtepA is 1500 bytes. icmpv6 error message MUST include
MTU of link between R1 and VtepB.

(3)   In a nut shell, the icmp PDU encapsulation SHOULD be performed
as mentioned in [RFC4884] , [RFC4443]. These standards atleast
ensure, that original packet carried in icmp error PDU
captures enough bytes to include the inner packets IPv6 header
atleast. The capture of application specific details depends
on the size of the Optional header in the original packet
(generated by H1 as in Figure 2b) and subsequent transport
header. This helps Vxlan Gateway to trace(L3 reachability) the
original packet generator (end-point device) atleast and
translate icmp error generted by underlay into icmpv6 one and
relay it to end-point device. The length field in icmp PDU,
include the maximum possible length permissible in reverse
path MTU

For simplicity, not including the original packet header in the flow
diagram in figure 4. icmp PDU details are depicted in the follow up
figure 5.

```
      +------------------------------------------------------------+
R1-|L2_Hdr(14 bytes): src-mac:R1_MAC, dest-mac:VtepA_MAC        |-->VtepA
      +------------------------------------------------------------+
      |IPv4_Hdr(20 bytes): src-ip:R1_IPv4, dest-ip:VtepA_IPv4     |
      +------------------------------------------------------------+
      |ICMP PDU,type:3,code:4,R1_VtepB_MTU, P3(No outer L2 Header)|
      +------------------------------------------------------------+
             Figure 4. Flow diagram from R1 to VtepA
```

The details of icmp PDU are in the following figure. Type '3' is
"Destination Unreachable". Code '4' is "Fragmentation Needed and
Don't Fragment bit is set".

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4s 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    Type=3    |    Code=4    |          Checksum              | ICMP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=3
|    unused    |   Length    |   Next Hop Mtu = R1_VtepB_MTU   | Code=4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5 |  TOS         |        Total length            |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|        Id                  |Flags| Fragment Offset          |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| TTL          | Protocol=UDP | Header Checksum               |(Outer)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+Max 40
|           src-ip  : VtepA_IPv4                              |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|           dest-ip : VtepB_IPv4                              |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|   Source UDP Port (ephemeral) | Dest UDP Port = 4789 (Vxlan) |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
|  Length                    |   Checksum                     |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| | | | | | | | |     Reserved                               |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
|Vxlan Network identifier (VNI)              | Reserved       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|            Inner Packet Dest-Mac = VtepB_MAC               |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                            |   Inner Packet Src-Mac =       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Inner
|                 VtepA_MAC                                   |14 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Inner Vlan if present      |Ethtype = 0X86dd (IPv6)         |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class  |     Flow Label                     |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|payload length             |Next Header    | Hop Limit       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                            |  |
|            src-ipv6   =    H1_IPv6                          |IPv6
|                                                            |Header
|                                                            |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                            |  |
|            dest-ipv6  =    H2_IPv6                          |  |
|                                                            |  |
|                                                            |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    ~       Optional Headers and  transport header/Payload ~ | Varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
        Figure 5. ICMP PDU Original Packet Capture in Detail
```

4.1.2.3. Relay ICMP(v6) Error to End Devices

This sub-section can also be generalized as: "handling of icmp errors, which are generated by underlay network in response to end-device packets, by Vxlan Gateway".

Processing at VtepA: Processing of icmp error message with code (Fragmentation Needed and Don't Fragment was Set):

(1)   The icmp error is processed by Vxlan gateways as per the standards defined in [RFC1981] , [RFC4884] and [RFC4443] .

(2)   If error code is (Fragmentation Needed and Don't Fragment was Set), it SHOULD perform further inspection of the original packet, P3(ethernet payload without its header) carried as data in icmp PDU in extension to standards referred in previous bullet. The extension processing MUST be done prior to taking a decision to either drop the packet or deliver to upper-layer protocols.

(3)   In extension to above, Vxlan gateway device SHOULD perform the vxlan decap as defined in [RFC7348], to arrive at the inner packet (P2, original packet with VtepA rewrite). The underlay encap is not carrying the layer-2 header in the icmp error packet. Once this processing is done, P2 is the packet which needs attention now, as it carries the credentials of actual host which should receive the relayed icmp packet.

(4)   Post encap, the VNI should be cached and a check should be made if its a Layer-2 VNI (L2VNI) or an Layer-3 VNI (L3VNI). If it's an L2VNI, go to the next step to check on the payload (ethernet) type. If and only if, it is ipv6 or ipv4, then only process it further else terminate the processing. If it's an L3VNI, then the mapping VRF should found to perform the route lookup for inner packet source IP address.

(5)   The layer-3 payload type SHOULD be verified using ethernet type field in ethernet header. In case it point to IPv6, src-ipv6 field should be picked up to check for reahability, as the icmp packet MUST be sent to original sender, that is, H1. In case H1 is reachable, icmp packet SHOULD be constructed as mentioned in the following bullet.

(6)   Now that P2 is out in the open, it's L2 header is decapsulated, and the leftover, in the figure 6, is run through the icmpv6 processing as mentioned in [RFC4443].

(7)   It SHOULD generate icmpv6 error message with type (Packet too big) destined to H1_IPv6, that is inner ipv6 packet's source

ipv6 address. The mtu 'R1_VtepB_MTU' is copied from icmp error
packet recieved from the underlay.

(8)   The IPv6 header is constructed from original payload as shown
in figure 5. The source ipv6 address is picked as local ipv6
address "VtepA_IPv6". The destination ipv6 address is set as
the "src-ipv6" in original payload, H1_IPv6. The Next Header
is set as "58" which denote icmpv6. The derivation of ethernet
header is based on next hop to mac address mapping as is
performed in any L3 lookup. The follow up figure 9, shows the
icmpv6 error packet sent out to node H1. H1 is the original
IPv6 packet generator as mentioned in Figure 2b.

(9)   The route lookup is performed for H1_IPv6 in the VNI mapped
VRF, as also mentioned in one of the earlier bullets. Thus
{inner packet source IP, VNI} as a tuple is required to
resolve the path back to the inner packet source.

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class  |      Flow Label                 |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|payload length                 |Next Header   | Hop Limit    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                             |  |
|             src-ipv6   =    H1_IPv6                         | Inner
|                                                             | IPv6
|                                                             | 40 byt
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                             |  |
|             dest-ipv6  =    H2_IPv6                         |  |
|                                                             |  |
|                                                             |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    ~  Optional Headers and  Transport/Application Payload ~  | Varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
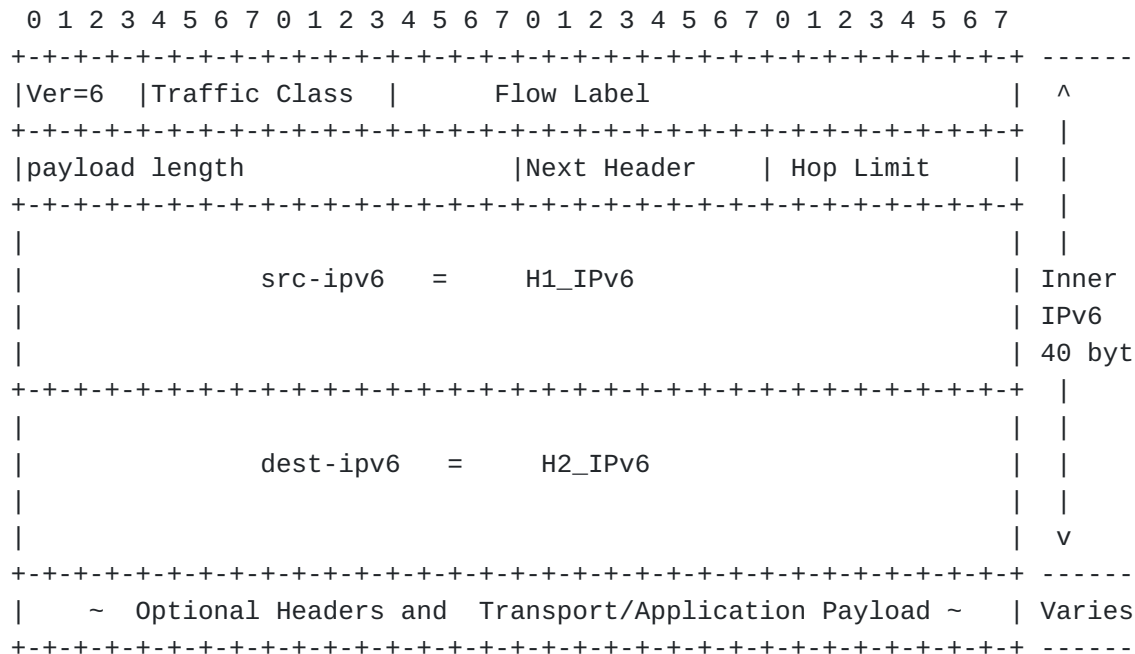   Figure 6. Original IPv6 Packet sent from H1 directed to H2

   Figure 6 gives a typical IPv6 format sent by end-host, H1 towards H2
   and encapsulated by Vxlan gateway, to translate the icmp error
   generated by underlay hop, R1, to the one understood in right
   context by H1.

```
  0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|   Type=2     | Code=0        |        CheckSum              |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=2
|             Mtu = R1_VtepB_MTU                              |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class  |     Flow Label                     |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|payload length               |Next Header   | Hop Limit    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                            |  |
|            src-ipv6   =    H1_IPv6                         | Orig
|                                                           | Packet
|                                                           |40 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                            |  |
|            dest-ipv6   =    H2_IPv6                        |  |
|                                                            |  |
|                                                            |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ -----
|    ~ Optional/Transport Headers and Application Payload ~    |varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
 Figure 7. ICMPv6 "Packet Too Big" PDU relayed
           to H1 by Vxlan Gateway (VtepA)

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|                    Dest-Mac = H1_MAC                     |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                               |     Inner Packet Src-Mac =  |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+eth hdr
|                    VtepA_MAC                             |14 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Inner Vlan if present        |Ethtype = 0X86dd (IPv6)    |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class  |     Flow Label                  |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|payload length                |Next Hdr = 58  | Hop Limit |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                         |  |
|            src-ipv6    =     VtepA_IPv6                  | IPv6
|                                                         |header
|                                                         |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                         |  |
|            dest-ipv6   =     H1_IPv6                     |  |
|                                                         |  |
|                                                         |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
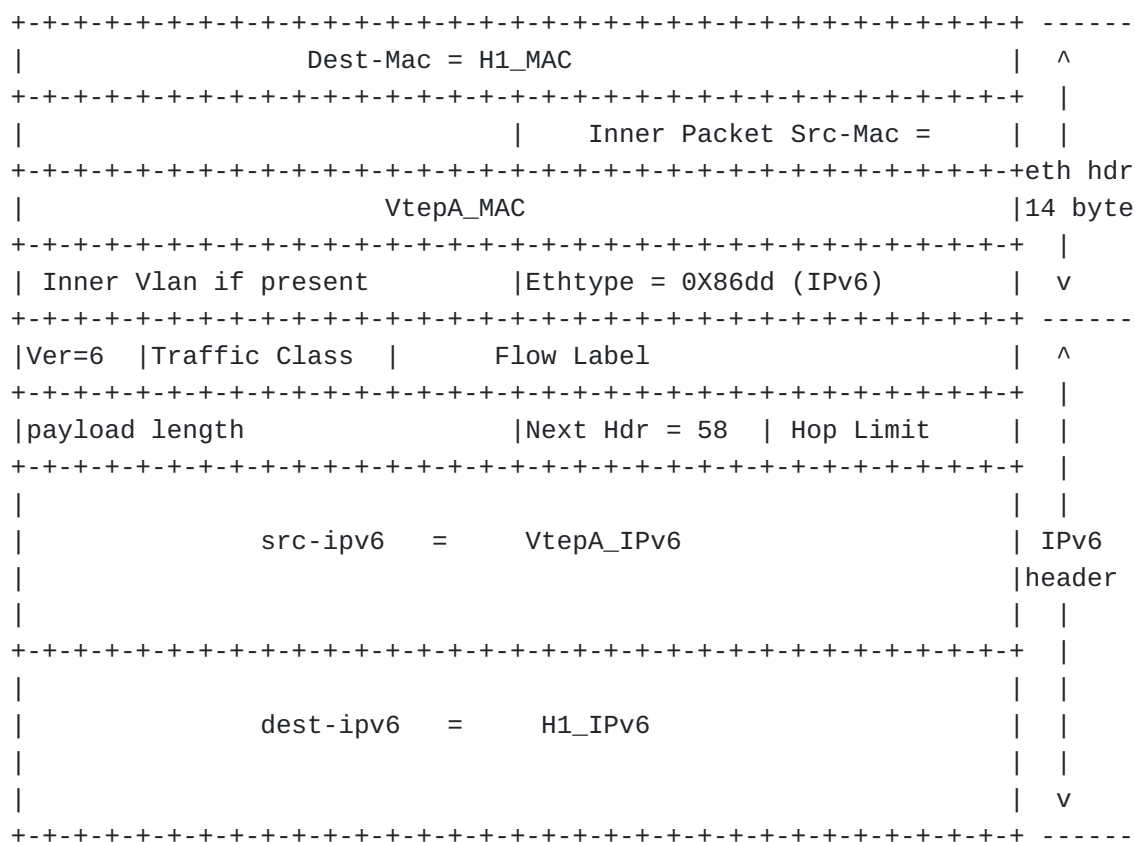
Figure 8. Ethernet and IPv6 encap for ICMPv6 PDU mentioned in
          figure 7

   The translated icmp packet encapsulation looks similar to, figure 7
   and figure 8 put together in reverse order. The flow diagram in
   figure 9 gives a concise form of "packet too big" icmpv6 error
   relayed by VtepA (Vxlan Gateway) towards H1 (end point device).

```
        +--------------------------------------------------------+
VtepA--|L2_Hdr(14): src-mac:VtepA_MAC and Dest_Mac: H1_MAC      |-->H1
        +--------------------------------------------------------+
        |IPv6_Hdr(40 bytes): src-ip:Vtep_IPv6, dest-ip:H1_IPv6   |
        +--------------------------------------------------------+
        |ICMPv6: Packet_Too_Big, mtu, data: first 128 bytes of P3|
        +--------------------------------------------------------+
```
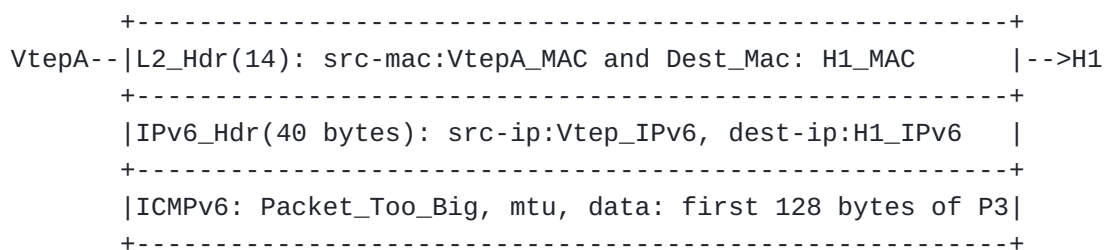Figure 9. Flow diagram: VtepA to H1

   There are few more potential flows worth mentioning in this section.
   These cases are related to, icmp error getting generated from,
   ingress Vxlan gateway (VtepA) and egress Vxlan gateway (VtepB) with
   respect to packet sent from H1 to H2. For ingress Vxlan gateway
   (VtepA) case, the legacy IPv6 PMTUD rules from [RFC4443] SHOULD be
   applied as no Vxlan encap is involved.

   Where as, egress Vxlan gateway (VtepB) SHOULD send packet P3
   (without L2 header) in the icmp data, even though mtu calculation

MAY be done post vxlan decapsulation. That is when the outgoing link
is identified as the one from VtepB to H2. It MAY buffer packet P3
prior to lookup based on inner packet (P2) credentials, so that P3
can be encapsulated in the icmp packet. This also ensures the packet
format consistency, when accessed at the VtepA for translation
before relaying it to H1.

4.1.3. ICMP(v6) Error Translation

This section specifically mentions about icmp and icmpv6 packet
translation, generated in an underlay network to the one which is,
understood by the end point device, with encapsulation aligning with
the network-type(IPv4 and IPv6), end-point device and underlay is
provisioned with. The last leg processing mentioned in previous sub-
section is specific to the topology mentioned in Section 3.1.1.
However, this subsection elaborates on all possible topology
combination of underlay and end-device networks with respect to IPv4
or IPv6. The explanation provided in form of figures for error
generated by underlay and the translated one relayed to the end-
point device by Vxlan gateway.

(a)   End-Point is IPv6 connected and Underlay is IPv4 provisioned.

(b)   End-Point is IPv4 connected and Underlay is IPv6 provisioned.

(c)   Both End-Point and Underlay are provisioned with IPv6.

(d)   Both End-Point and Underlay are provisioned with IPv4.

4.1.3.1. End-Point is IPv6 connected and Underlay is IPv4 provisioned

This case is similar to the last leg processing described in Section
4.1.2 and does not needs any more description.

4.1.3.2. End-Point is IPv4 connected and Underlay is IPv6 provisioned

Topology drawn in figure 10, provides for the icmpv6 PDU encap
generated by R1. H1_IPv4 and H2_IPv4 are in distinct ipv4 subnets.
R1_IPv6 represents IPv6 addresses falling in both subnets connecting
to VtepA and VtepB.

Another difference between an IPv4 and IPv6 underlay is that for
IPv6 underlay there is no concept of DF-bit. The fragmentation can
only be done at ingress. At all other underlay nodes "Packet too
big" icmpv6 error is generated. Vxlan Gateway SHOULD ensure that
fragmentation is avoided at Vxlan Gateway and icmp error is sent
back to H1. This procedure is applicable if and only if, original
packet contains DF-bit set in it's IP header.

```
+----------+                    +----------+
|    H1    |                    |    H2    |
|          |                    |          |
|(H1_IPv4) |                    |(H2_IPv4) |
+----------+                    +----------+
     |                               |
     |                               |
+-----------+   +----------+   +------------+
|(VtepA_IPv4)|   |          |   |(VtepB_IPv4)|
|   VtepA    |   |    R1    |   |   VtepB    |
|(VtepA_IPv6)|---| (R1_IPv6)|---|(VtepB_IPv6)|
+-----------+   +----------+   +------------+
```

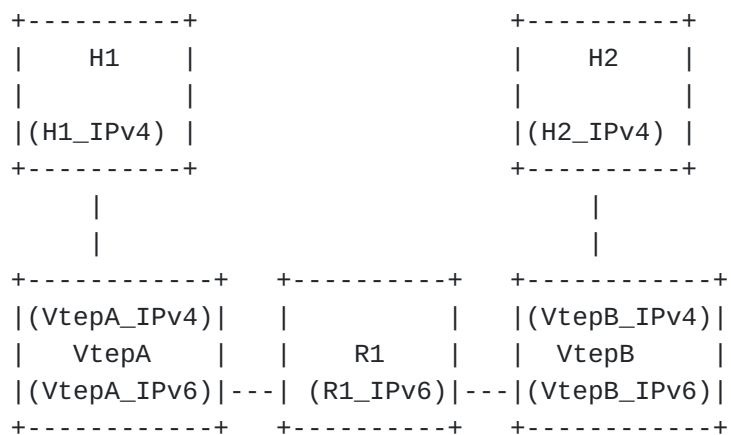   Figure 10. L3 Overlay

        LEGEND:
        MAC address : <Node_name>_MAC
        IPv4 address: <Node_name>_IPv4
        IPv6 address: <Node_name>_IPv6
        <Node_name> : node names in the above topology are
                      H1, VtepA, R1, VtepB, H2.
        VtepA, VtepB: Vxlan gateways to underlay network

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    Type=2     |    Code=0    |          Checksum             | ICMPv6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=2
|        Next Hop Mtu = R1_VtepB_MTU                           | Code=0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class  |      Flow Label                     |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|payload length                 |Next Hdr      | Hop Limit    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                             |  |
|           src-ipv6   =     R1_IPv6                          | IPv6
|                                                             |40 byte
|                                                             |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                             |  |
|           dest-ipv6  =     VtepA_IPv6                       |  |
|                                                             |  |
|                                                             |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|        ~        Extension Headers ~ (payload type is UDP)   |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|   Source UDP Port (ephemeral) | Dest UDP Port = 4789 (Vxlan) |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ 8 byte
|   Length                      |     Checksum                |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| | | | | | | | |    Reserved                                 |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ 8 byte
|Vxlan Network identifier (VNI)              | Reserved       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|             Inner Packet Dest-Mac = VtepA_MAC              |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                               |   Inner Packet Src-Mac =   |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+eth hdr
|                    VtepB_MAC                               |14 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Inner Vlan if present         |Ethtype = 0X0800 (IPv4)     |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5  |   TOS         |      Total length           |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|       Id                     |Flags| Fragment Offset       |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| TTL          | Protocol      | Header Checksum             | Orig
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Hdr
|         src-ip  : H1_IPv4                                   |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          dest-ip : H2_IPv4                                  |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    ~       transport-header and Application specific Payload ~ | varies
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
          Figure 11. ICMPV6 PDU Sent by R1 to VtepA

R1 sends an icmpv6 error "Packet Too Big" directed towards VtepA.
The icmpv6 PDU is shown in Figure 11. VtepA receives the packet with
this icmpv6 PDU and translates it to icmp PDU with type "Destination
Unreachable" and code "Fragmentation Needed" before relaying it to
H1 over ipv4 network. Figure 12, reflects the relayed packet sent by
VtepA to H1. All other references SHOULD be taken as it is from
Section 4.1.2.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|                    Dest-Mac = H1_MAC                       |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                               |     Inner Packet Src-Mac = |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+eth hdr
|                    VtepA_MAC                               |14 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Inner Vlan if present       |Ethtype = 0X0800 (IPv4)       |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5  |   TOS         |        Total length         |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          Id                   |Flags| Fragment Offset      |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| TTL          | Protocol=1    | Header Checksum             | IPv4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          src-ip  : VtepA_IPv4                              |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          dest-ip : H1_IPv4                                 |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          Optional Header                                   |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    Type=3    |    Code=4     |          Checksum            | ICMP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=3
|    unused    |    Length     |   Next Hop Mtu = R1_VtepB_MTU | Code=4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5  |   TOS         |        Total length         |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          Id                   |Flags| Fragment Offset      |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| TTL          | Protocol      | Header Checksum             |Orig
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+iPv4
|          src-ip  : H1_IPv4                                 |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|          dest-ip : H2_IPv4                                 |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|      Optional and Transport Header and Application data    | varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
Figure 12. ICMPv4 error Packet relayed to end point Host, H1

4.1.3.3. Both End-Point and Underlay are provisioned with IPv6

   Topology is mentioned in Figure 13 with minor changes along with the
   legend. Figure 14, outlines the icmpv6 PDU, encapsulation generated
   by R1. H1_IPv6 and H2_IPv6 in different ipv6 subnets. R1_IPv6
   reflects both subnets connecting to VtepA and VtepB.

```
+----------+                        +----------+
|    H1    |                        |    H2    |
|          |                        |          |
|(H1_IPv6) |                        |(H2_IPv6) |
+----------+                        +----------+
     |                                   |
     |                                   |
+------------+   +----------+   +------------+
|(VtepA_IPv6)|   |          |   |(VtepB_IPv6)|
|   VtepA    |   |    R1    |   |   VtepB    |
|(VtepA_IPv6)|---| (R1_IPv6)|---|(VtepB_IPv6)|
+------------+   +----------+   +------------+
```

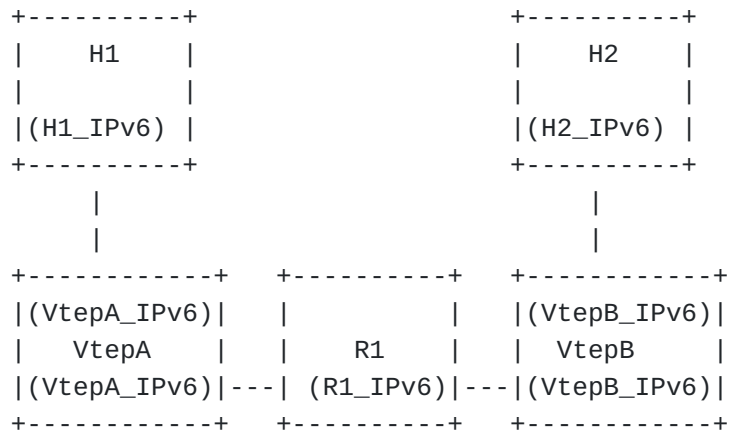   Figure 13. L3 Overlay

        LEGEND:
        MAC address : <Node_name>_MAC
        IPv6 address: <Node_name>_IPv6
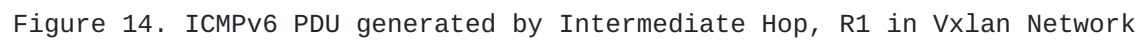   <Node_name> : node names in the above topology are
                      H1, VtepA, R1, VtepB, H2.
        VtepA, VtepB: Vxlan gateways to underlay network

```
  0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |    Type=2     |    Code=0     |             Checksum          | ICMPv6
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=2
 |        Next Hop Mtu = R1_VtepB_MTU                            | Code=0
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |Ver=6  |Traffic Class  |      Flow Label                      |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |payload length                 |Next Hdr       | Hop Limit    |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                                                              |  |
 |            src-ipv6   =      R1_IPv6                          | IPv6
 |                                                              | Header
 |                                                              |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                                                              |  |
 |            dest-ipv6  =      VtepA_IPv6                       |  |
 |                                                              |  |
 |                                                              |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |       ~       Extension Headers ~ (payload type is UDP)      |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |   Source UDP Port (ephemeral) | Dest UDP Port = 4789 (Vxlan) |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
 |   Length                      |   Checksum                   |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 | | | | | | | | |     Reserved                                |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
 |Vxlan Network identifier (VNI)                  | Reserved    |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |             Inner Packet Dest-Mac = VtepB_MAC               |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                               |  Inner Packet Src-Mac =      |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+eth hdr
 |                 VtepA_MAC                                    |14 byte
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 | Inner Vlan if present        |Ethtype = 0X0800 (IPv4)       |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |Ver=6  |Traffic Class  |      Flow Label                      |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |payload length                 |Next Hdr       | Hop Limit    |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                                                              |  |
 |            src-ipv6   =      VtepA_IPv6                       |Inner
 |                                                              | Ipv6
 |                                                              |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                                                              |  |
 |            dest-ipv6  =      H1_IPv6                          |  |
```

```
|                                                           |  |
|                                                           |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|          ~ Extension and Transport Headers, Application Data ~ | varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
Figure 14. ICMPv6 PDU generated by Intermediate Hop, R1 in Vxlan Network

R1 sends an icmpv6 error "Packet Too Big" directed towards VtepA. The icmpv6 PDU is shown in Figure 14. VtepA receives the packet with this icmpv6 PDU and relays it to H1 without any translation as H1 is connected to VtepA over ipv6 network. All other references about original packet to be include in the icmpv6 PDU can be taken as it is from Section 4.1.2.

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|                  Dest-Mac = H1_MAC                            |   ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|                              |    Inner Packet Src-Mac =      |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+eth hdr
|                   VtepA_MAC                                   |14 byte
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
| Inner Vlan if present        |Ethtype = 0X86dd (IPv6)         |   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class |      Flow Label                       |   ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|payload length                |Next Hdr      | Hop Limit       |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|                                                              |   |
|           src-ipv6   =      VtepA_IPv6                        |IPv6
|                                                              | Header
|                                                              |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|                                                              |   |
|           dest-ipv6  =      H1_IPv6                           |   |
|                                                              |   |
|                                                              |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|         ~      Extension Headers ~ (payload type is ICMPV6)   |   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    Type=2    |    Code=0    |         Checksum                | ICMPv6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=2
|       Next Hop Mtu = R1_VtepB_MTU                             | Code=0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|Ver=6  |Traffic Class |      Flow Label                       |   ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|payload length                |Next Hdr      | Hop Limit       |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|                                                              |   |
|           src-ipv6   =      H1_IPv6                           |Orig
|                                                              |IPv6
|                                                              |   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|                                                              |   |
|           dest-ipv6  =      H2_IPv6                           |   |
|                                                              |   |
|                                                              |   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| ~ Extension and Transport Headers and Applcation data    ~   | varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```
Figure 15. ICMPv6 error Complete Packet sent to H1 by VtepA

4.1.3.4. Both End-Point and Underlay are provisioned with IPv4

   Topology is mentioned in figure 16, with minor changes along with
   the legend, figure 17, provides the icmp PDU encap generated by R1.
   H1_IPv4 and H2_IPv4 are in different ipv4 subnets.

```
+----------+                        +----------+
|    H1    |                        |    H2    |
|          |                        |          |
|(H1_IPv4) |                        |(H2_IPv4) |
+----------+                        +----------+
     |                                   |
     |                                   |
+------------+   +----------+   +------------+
|(VtepA_IPv4)|   |          |   |(VtepB_IPv4)|
|   VtepA    |   |    R1    |   |   VtepB    |
|(VtepA_IPv4)|---| (R1_IPv4)|---|(VtepB_IPv4)|
+------------+   +----------+   +------------+
```

                  Figure 16. L3 Overlay
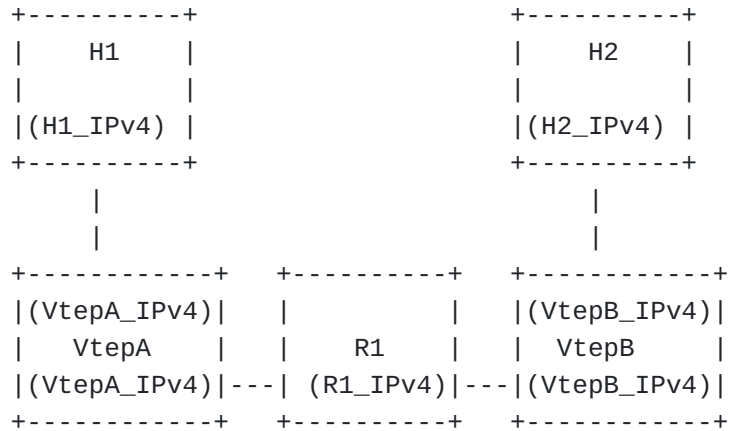
        LEGEND:
        MAC address : <Node_name>_MAC
        IPv4 address: <Node_name>_IPv4
        <Node_name> : node names in the above topology are
                      H1, VtepA, R1, VtepB, H2.
        VtepA, VtepB: Vxlan gateways to underlay network

```
  0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |    Type=3     |    Code=4     |           Checksum            | ICMP
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=3
 |    unused     |    Length     |   Next Hop Mtu = R1_VtepB_MTU | Code=4
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 | Ver=4|IHL=5   |  TOS          |        Total length          | ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |        Id                     |Flags| Fragment Offset        |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 | TTL           | Protocol=UDP  | Header Checksum              | IPv4
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+Header
 |          src-ip  : VtepA_IPv4                                |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |          dest-ip : H1_IPv4                                   |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |   Source UDP Port (ephemeral) |  Dest UDP Port = 4789 (Vxlan) |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
 |  Length                       |  Checksum                    |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 | | | | | | | | |      Reserved                                |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+8 bytes
 |Vxlan Network identifier (VNI)                 |  Reserved     |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 |             Inner Packet Dest-Mac = VtepB_MAC                |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |                               |   Inner Packet Src-Mac =     |inner
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+packet
 |                   VtepA_MAC                                  |eth hdr
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 | Inner Vlan if present         |Ethtype = 0X0800 (IPv4)       |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 | Ver=4|IHL=5   |  TOS          |        Total length          |  ^
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |        Id                     |Flags| Fragment Offset        |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 | TTL           | Protocol      | Header Checksum              | IPv4
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ hdr
 |          src-ip  : H1_IPv4                                   |  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
 |          dest-ip : H2_IPv4                                   |  v
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
 | ~ Optional and Transport Header and Application Payload ~    |varies
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```

 Figure 17. ICMP PDU generated by R1 towards VtepA

    R1 sends an icmp error directed towards VtepA. The icmp PDU is shown
    in figure 17. VtepA receives the packet with this icmp PDU and

relays it to H1 over ipv4 network. Figure 16, displays the packet
sent by VtepA to H1. All other references can be taken as it is from
Section 4.1.2.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|                    Dest-Mac = H1_MAC                         |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                              |        Src-Mac =              |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  eth
|                    VtepA_MAC                                 |header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Inner Vlan if present        |Ethtype = 0X0800 (IPv4)       |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5  |  TOS          |       Total length           |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|         Id                   |Flags| Fragment Offset        |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|   TTL        | Protocol=1    | Header Checksum              |IPv4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+Header
|         src-ip  : VtepA_IPv4                                 |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|         dest-ip : H1_IPv4                                    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|         Optional Header                                      |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|    Type=3     |   Code=4     |          Checksum            | ICMP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Type=3
|    unused     |   Length     |   Next Hop Mtu = R1_VtepB_MTU | Code=4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
| Ver=4|IHL=5  |  TOS          |       Total length           |  ^
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|         Id                   |Flags| Fragment Offset        |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|   TTL        | Protocol      | Header Checksum              |Orig
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+IPv4
|         src-ip  : H1_IPv4                                    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|         dest-ip : H2_IPv4                                    |  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|~ Optional and Transport Header and Application Payload    ~ | varies
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
```

Figure 18. Complete ICMP error Packet sent to H1 by VtepA

5. Inter-site MTU Propagation

```
                        +--+
                        |CE|
                        +--+
                         |
                       +----+
                +----|  PE |----+
        +---------+ |     +----+     | +---------+
+----+  |         +---+           +---+        | +----+
|NVE1|--|         |   |           |   |        |--|NVE3|
+----+  |---U1---|GW1|-----P------|GW3|---U2---|  +----+
        |         +---+           +---+        |
        |   NVO-1  |      WAN        |  NVO-2  |
        |         +---+           +---+        |
        |         |   |           |   |        |
+----+  |         |GW2|           |GW4|        | +----+
|NVE2|--|         +---+           +---+        |--|NVE4|
+----+  +---------+ |             | +---------+  +----+
                  +--------------+
```
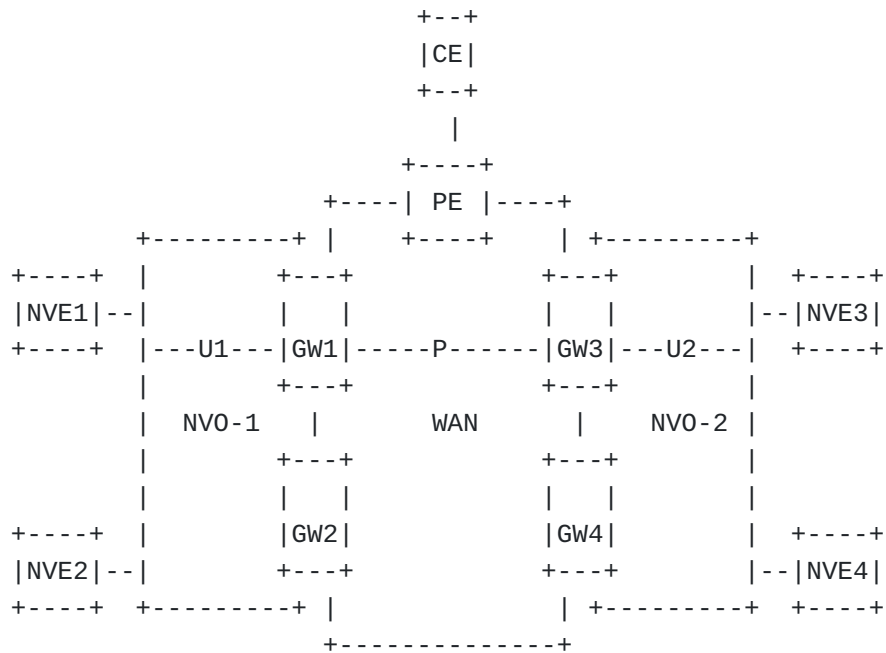
Figure 18. Datacenter/Site Interconnect Between Remote EVPN fabrics

This section specifically calls out the relay of icmp errors
generated by underlay in an intersite/interfabric connectivity
across EVPN-overlays. The reference diagram shown above is picked up
from [RFC9014].

The topology in the above diagram describes two disparate NVO
fabrics connected across WAN; leveraging an EVPN provisioned
overlay. Lets consider the interconnect as EVPN-Overlay over the WAN
network (between GW1/2 and GW3/4). Thus there is a multi-hop overlay
(tunnel) reachability between hosts in fabric(s) behind the edges
NVE1/2 and NVE3/4. There is a EVPN-Overlay tunnel between NVEs and
their respective gateways, i.e., between NVE1/2 and GW1/2 and
another one between NVE3/4 and GW3/4. There is an intersite connect
leveraging EVPN-Overlay, thus ensuring end to end connectivity
between NVEs across the WAN. The fabric in dataplane can be Vxlan,
MPLS, NVGRE, GENEVE, GUE, GPE etc.

The packet traversing between networks behind NVE1 to NVE3 shall
transit through three EVPN-Overlay tunnels. First one, between NVE1
and GW1; second one, between the WAN gateways GW1 and GW3 and the
third one, between GW3 and NVE3. There is an EVPN-Overlay handoff at
all the EVPN-tunnel end-points in the packet path, GW1 and GW2
respectively.

There is a possibility that the overlay encapsulated packet hits the
MTU blockage at one of the underlay routers, lets say, P3 in this
case. P3 generates icmp error targetted towards GW3 as the tunnel

end-point. GW3 should check the credentials of the original PDU,
carried in the icmp error and perform the route lookup. It's very
likely that the path to reach packet source (behind NVE1) is also
via the EVPN-Overlay tunnel from GW3 to GW1. The icmp error is
relayed back over the EVPN-Overlay construct towards GW3. In the
same flow GW1 should peek into the original PDU credentials to get
the reachability to the inner packet source. As luck may have it,
the packet source is reachable over the EVPN-overlay tunnel from GW1
to NVE1. It should go through the similar decap/re-encap as
mentioned in earlier sections. The EVIs at each stitching point may
be different, although ensuring that routes are exported between the
VNIs. The first-hop vtep towards the source i.e. NVE1 should perform
procedures mentioned in Section 4.1.2, to relay out the icmp packet
to the original source of the packet.

6. Same subnet Considerations

This section proposes propagation of icmp or icmpv6 error (specific
to MTU) at source Vtep to inner packet source, which is generated by
an underlay device for a case, when, inner packet source and
destination ipv4(or ipv6) addresses are in the same subnet.

The steps in section Section 4.1.2.3, elaborate on the check to be
performed, if the icmp error is carrying the original PDU
encapsulated with an L2VNI or L3VNI. In case it is an L2VNI, then
the possibility of the inner packet traffic being a "same subnet"
one is, very high. Hence Section 4.1.2.3, also talks about doing the
ethernet type check in the inner packet payload. If and only if,
it's ipv4 or ipv6, relay the icmp error back to the inner packet
source ip (or ipv6) address. Else, don't process the relay message
further. As the inner packet is a non layer-3 PDU, it does not makes
sense to relay back the icmp error.

7. Ecmp Considerations

Ecmp considerations are driven by the packet sent by the end host
application and the way it's leveraged.

To ensure "MTU propagation" via "icmpv6 error", is agnostic to ecmp
paths in a Vxlan network, there are few more consideration. In Vxlan
Gateway, the route look-up is done based on attributes carried in
packet generated by end point host. The packet generated can
potentially be from a tcp based end host application (although
should not be generalized).

Where as, for an intermediate node, (lets say, Spine node in Clos
topology) in underlay network the look ups are based on Outer Encap
(Vtep ip addresses and and UDP Header).

The packet traversing from site behind NE1 to NVE3 shall transit
through three EVPN-Overlay tunnels. First one, between NVE1 and GW1;
second one, between the WAN gateways GW1 and GW3 and the third one,
between GW3 and NVE3. There is an EVPN-Overlay handoff at all the
EVPN-tunnel end-points in the packet path, GW1 and GW2 respectively.
On another note, for an L2 gateway case, wherein Vxlan gateway (Vtep
Node) bridges (and not routes) host packets destined to same subnet
destination, MTU calculation SHOULD come into play only in the Spine
devices.

As a potential solution,the MTU values recieved over ECMP underlay
paths can be cached at the ingress Vteps. The Vtep MAY propagate/
relay the lowest of the all MTUs received across ECMP underlay
paths, to the end-host.

8. Security Considerations

This document inherits all the security considerations discussed in
[RFC1981] and [RFC1191].

9. IANA Considerations

TBD

10. Acknowledgements

Thanks to Vengada Prasad Govindan, Deepak Kumar, Matthew Bocci and
Rohit Mendiratta for providing the inputs.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, March 1997,
           <http://www.rfc-editor.org/rfc/rfc2119.txt>.

11.2. Informative References

[I-D.draft-gross-geneve]
           Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, G.,
           Agarwal, P., Duda, C., Dutt, D., and J. Hudson, "Geneve:
           Generic Network Virtualization Encapsulation", Work in
           Progress, Internet-Draft, draft-gross-geneve-02, 25
           October 2015, <http://www.ietf.org/internet-drafts/draft-
           gross-geneve-02.txt>.

[I-D.draft-ietf-nvo3-gue] Herbert, T., Yong, L., and O. Zia,
           "Generic Protocol Extension for VXLAN", Work in Progress,
           Internet-Draft, draft-ietf-nvo3-gue-03, 6 March 2015,

          <http://www.ietf.org/internet-drafts/draft-ietf-nvo3-
          gue-03.txt>.

[I-D.draft-ietf-nvo3-vxlan-gpe]
          Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F.,
          Smith, M., Agarwal, P., Yong, L., Xu, X., Elzur, U., and
          D. Melman, "Generic Protocol Extension for VXLAN", Work
          in Progress, Internet-Draft, draft-ietf-nvo3-vxlan-
          gpe-02, 1 May 2015, <http://www.ietf.org/internet-drafts/
          draft-ietf-nvo3-vxlan-gpe-02.txt>.

[I-D.nordmark-nvo3-transcending-traceroute] Nordmark, E., Appanna,
          C., and A. Lo, "Layer-Transcending Traceroute for Overlay
          Networks like VXLAN", Work in Progress, Internet-Draft,
          draft-nordmark-nvo3-transcending-traceroute-02, 4 March
          2015, <http://www.ietf.org/internet-drafts/draft-
          nordmark-nvo3-transcending-traceroute-02.txt>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
          1 November 1990, <http://www.rfc-editor.org/rfc/
          rfc1191.txt>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU
          Discovery for IP version 6", RFC 1981, August 1996,
          <http://www.rfc-editor.org/rfc/rfc1981.txt>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control
          Message Protocol (icmpv6) for the Internet Protocol
          Version 6 (IPv6) Specification", RFC 4443, March 2006,
          <http://www.rfc-editor.org/rfc/rfc4443.txt>.

[RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-
          Network Tunneling", RFC 9014, April 2006, <http://
          www.rfc-editor.org/rfc/rfc9014.txt>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
          Discovery", RFC 4821, March 2007, <http://www.rfc-
          editor.org/rfc/rfc4821.txt>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro,
          "Extended ICMP to Support Multi-Part Messages", RFC 4884,
          April 2007, <http://www.rfc-editor.org/rfc/rfc4884.txt>.

[RFC7348]
          Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger,
          L., Sridhar, T., Bursell, M., and C. Wright, "Virtual
          eXtensible Local Area Network (VXLAN): A Framework for
          Overlaying Virtualized Layer 2 Networks over Layer 3
          Networks", RFC 7348, August 2014, <http://www.rfc-
          editor.org/rfc/rfc7348.txt>.

[RFC7637] Yang, S. and M. Garg, "Network Virtualization Using
           Generic Routing Encapsulation", RFC 7637, September 2015,
           <http://www.rfc-editor.org/rfc/rfc7637.txt>.

[RFC9014] Rabadan, J., Sathappan, S., Henderickx, W., Sajassi, A.,
           and W. Drake, "Interconnect Solution for Ethernet VPN
           (EVPN) Overlay Networks", RFC 9014, May 2021, <http://
           www.rfc-editor.org/rfc/rfc9014.txt>.

Authors' Addresses

Saumya Dikshit
Aruba Networks, HPE
Mahadevpura
Bangalore 560 048
Karnataka
India

Email: saumya.dikshit@hpe.com

Vinayak Joshi
Aruba Networks, HPE
Mahadevpura
Bangalore 560 048
Karnataka
India

Email: vinayak.joshi@hpe.com

A. Sujeet Nayak
Cisco
Cessna Business Park
Bangalore 560 087
Karnataka
India

Email: sua@cisco.com