Layer-Transcending Traceroute for Overlay Networks like VXLAN
draft-nordmark-nvo3-transcending-traceroute-03

Abstract

   Tools like traceroute have been very valuable for the operation of
   the Internet. Part of that value comes from being able to display
   information about routers and paths over which the user of the tool
   has no control, but the traceroute output can be passed along to
   someone else that can further investigate or fix the problem.

   In overlay networks such as VXLAN and NVGRE the prevailing view is
   that since the overlay network has no control of the underlay there
   needs to be special tools and agreements to enable extracting traces
   from the underlay. We argue that enabling visibility into the
   underlay and using existing tools like traceroute has been overlooked
   and would add value in many deployments of overlay networks.

   This document specifies an approach that can be used to make
   traceroute transcend layers of encapsulation including details for
   how to apply this to VXLAN. The technique can be applied to other
   encapsulations used for overlay networks. It can also be implemented
   using current commercial silicon.

Status of this Memo

   This Internet-Draft will expire on January 2, 2017.

Copyright Notice

Table of Contents

1. Introduction

   Tools like traceroute have been very valuable for the operation of
   the Internet. Part of that value comes from being able to display
   information about routers and paths over which the user of the tool
   has no control, but the traceroute output can be passed along to
   someone else that can further investigate or fix the problem. The
   output of traceroute can be included in an email or a trouble ticket
   to report the problem. This provide a lot more information than the
   mere indication that A can't communicate with B, in particular when
   the failures are transient. The ping tool provides some of the same
   benefits in being able to return ICMP errors such as host unreachable
   messages.

   This document shows how those tools can be used to gather information
   for both the overlay and underlay parts of an end-to-end path by
   providing the option to have some packets use a uniform time-to-live
   (ttl) model for the tunnels, and associated ICMP error handling.
   These changes are limited to the tunnel ingress and egress points.

   The desire to make traceroute provide useful information for overlay
   network is not an argument against also using a layered approach for
   OAM as specified in e.g., [I-D.tissa-lime-yang-oam-model]. Such
   approaches are quite appropriate for continuos monitoring at
   different layers and across different domains. A layer transcending
   traceroute complements the ability to do layered and/or continuos
   monitoring.

   The traceroute tool relies on receiving ICMP errors [RFC0792] in
   combination with using different IP time-to-live values. That
   results in the packet making it further and further towards the
   destination with ICMP ttl exceeded errors being received from each
   hop. That provides the user the working path even if the packets are
   black holed eventually, and also provides any errors like ICMP host
   unreachable. The fundamental assumption is that the ttl is
   decremented for each hop and that the resulting ICMP ttl exceeded
   errors are delivered back to the host.

   When some encapsulation is used to tunnel packets there is an
   architectural question how those tunnels should be viewed from the
   rest of the network. Different models were described first for
   diffserv in [RFC2983] and then applied to MPLS in [RFC3270] and
   expanded to MPLS ttl handling in [RFC3443] and those models apply to
   other forms of direct or indirect IP in IP tunnels. Those RFCs
   define two models for ttl that are of interest to us:

   o A pipe model, where the tunnel is invisible to the rest of the
      network in that it looks like a direct connection between the

tunnel ingress and egress.

o A uniform model, where the ttl decrements uniformly for hops
   outside and inside the tunnel.

The tunneling mechanisms discussed in NVO3 (such as VXLAN [RFC7348],
NVGRE [I-D.sridharan-virtualization-nvgre], GENEVE
[I-D.gross-geneve], and GUE [I-D.herbert-gue]), have either been
specified to provide the pipe model of a tunnel or are silent on the
setting of the outer ttl. Those protocols can be extended to have an
optional uniform tunnel model when the payload is IP, following the
same model as in [RFC3443]. Note that these encapsulations carry
Ethernet frames hence are not even aware that the payload is IP.
However, IP is the bulk of what is carried over such tunnels and the
ingress NVE can inspect the IP part of the Ethernet frame.

However, for general application traffic the pipe model is fine and
might even be expected by some applications. In general, when the
source and destination IP are in the same IP subnet the ttl should
not be decremented. Thus it makes sense to have a way to selectively
enable the uniform model perhaps based on some method to identify
packets associated with traceroute or some marker in the packet
itself that the traceroute tool can set.

2. **Solution Overview**

The pieces needed to accomplish this are:

o One or more ways to select the uniform model packets at the tunnel
   ingress.

o Tunnel ingress copying out the original ttl from a selected packet
   to the outer IP header, and then doing a check and decrement of
   that ttl.

o If that ttl check results in ttl expiry at the tunnel ingress,
   then deliver an ICMP ttl exceeded packet back to the host.

o A mechanism by which the tunnel egress knows which packets should
   have uniform model, for instance a bit in the encapsulation
   header.

o The tunnel egress copying in the ttl (for identified packets) from
   the outer header to the inner IP header, then doing a check and
   decrement of that ttl.

o If ttl check results in ttl expiry at the tunnel egress, then
   deliver an ICMP error back to the original host (or, perhaps
   better, to tunnel ingress the same way as underlay routers do).

o IP routers in the underlay will deliver any ICMP errors to the
   source IP address of the packet. For tunneled packets that will
   be the tunnel ingress. Hence the tunnel ingress needs to be able
   to take such ICMP errors and form corresponding ICMP errors that
   are sent back to the host. The requirement in [RFC1812] ensures
   that the ICMP errors will contain enough headers to form such an
   ICMP error. It has been noted that there are routers in the
   Internet which decades later fail to conform to that aspect of
   [RFC1812].

The idea to reflect (some) ICMP errors from inside a tunnel back to
the original source goes back to IPv6 in IPv4 encapsulation as
specified in [RFC1933] and [RFC2473]. However, those drafts did not
advocate using a uniform ttl model for the tunnels but did handle
ICMP packet too big and other unreachable messages. Those drafts
specify how to reflect ICMP errors received from underlay routers to
ICMP errors sent to the original host. The addition of handling ICMP
ttl exceeded errors for uniform tunnel model is straight forward.

The information carried in the ICMP errors are quite limited - the
original packet plus an ICMP type and code. However, there are
extension mechanisms specified in [RFC4884] and used for MPLS in
[RFC4950] which include TLVs with additional information. If there
are additional information to include for overlay networks that
information could be added by defining new ICMP Extensions Objects
based on [RFC4884]. An example of such an extension for ECMP
information is included in this document.

3. Goals and Requirements

The following goals and requirements apply:

o No changes needed in the underlay.

o Optional changes on the decapsulating end.

o ECMP friendly. If the underlay employs equal cost multipath
   routing then one should be able to use this mechanism to trace the
   same path as a given TCP or UDP flow is using. In addition, one
   should be able to explore different ECMP paths by varying the IP
   addresses and port numbers in the packets originated by traceroute
   on the host.

   o Provide output which makes it possible to compare a regular
      overlay traceroute with the layer-transcending output.

[4](#). **Definition Of Terms**

   The keywords "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](#)].

   The terminology such as NVE, and TS are used as specified in
   [[RFC7365](#)]:

   o Network Virtualization Edge (NVE): An NVE is the network entity
      that sits at the edge of an underlay network and implements L2
      and/or L3 network virtualization functions.

   o Tenant System (TS): A physical or virtual system that can play the
      role of a host or a forwarding element such as a router, switch,
      firewall, etc.

   o Virtual Access Points (VAPs): A logical connection point on the
      NVE for connecting a Tenant System to a virtual network.

   o Virtual Network (VN): A VN is a logical abstraction of a physical
      network that provides L2 or L3 network services to a set of Tenant
      Systems.

   o Virtual Network Context (VN Context) Identifier: Field in an
      overlay encapsulation header that identifies the specific VN the
      packet belongs to.

   We use the VTEP term in [[RFC7348](#)] as synonymous with NVE, and VNI as
   synonymous to VN Context Identifier.

[5](#). **Example Topologies**

   The following example topologies illustrate different cases where we
   want a tracing capability. The examples are for overlay technologies
   such as VXLAN which provide a layer 2 overlay on IP. The cases for
   layer 3 overlay on top of IP are simpler and not shown in this
   document.

   The VXLAN term VTEP is used as synonymous to NVO3's NVE term.

```
      -----------                   -----------
     |    H1   |                   |    H2   |
     | 1.0.1.1 |                   | 1.0.1.2 |
     |         |                   |         |
      -----------                   -----------
          |                             |
          |                             |
      -----------    -----------    -----------
     |  VtepA  |    |    R1   |    |  VtepB  |
     | 2.0.1.1 | --| 2.0.1.2 |    | 2.0.2.1 |
     |         |    | 2.0.2.2 |--|          |
      -----------    -----------    -----------
```
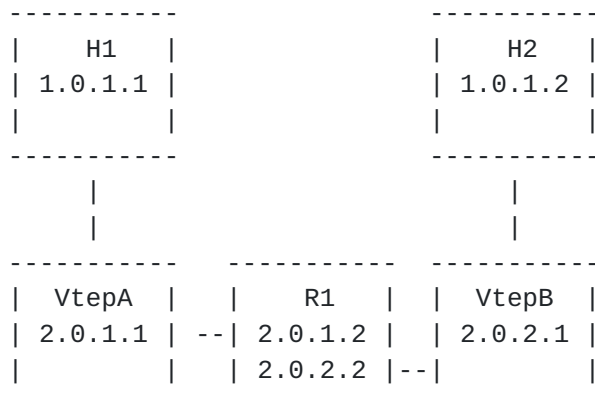
                    Simple L2 overlay

The figure above shows two hosts connected using an underlay which
provides a layer two service. Thus H1 and H2 are in the same subnet
and unaware of the existence of the underlay. Thus a normal ping or
traceroute would not be able to provide any information about the
nature of a failure; either packets get through or they do not. When
the packets get through traceroute would output something like:

traceroute to 1.0.1.2 (1.0.1.2), 30 hops max, 60 byte packets
 1 1.0.2.1 (1.0.2.1) 1.104 ms 1.235 ms 1.729 ms

In this case it would be desirable to be able to traceroute from H1
to H2 (and vice versa) and observe VtepA, R1, VtepB and H2. Thus in
the case of packets getting through traceroute would output:

traceroute to 1.0.1.2 (1.0.1.2), 30 hops max, 60 byte packets
 1 2.0.1.1 (2.0.1.1) 1.104 ms 1.235 ms 1.729 ms
 2 2.0.1.2 (2.0.1.2) 2.106 ms 2.007 ms 2.156 ms
 3 2.0.2.1 (2.0.2.1) 35.034 ms 24.490 ms 21.626 ms
 4 1.0.1.2 (1.0.1.2) 40.830 ms 44.694 ms 75.620 ms

Note that the underlay and overlay might exist in completely separate
addressing domains. Thus H1 might not be able to reach any of the
underlay addresses. And the underlay IP addresses might overlap the
overlay IP addresses. For example, it would be completely valid to
see e.g. VtepA having the same IP address as H1. The user of this
tool need to understand that the utility of the traceroute output is
to get information to determine whether the issue is in the underlay
or overlay, and be able to pass the underlay information to the
operator of the underlay.

In overlay networks without any ARP/ND optimizations ARP/ND packets
would be flooded between the tunnel endpoints. Thus if there is some
communication failure between H1 and H2, then H1 above might not have

an ARP entry for H2. This results in traceroute not being able to
output any data. This implies that in order to use traceroute to
trouble shoot the issue one would need some workaround, such as
installing some temporary ARP entries on the hosts.

```
 -----------                   -----------  -----------  -----------
|    H1     |                 |    R2     |  |    R3    |  |    H4     |
|  1.0.1.1  |                 |  1.0.2.2  |--|  1.0.2.3 |  |           |
|           |                 |  1.0.1.2  |  |  1.0.3.3 |--|  1.0.3.4  |
 -----------                   -----------  -----------  -----------
      |                              |
      |                              |
 -----------   -----------  -----------
|  VtepA    |  |    R1     |  |  VtepB   |
|  2.0.1.1  | --|  2.0.1.2 |  |  2.0.2.1 |
|           |  |  2.0.2.2  |--|          |
 -----------   -----------  -----------
```
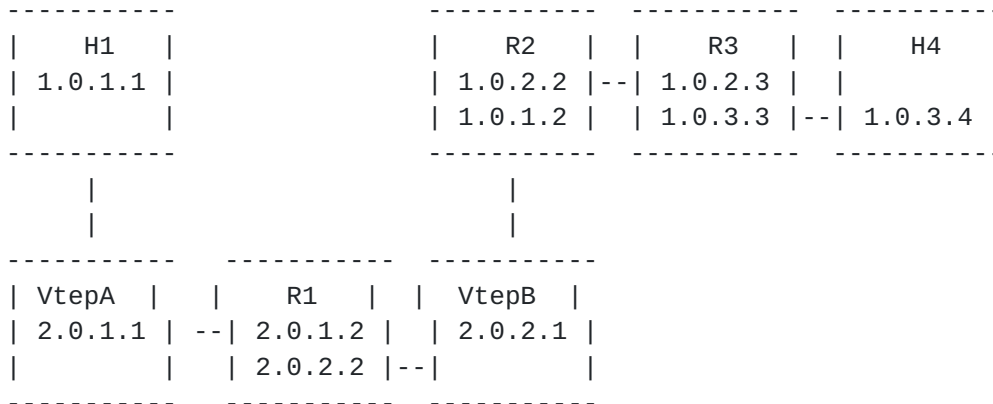
                L2 overlay as part of larger network

The figure above has a overlay router the nexthop as seen by H1. In
this case a normal overlay traceroute would be able to display the
overlay path i.e.

traceroute to H4, 30 hops max, 60 byte packets
 1 R2
 2 R3
 3 H4

The layer-transcending traceroute would show the combination of the
underlay and overlay paths i.e.,

traceroute to H4, 30 hops max, 60 byte packets
 1 VtepA
 2 R1
 3 VtepB
 4 R2
 5 R3
 6 H4

```
 -----------                -------------------              -----------
|    H1     |              |        R5         |            |    H6     |
|  1.0.1.1  |              |                   |            |           |
|           |              |  1.0.1.2 1.0.5.5  |            |  1.0.5.6  |
 -----------               |-----------------  |             -----------
     |                     |    |      |    |                     |
     |                     |    |      |    |                     |
 -----------  -----------  |-----------------  |  -----------  -----------
| VtepA    | |    R1     | |  VtepB   VtepC   | |    R6     | |  VtepD   |
| 2.0.1.1  |-|  2.0.1.2  | |  2.0.2.1 3.0.1.1 |-|  3.0.1.2  | |          |
|          | |  2.0.2.2  |-|                  | |  3.0.2.2  |-|  3.0.3.1 |
 -----------  -----------   -------------------  -----------  -----------
```
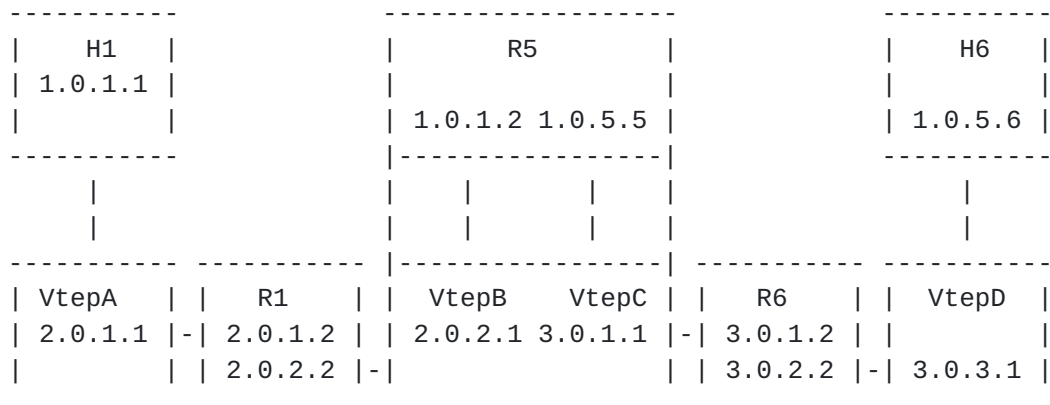
                       Multiple L2 overlays in path

The figure above has multiple overlay network segments, that are
connected in one router which provides the tunnel endpoints for both
overlay segments plus routing for the overlay. A more general
picture would be to have an overlay routed path between the two NVEs
e.g., VtepB and VtepC connected to different routers in the overlay.
However, such a drawing in ASCII art doesn't fit on the page.

An normal overlay traceroute in the above topology would show the
overlay router i.e.,

traceroute to H6, 30 hops max, 60 byte packets
 1 R5
 2 H6

The layer-transcending traceroute would show the combination of the
underlay and overlay paths i.e.,

traceroute to H6, 30 hops max, 60 byte packets
 1 VtepA
 2 R1
 3 VtepB
 4 R5
 5 VtepC
 6 R6
 7 VtepD
 8 H6

Note that the R3 device, which include VtepB and VtepC, appears as
three hops in the traceroute output. That is needed to be able to
correlate the output with the overlay output which has R3. That
correlation would be hard if the R3 device only appeared as VtepB in
the LTTON output. The three-hop representation also stays invariant
whether or not the NVEs and overlay router are implemented by a

single device or multiple devices.

6. Controlling and selecting ttl behavior

The network admin needs to be able to control who can use the layer
transcending traceroute, since the operator might not want to
disclose the underlay topology to all its users all the time. There
are different approaches for this such as designating particular
ports (Virtual Access Points in NVO3 terminology) on a NVE to have
uniform ttl tunnel model. We have found it useful to be able to
enable this capability on a per port and/or virtual network basis, in
addition to having a global setting per NVE.

When enabled on the NVEs the user on the TS needs to be able to
control which traffic is subject to which tunnel mode. The normal
traffic would use the pipe ttl tunnel model and only explicit trace
applications are likely to want to use the uniform ttl tunnel model.
Hence it makes sense to use some marker in the packets sent by the TS
to select those packets for uniform model on the NVE. Such a
mechanism should usable so that the user can perform both a regular
traceroute and a LTTON.

Potentially different fields in the packets originated by traceroute
on the TS can be used to mark the packets for uniform ttl tunnel
model. However, many of those fields such as source and destination
port numbers and protocol might be used in hashing for ECMP. The
marking that can be used without impacting ECMP is the DSCP field in
the packet. That field can be set with an option (--tos) in at least
some existing traceroute implementations.

Note that when DSCP is used for such marking it is a configured
choice subject to agreement between the operator of the TS and NVE.
The matching on the NVE should ignore the ECN bits as to not
interfere with ECN.

However, the DSCP value used in the overlay might have an impact on
the forwarding of the packets. In such a case one can use an
alternative selector such as the UDP source port number. That has
the downside of affecting the has values used for ECMP and link
aggregation port selection.

7. Introducing a ttl copyin flag in the encapsulation header

When this approach is applied to VXLAN [RFC7348] the decapsulating
NVE has to be able to identify packets that have to be processed in
the uniform ttl tunnel model way. For that purpose we define a new

flag which is sent by the encapsulating NVE on selected packets, and
is used by the decapsulating NVE to perform the ttl copyin, decrement
and check.

In addition to the one I-flag defined in [RFC7348] we define a new
T-flag to capture this the trace behavior at the decapsulating tunnel
endpoint.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |R|R|R|R|I|R|R|T|            Reserved                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                VXLAN Network Identifier (VNI) |   Reserved    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

New fields:

T-flag: When set indicates that decapsulator should take the
                outer ttl and copy it to the inner ttl, and then check
                and decrement the resulting ttl.

8. Encapsulation Behavior

If the uniform ttl model is enabled for the input, and the received
naked packet matches the selector, then the ingress NVE will perform
these additional operations as part of encapsulating an IPv4 or IPv6
packet:

o Examine the IPv4 TTL (or IPv6 hopcount, respectively) on receipt
   and if 1 or less, then drop the packet and send an ICMPv4 (or
   ICMPv6) ttl exceeded back to the original host. Since the NVE is
   operating on a L2 packet, it might not have any layer 3 interfaces
   or routes for the originating host. Thus it sends the packet back
   to the source L2 address of the packet back out the ingress port -
   without any IP address lookup.

o If ttl did not expire, then decrement the above ttl/hopcount and
   place it in the outer IP header. Encapsulate and send the packet
   as normal.

o If some other errors prevent sending the packet (such as unknown
   VN Context Id, no flood list configured), then the NVE SHOULD send
   an ICMP host unreachable back to the host.

The ingress NVE will receive ICMP errors from underlay routers and
the egress NVE; whether due to ttl exceeded or underlay issues such

as host unreachable, or packet too big errors. The NVE should take
such errors, and in addition to any local syslog etc, generate an
ICMP error sent back to the host. The principle for this is
specified in [RFC1933] and [RFC2473]. Just like in those
specifications, for the inner and outer IP header could be off
different version. A common case of that might be an IPv6 overlay
with an IPv4 underlay. That case requires some changes in the ICMP
type and code values in addition to recreating the packets. The
place where LTTON differs from those specifications is that there is
an NVO3 header and (for L2 over L3) and L2 header in the packet.

The figures below show an example of ICMP header re-generation at
VtepA for the case of IPv6 overlay with IPv4 underlay. The case of
IPv4 over IPv4 is similar and simpler since the ICMP header is the
same for both overlay and underlay. The example uses VXLAN
encapsulation to provide the concrete details, but the approach
applies to other NVO3 proposals.

```
              +--------------+
              | IPv4 Header  |
              | src = R1     |
              | dst = VtepA  |
              +--------------+
              |    ICMPv4    |
              |    Header    |
              |   type = X   |
              |   code = Y   |
    - -       +--------------+
              | IPv4 Header  |
              | src = VtepA  |
     IPv4     | dst = VtepB  |
              +--------------+
    Packet    |     UDP      |
              | dst = VXLAN  |
      in      +--------------+
              |   Ethernet   |
    Error     | DA = H2 mac  |
              | SA = H1 mac  |
              +--------------+   - -
              |     IPv6     |
              | src = H1 ipv6|
              | dst = H2 ipv6|   Original IPv6
              +--------------+   Packet.
              |  Transport   |   Used to
              |    Header     |   generate an
              +--------------+   ICMPv6
              |              |   error message
              ~      Data     ~   back to the source.
              |              |
    - -       +--------------+   - -
```
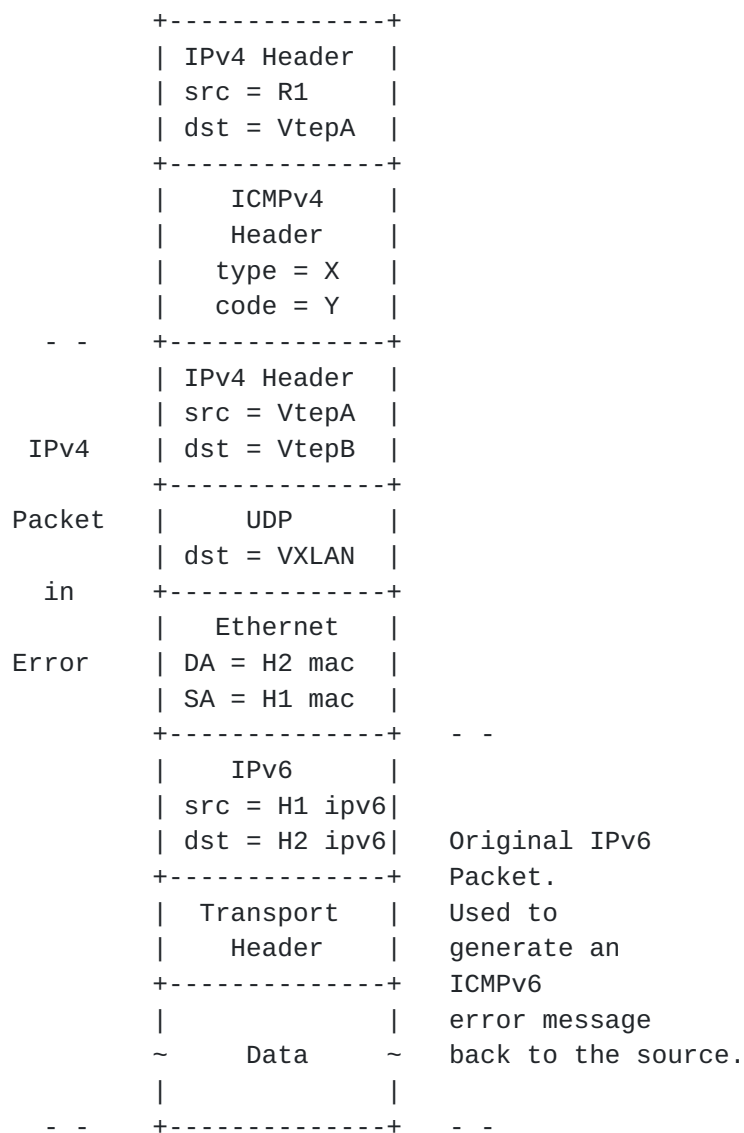
            ICMPv4 Error Message Returned to Encapsulating Node

   The above underlay ICMPv4 is used to form an overlay ICMPv6 packet by
   extracting the Ethernet DA from the inner Ethernet SA, and forming an
   IPv6 header where the source address is based on the source address
   of the ICMPv4 error. The ICMPv6 type and code values are set based
   on the ICMPv4 type and code values.

```
                +--------------+
                |   Ethernet   |
                | DA = H1 mac  |   From ICMPv4 packet
                | SA = VtepA   |   in error
                +--------------+
                | IPv6 Header  |
                | src = ::R1   |   96 zeros + IPv4 address
                | dst = H1 ipv6|
                +--------------+
                |    ICMPv6    |
                |    Header    |
                |   type = X'  |   Type and code mapped
                |   code = Y'  |   from v4 to v6 values
      - -       +--------------+   - -
                |     IPv6     |
      IPv6      | src = H1 ipv6|
                | dst = H2 ipv6|   Unmodified from
     Packet     +--------------+   ICMPv4 error
                |  Transport   |
       in       |    Header    |
                +--------------+
     Error      |              |
                ~      Data     ~
                |              |
      - -       +--------------+   - -
```

           Generated ICMPv6 Error Message for Overlay Source

   In the case of IPv6 over IPv4 the above example setting of the IPv6
   source address results in this type of traceroute output:

   traceroute to 2000:0:0:40::2, 30 hops max, 80 byte packets
    1 ::2.0.1.1 (::2.0.1.1) 1.231 ms 1.004 ms 1.126 ms
    2 ::2.0.1.2 (::2.0.1.2) 1.994 ms 2.301 ms 2.016 ms
    3 ::2.0.2.1 (::2.0.2.1) 18.846 ms 30.582 ms 19.776 ms
    4 2000:0:0:40::2 (2000:0:0:40::2) 48.964 ms 60.131 ms 53.895 ms

9. Decapsulating Behavior

   If this uniform ttl model is enabled on the decapsulating NVE, and
   the overlay header indicates that uniform ttl model applies (the
   T-bit in the case of VXLAN), then the NVE will perform these
   additional operations as part of decapsulating a packet where the
   inner packet is an IPv4 or IPv6 packet:

   o Examine the outer IPv4 TTL (or outer IPv6 hopcount, respectively)
      on receipt and if 1 or less, then drop the packet and send an

outer ICMPv4 (or ICMPv6) ttl exceeded back to the source of the
outer packet i.e., the ingress NVE. This ICMP packet should look
the same as an ICMP error generated by an underlay router, and the
requirement in [RFC1812] on the size of the packet in error
applies.

o If ttl did not expire, then decrement the above ttl/hopcount and
   place it in the inner IP header. If the inner IP header is IPv4
   then update the IPv4 header checksum. Then decapsulate and send
   the packet as for other decapsulated packets.

o If some other errors prevent sending the packet (such as unknown
   VN Context Id), then the NVE SHOULD send an ICMP host unreachable
   instead of a ttl exceeded error.

10. Other ICMP errors

The technique for selecting ttl behavior specified in this draft can
also be used to trigger other ICMPv4 and ICMPv6 errors. For example,
[RFC1933] specifies how ICMP packet too big from underlay routers can
be used to report over ICMP packet too big errors to the original
source. Other errors that are more specific to the overlay protocol
might also be useful, such as not being able to find a VNI ID for the
incoming port,vlan, or not being able to flood the packet if the
packet is a Broadcast, Unknown unicast, or Multicast packet.

11. Downstream Egress Paths Object

The Downstream Egress Paths Object MAY be appended to the ICMP Time
Exceeded and Destination Unreachable messages. A single instance of
the Downstream Egress Paths Object represents the egress paths at the
router that sends the ICMP message. The Downstream Egress Paths
Object must be preceded by an ICMP Extension Structure Header and an
ICMP Object Header. Both are defined in [RFC4884]. The format
follows closely [RFC4379] with some generalizations for Multipath
types.

   Class-Num = TBA by IANA, Downstream Egress Paths Class

   C-Type = 1.

If the replying router is the destination of the echo request, then a
Downstream Egress Paths Object SHOULD NOT be included in the ICMP
Error message. Otherwise the replying router MAY append a Downstream
Egress Paths Object for all interfaces over which the echo request
packet could be forwarded.

The Object Length is K*N + M*N, where M is the Multipath Length for
each egress path, M may not be the same for different paths. Values
for K are found in the description of Address Type below.

The Downstream Egress Paths Object has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path-1      MTU              | Address Type | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Downstream IP Address (4 or 16 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Downstream Interface Address (4 or 16 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MultipathType |      Multipath Length      | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                             .
.                   (Multipath Information)                   .
.                                                             .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                             ~
~                                                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path-N      MTU              | Address Type | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Downstream IP Address (4 or 16 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Downstream Interface Address (4 or 16 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MultipathType |      Multipath Length      | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                             .
.                   (Multipath Information)                   .
.                                                             .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Downstream Egress Paths Object

Maximum Transmission Unit (MTU):
   The MTU is the size in octets of the largest IP frame that fits on
   the downstream interface.

Address Type:
   The Address Type indicates if the interface is numbered or
   unnumbered. It also determines the length of the Downstream IP
   Address and Downstream Interface fields. The resulting total for
   the initial part of the one path of the downstream Egress Paths

Object is listed in the table below as "K Octets".

The Address Type is set to one of the following values:

```
Type #        Address Type            K Octets
------        ------------            --------
  1           IPv4 Numbered              16
  2           IPv4 Unnumbered            16
  3           IPv6 Numbered              40
  4           IPv6 Unnumbered            28
```

Downstream IP Address and Downstream Interface Address:
  IPv4 addresses and interface indices are encoded in 4 octets; IPv6
  addresses are encoded in 16 octets.

  If the interface to the downstream router has a unique IP address
  (e.g., it is numbered and not a LAG), then the Address Type MUST
  be set to IPv4 or IPv6, the Downstream IP Address MUST be set to
  either the downstream router's Router ID or the interface address
  of the downstream router, and the Downstream Interface Address
  MUST be set to the downstream router's interface address.

  If the interface to the downstream router does not have a unique
  IP address (e.g., it is is unnumbered or a LAG), the Address Type
  MUST be IPv4 Unnumbered or IPv6 Unnumbered, the Downstream IP
  Address MUST be the downstream router's Router ID or the interface
  address of the downstream router, and the Downstream Interface
  Address MUST be set to the index assigned by the upstream router
  to the interface.

Multipath Type:
  The following Multipath Types are defined:

```
Key   Type                Multipath Information
---   ----------------    ---------------------
 0    no multipath        Empty (Multipath Length = 0)
 1    MAC SA/DA           Inner MAC in tunnel payload
 2    IP Src/Dest         Inner IP src/dest in tunnel payload
 3    L4 src port         L4 src ports in tunnel payload
 4    L4 src port range   low/high L4 src port pairs
```

  Type 0 indicates that all packets will be forwarded out this one
  interface.

Types 1 through 4 specify that the supplied Multipath Information
will serve to exercise this path.

Multipath Length:
The length in octets of the Multipath Information.

Multipath Information:
The Multipath Information encodes L4 source ports that will
exercise this path. The Multipath Information depends on the
Multipath Type. The contents of the field are shown in the table
above. For Type 4, ranges indicated by L4 source port pairs MUST
NOT overlap and MUST be in ascending sequence.

12. Security Considerations

The considerations in [I-D.ietf-nvo3-security-requirements] apply.

In addition, the use of the uniform ttl tunnel model will result in
ICMP errors being generated by underlay routers and consumed by NVEs.
That resents an attack vector which does not exist in a pipe ttl
tunnel model. However, ICMP errors should be rate limited [RFC1812].
Implementations should also take appropriate measures in rate
limiting the input rate for ICMP errors that are processed by limited
CPU resources.

Some implementations might handle the trace packets (with uniform ttl
model) in software while the pipe ttl model packets can be handled in
hardware. In such a case the implementation should have mechanisms
to avoid starvation of limited CPU resources due to these packets.

13. IANA Considerations

TBD

14. Acknowledgements

The authors acknowledge the helpful comments from David Black and
Diego Garcia del Rio.

15. References

15.1. **Normative References**

[RFC0792]   Postel, J., "Internet Control Message Protocol", STD 5,
            RFC 792, DOI 10.17487/RFC0792, September 1981,
            <http://www.rfc-editor.org/info/rfc792>.

[RFC1812]   Baker, F., Ed., "Requirements for IP Version 4 Routers",
            RFC 1812, DOI 10.17487/RFC1812, June 1995,
            <http://www.rfc-editor.org/info/rfc1812>.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/
            RFC2119, March 1997,
            <http://www.rfc-editor.org/info/rfc2119>.

[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, DOI 10.17487/RFC7348, August 2014,
            <http://www.rfc-editor.org/info/rfc7348>.

[RFC7365]   Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y.
            Rekhter, "Framework for Data Center (DC) Network
            Virtualization", RFC 7365, DOI 10.17487/RFC7365,
            October 2014, <http://www.rfc-editor.org/info/rfc7365>.

15.2. **Informative References**

[I-D.gross-geneve]
            Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I.,
            Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve:
            Generic Network Virtualization Encapsulation",
            draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
            Herbert, T., Yong, L., and O. Zia, "Generic UDP
            Encapsulation", draft-herbert-gue-03 (work in progress),
            March 2015.

[I-D.ietf-nvo3-security-requirements]
            Hartman, S., Zhang, D., Wasserman, M., Qiang, Z., and M.
            Zhang, "Security Requirements of NVO3",
            draft-ietf-nvo3-security-requirements-07 (work in
            progress), June 2016.

[I-D.sridharan-virtualization-nvgre]
            Garg, P. and Y. Wang, "NVGRE: Network Virtualization using

               Generic Routing Encapsulation",
               draft-sridharan-virtualization-nvgre-08 (work in
               progress), April 2015.

     [I-D.tissa-lime-yang-oam-model]
               Senevirathne, T., Finn, N., Kumar, D., Salam, S., Wu, Q.,
               and Z. Wang, "Generic YANG Data Model for Operations,
               Administration, and Maintenance (OAM)",
               draft-tissa-lime-yang-oam-model-06 (work in progress),
               August 2015.

     [RFC1933] Gilligan, R. and E. Nordmark, "Transition Mechanisms for
               IPv6 Hosts and Routers", RFC 1933, DOI 10.17487/RFC1933,
               April 1996, <http://www.rfc-editor.org/info/rfc1933>.

     [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in
               IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,
               December 1998, <http://www.rfc-editor.org/info/rfc2473>.

     [RFC2983] Black, D., "Differentiated Services and Tunnels",
               RFC 2983, DOI 10.17487/RFC2983, October 2000,
               <http://www.rfc-editor.org/info/rfc2983>.

     [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen,
               P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-
               Protocol Label Switching (MPLS) Support of Differentiated
               Services", RFC 3270, DOI 10.17487/RFC3270, May 2002,
               <http://www.rfc-editor.org/info/rfc3270>.

     [RFC3443] Agarwal, P. and B. Akyol, "Time To Live (TTL) Processing
               in Multi-Protocol Label Switching (MPLS) Networks",
               RFC 3443, DOI 10.17487/RFC3443, January 2003,
               <http://www.rfc-editor.org/info/rfc3443>.

     [RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol
               Label Switched (MPLS) Data Plane Failures", RFC 4379,
               DOI 10.17487/RFC4379, February 2006,
               <http://www.rfc-editor.org/info/rfc4379>.

     [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro,
               "Extended ICMP to Support Multi-Part Messages", RFC 4884,
               DOI 10.17487/RFC4884, April 2007,
               <http://www.rfc-editor.org/info/rfc4884>.

     [RFC4950] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "ICMP
               Extensions for Multiprotocol Label Switching", RFC 4950,
               DOI 10.17487/RFC4950, August 2007,
               <http://www.rfc-editor.org/info/rfc4950>.

Authors' Addresses

    Erik Nordmark
    Arista Networks
    Santa Clara, CA
    USA

    Email: nordmark@arista.com

    Chandra Appanna
    Arista Networks
    Santa Clara, CA
    USA

    Email: achandra@arista.com

    Alton Lo
    Arista Networks
    Santa Clara, CA
    USA

    Email: altonlo@arista.com

    Sami Boutros
    VMware

    Email: sboutros@vmware.com

    Ankur Dubey
    VMware

    Email: adubey@vmware.com