          Geneve: Generic Network Virtualization Encapsulation
                      draft-ietf-nvo3-geneve-01

Abstract

   Network virtualization involves the cooperation of devices with a
   wide variety of capabilities such as software and hardware tunnel
   endpoints, transit fabrics, and centralized control clusters. As a
   result of their role in tying together different elements in the
   system, the requirements on tunnels are influenced by all of these
   components. Flexibility is therefore the most important aspect of a
   tunnel protocol if it is to keep pace with the evolution of the
   system. This draft describes Geneve, a protocol designed to
   recognize and accommodate these changing capabilities and needs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Networking has long featured a variety of tunneling, tagging, and
other encapsulation mechanisms. However, the advent of network
virtualization has caused a surge of renewed interest and a
corresponding increase in the introduction of new protocols. The

large number of protocols in this space, ranging all the way from
VLANs [IEEE.802.1Q-2014] and MPLS [RFC3031] through the more recent
VXLAN [RFC7348], NVGRE [RFC7637], and STT [I-D.davie-stt], often
leads to questions about the need for new encapsulation formats and
what it is about network virtualization in particular that leads to
their proliferation.

While many encapsulation protocols seek to simply partition the
underlay network or bridge between two domains, network
virtualization views the transit network as providing connectivity
between multiple components of a distributed system. In many ways
this system is similar to a chassis switch with the IP underlay
network playing the role of the backplane and tunnel endpoints on the
edge as line cards. When viewed in this light, the requirements
placed on the tunnel protocol are significantly different in terms of
the quantity of metadata necessary and the role of transit nodes.

Current work such as [VL2] and the NVO3 working group
[I-D.ietf-nvo3-dataplane-requirements] have described some of the
properties that the data plane must have to support network
virtualization. However, one additional defining requirement is the
need to carry system state along with the packet data. The use of
some metadata is certainly not a foreign concept - nearly all
protocols used for virtualization have at least 24 bits of identifier
space as a way to partition between tenants. This is often described
as overcoming the limits of 12-bit VLANs, and when seen in that
context, or any context where it is a true tenant identifier, 16
million possible entries is a large number. However, the reality is
that the metadata is not exclusively used to identify tenants and
encoding other information quickly starts to crowd the space. In
fact, when compared to the tags used to exchange metadata between
line cards on a chassis switch, 24-bit identifiers start to look
quite small. There are nearly endless uses for this metadata,
ranging from storing input ports for simple security policies to
service based context for interposing advanced middleboxes.

Existing tunnel protocols have each attempted to solve different
aspects of these new requirements, only to be quickly rendered out of
date by changing control plane implementations and advancements.
Furthermore, software and hardware components and controllers all
have different advantages and rates of evolution - a fact that should
be viewed as a benefit, not a liability or limitation. This draft
describes Geneve, a protocol which seeks to avoid these problems by
providing a framework for tunneling for network virtualization rather
than being prescriptive about the entire system.

1.1. 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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

1.2. Terminology

The NVO3 framework [RFC7365] defines many of the concepts commonly used in network virtualization. In addition, the following terms are specifically meaningful in this document:

Checksum offload. An optimization implemented by many NICs which enables computation and verification of upper layer protocol checksums in hardware on transmit and receive, respectively. This typically includes IP and TCP/UDP checksums which would otherwise be computed by the protocol stack in software.

Clos network. A technique for composing network fabrics larger than a single switch while maintaining non-blocking bandwidth across connection points. ECMP is used to divide traffic across the multiple links and switches that constitute the fabric. Sometimes termed "leaf and spine" or "fat tree" topologies.

ECMP. Equal Cost Multipath. A routing mechanism for selecting from among multiple best next hop paths by hashing packet headers in order to better utilize network bandwidth while avoiding reordering a single stream.

Geneve. Generic Network Virtualization Encapsulation. The tunnel protocol described in this draft.

LRO. Large Receive Offload. The receive-side equivalent function of LSO, in which multiple protocol segments (primarily TCP) are coalesced into larger data units.

NIC. Network Interface Card. A NIC could be part of a tunnel endpoint or transit device and can either process Geneve packets or aid in the processing of Geneve packets.

OAM. Operations, Administration, and Management. A suite of tools used to monitor and troubleshoot network problems.

Transit device. A forwarding element along the path of the tunnel
making up part of the Underlay Network. A transit device MAY be
capable of understanding the Geneve frame format but does not
originate or terminate Geneve packets.

LSO. Large Segmentation Offload. A function provided by many
commercial NICs that allows data units larger than the MTU to be
passed to the NIC to improve performance, the NIC being responsible
for creating smaller segments of size less than or equal to the MTU
with correct protocol headers. When referring specifically to TCP/
IP, this feature is often known as TSO (TCP Segmentation Offload).

Tunnel endpoint. A component performing encapsulation and
decapsulation of packets, such as Ethernet frames or IP datagrams, in
Geneve headers. As the ultimate consumer of any tunnel metadata,
endpoints have the highest level of requirements for parsing and
interpreting tunnel headers. Tunnel endpoints may consist of either
software or hardware implementations or a combination of the two.
Endpoints are frequently a component of an NVE but may also be found
in middleboxes or other elements making up an NVO3 Network.

VM. Virtual Machine.

2. Design Requirements

Geneve is designed to support network virtualization use cases, where
tunnels are typically established to act as a backplane between the
virtual switches residing in hypervisors, physical switches, or
middleboxes or other appliances. An arbitrary IP network can be used
as an underlay although Clos networks composed using ECMP links are a
common choice to provide consistent bisectional bandwidth across all
connection points. Figure 1 shows an example of a hypervisor, top of
rack switch for connectivity to physical servers, and a WAN uplink
connected using Geneve tunnels over a simplified Clos network. These
tunnels are used to encapsulate and forward frames from the attached
components such as VMs or physical links.

```
   +--------------------+            +-------+  +------+
   | +--+  +-------+---+ |            |Transit|--|Top of|==Physical
   | |VM|--|       |   | |  | | +------+ /|Router |  | Rack |==Servers
   | +--+  |Virtual|NIC|---|Top of|/ +-------+\/+------+
   | +--+  |Switch |   | | | Rack |\ +-------+/\+------+
   | |VM|--|       |   | |  | | +------+ \|Transit|  |Uplink|   WAN
   | +--+  +-------+---+ |            |Router |--|      |========>
   +--------------------+            +-------+  +------+
          Hypervisor

              ()================================()
                   Switch-Switch Geneve Tunnels
```
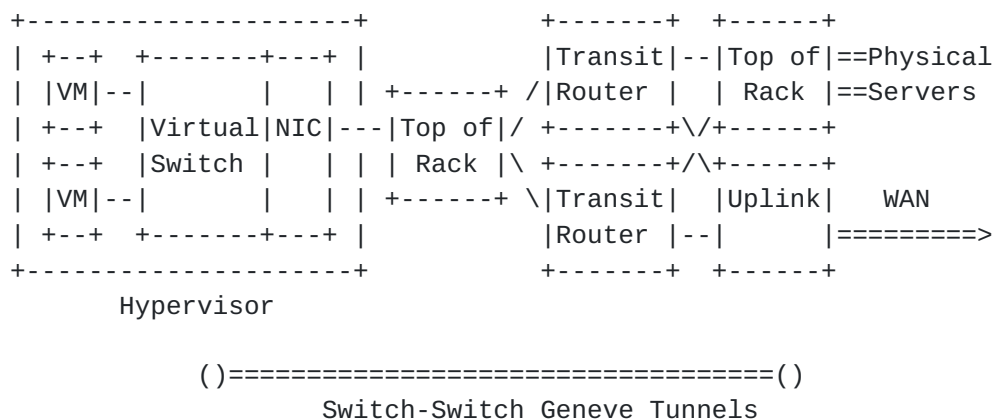
                 Figure 1: Sample Geneve Deployment

   To support the needs of network virtualization, the tunnel protocol
   should be able to take advantage of the differing (and evolving)
   capabilities of each type of device in both the underlay and overlay
   networks. This results in the following requirements being placed on
   the data plane tunneling protocol:

   o The data plane is generic and extensible enough to support current
      and future control planes.

   o Tunnel components are efficiently implementable in both hardware
      and software without restricting capabilities to the lowest common
      denominator.

   o High performance over existing IP fabrics.

   These requirements are described further in the following
   subsections.

2.1. Control Plane Independence

   Although some protocols for network virtualization have included a
   control plane as part of the tunnel format specification (most
   notably, the original VXLAN spec prescribed a multicast learning-
   based control plane), these specifications have largely been treated
   as describing only the data format. The VXLAN frame format has
   actually seen a wide variety of control planes built on top of it.

   There is a clear advantage in settling on a data format: most of the
   protocols are only superficially different and there is little
   advantage in duplicating effort. However, the same cannot be said of
   control planes, which are diverse in very fundamental ways. The case
   for standardization is also less clear given the wide variety in
   requirements, goals, and deployment scenarios.

As a result of this reality, Geneve aims to be a pure tunnel format
specification that is capable of fulfilling the needs of many control
planes by explicitly not selecting any one of them. This
simultaneously promotes a shared data format and increases the
chances that it will not be obsoleted by future control plane
enhancements.

2.2. Data Plane Extensibility

Achieving the level of flexibility needed to support current and
future control planes effectively requires an options infrastructure
to allow new metadata types to be defined, deployed, and either
finalized or retired. Options also allow for differentiation of
products by encouraging independent development in each vendor's core
specialty, leading to an overall faster pace of advancement. By far
the most common mechanism for implementing options is Type-Length-
Value (TLV) format.

It should be noted that while options can be used to support non-
wirespeed control frames, they are equally important on data frames
as well to segregate and direct forwarding (for instance, the
examples given before of input port based security policies and
service interposition both require tags to be placed on data
packets). Therefore, while it would be desirable to limit the
extensibility to only control frames for the purposes of simplifying
the datapath, that would not satisfy the design requirements.

2.2.1. Efficient Implementation

There is often a conflict between software flexibility and hardware
performance that is difficult to resolve. For a given set of
functionality, it is obviously desirable to maximize performance.
However, that does not mean new features that cannot be run at that
speed today should be disallowed. Therefore, for a protocol to be
efficiently implementable means that a set of common capabilities can
be reasonably handled across platforms along with a graceful
mechanism to handle more advanced features in the appropriate
situations.

The use of a variable length header and options in a protocol often
raises questions about whether it is truly efficiently implementable
in hardware. To answer this question in the context of Geneve, it is
important to first divide "hardware" into two categories: tunnel
endpoints and transit devices.

Endpoints must be able to parse the variable header, including any
options, and take action. Since these devices are actively
participating in the protocol, they are the most affected by Geneve.

However, as endpoints are the ultimate consumers of the data,
transmitters can tailor their output to the capabilities of the
recipient. As new functionality becomes sufficiently well defined to
add to endpoints, supporting options can be designed using ordering
restrictions and other techniques to ease parsing.

Transit devices MAY be able to interpret the options and participate
in Geneve packet processing. However, as non-terminating devices,
they do not originate or terminate the Geneve packet. The
participation of transit devices in Geneve packet processing is
OPTIONAL.

Further, either tunnel endpoints or transit devices MAY use offload
capabilities of NICs such as checksum offload to improve the
performance of Geneve packet processing. The presence of a Geneve
variable length header SHOULD NOT prevent the tunnel endpoints and
transit devices from using such offload capabilities.

2.3. Use of Standard IP Fabrics

IP has clearly cemented its place as the dominant transport mechanism
and many techniques have evolved over time to make it robust,
efficient, and inexpensive. As a result, it is natural to use IP
fabrics as a transit network for Geneve. Fortunately, the use of IP
encapsulation and addressing is enough to achieve the primary goal of
delivering packets to the correct point in the network through
standard switching and routing.

In addition, nearly all underlay fabrics are designed to exploit
parallelism in traffic to spread load across multiple links without
introducing reordering in individual flows. These equal cost
multipathing (ECMP) techniques typically involve parsing and hashing
the addresses and port numbers from the packet to select an outgoing
link. However, the use of tunnels often results in poor ECMP
performance without additional knowledge of the protocol as the
encapsulated traffic is hidden from the fabric by design and only
endpoint addresses are available for hashing.

Since it is desirable for Geneve to perform well on these existing
fabrics, it is necessary for entropy from encapsulated packets to be
exposed in the tunnel header. The most common technique for this is
to use the UDP source port, which is discussed further in
Section 3.3.

3. Geneve Encapsulation Details

The Geneve frame format consists of a compact tunnel header
encapsulated in UDP over either IPv4 or IPv6. A small fixed tunnel
header provides control information plus a base level of
functionality and interoperability with a focus on simplicity. This
header is then followed by a set of variable options to allow for
future innovation. Finally, the payload consists of a protocol data
unit of the indicated type, such as an Ethernet frame. Section 3.1
and Section 3.2 illustrate the Geneve frame format transported (for
example) over Ethernet along with an Ethernet payload.

3.1. Geneve Frame Format Over IPv4

```
     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

  Outer Ethernet Header:
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                 Outer Destination MAC Address                 |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Outer Destination MAC Address |   Outer Source MAC Address    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                   Outer Source MAC Address                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Optional Ethertype=C-Tag 802.1Q|  Outer VLAN Tag Information    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |      Ethertype=0x0800         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

  Outer IPv4 Header:
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Version|  IHL  |Type of Service|          Total Length         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |         Identification        |Flags|      Fragment Offset    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Time to Live |Protocol=17 UDP|         Header Checksum        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                   Outer Source IPv4 Address                   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                Outer Destination IPv4 Address                 |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Outer UDP Header:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Source Port = xxxx      |        Dest Port = 6081      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           UDP Length            |         UDP Checksum         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Geneve Header:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Ver|  Opt Len  |O|C|   Rsvd.  |              Protocol Type       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Virtual Network Identifier (VNI)     |   Reserved    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Variable Length Options                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Inner Ethernet Header (example payload):

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   Inner Destination MAC Address              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Inner Destination MAC Address |   Inner Source MAC Address    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   Inner Source MAC Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Optional Ethertype=C-Tag 802.1Q|  Inner VLAN Tag Information   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Payload:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Ethertype of Original Payload |                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                              |
 |                                   Original Ethernet Payload   |
 |                                                              |
 | (Note that the original Ethernet Frame's FCS is not included) |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Frame Check Sequence:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   New FCS (Frame Check Sequence) for Outer Ethernet Frame    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2. Geneve Frame Format Over IPv6

```
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
```

Outer Ethernet Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outer Destination MAC Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Outer Destination MAC Address |   Outer Source MAC Address    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outer Source MAC Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Optional Ethertype=C-Tag 802.1Q|  Outer VLAN Tag Information    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Ethertype=0x86DD        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Outer IPv6 Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |            Flow Label                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        | NxtHdr=17 UDP |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                    Outer Source IPv6 Address                  +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                  Outer Destination IPv6 Address               +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Outer UDP Header:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source Port = xxxx     |       Dest Port = 6081        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           UDP Length          |        UDP Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Geneve Header:
       +-+
       |Ver| Opt Len |O|C| Rsvd. | Protocol Type |
       +-+
       | Virtual Network Identifier (VNI) | Reserved |
       +-+
       | Variable Length Options |
       +-+

   Inner Ethernet Header (example payload):
       +-+
       | Inner Destination MAC Address |
       +-+
       | Inner Destination MAC Address | Inner Source MAC Address |
       +-+
       | Inner Source MAC Address |
       +-+
       |Optional Ethertype=C-Tag 802.1Q| Inner VLAN Tag Information |
       +-+

   Payload:
       +-+
       | Ethertype of Original Payload | |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
       | Original Ethernet Payload |
       | |
       | (Note that the original Ethernet Frame's FCS is not included) |
       +-+

   Frame Check Sequence:
       +-+
       | New FCS (Frame Check Sequence) for Outer Ethernet Frame |
       +-+

3.3. UDP Header

   The use of an encapsulating UDP [RFC0768] header follows the
   connectionless semantics of Ethernet and IP in addition to providing
   entropy to routers performing ECMP. The header fields are therefore
   interpreted as follows:

   Source port: A source port selected by the originating tunnel
      endpoint. This source port SHOULD be the same for all packets
      belonging to a single encapsulated flow to prevent reordering due
      to the use of different paths. To encourage an even distribution
      of flows across multiple links, the source port SHOULD be
      calculated using a hash of the encapsulated packet headers using,
      for example, a traditional 5-tuple. Since the port represents a

      flow identifier rather than a true UDP connection, the entire
      16-bit range MAY be used to maximize entropy.

   Dest port: IANA has assigned port 6081 as the fixed well-known
      destination port for Geneve. Although the well-known value should
      be used by default, it is RECOMMENDED that implementations make
      this configurable. The chosen port is used for identification of
      Geneve packets and MUST NOT be reversed for different ends of a
      connection as is done with TCP.

   UDP length: The length of the UDP packet including the UDP header.

   UDP checksum: The checksum MAY be set to zero on transmit for
      packets encapsulated in both IPv4 and IPv6 [RFC6935]. When a
      packet is received with a UDP checksum of zero it MUST be accepted
      and decapsulated. If the originating tunnel endpoint optionally
      encapsulates a packet with a non-zero checksum, it MUST be a
      correctly computed UDP checksum. Upon receiving such a packet,
      the egress endpoint MUST validate the checksum. If the checksum
      is not correct, the packet MUST be dropped, otherwise the packet
      MUST be accepted for decapsulation. It is RECOMMENDED that the
      UDP checksum be computed to protect the Geneve header and options
      in situations where the network reliability is not high and the
      packet is not protected by another checksum or CRC.

3.4. Tunnel Header Fields

   Ver (2 bits): The current version number is 0. Packets received by
      an endpoint with an unknown version MUST be dropped. Non-
      terminating devices processing Geneve packets with an unknown
      version number MUST treat them as UDP packets with an unknown
      payload.

   Opt Len (6 bits): The length of the options fields, expressed in
      four byte multiples, not including the eight byte fixed tunnel
      header. This results in a minimum total Geneve header size of 8
      bytes and a maximum of 260 bytes. The start of the payload
      headers can be found using this offset from the end of the base
      Geneve header.

   O (1 bit): OAM frame. This packet contains a control message
      instead of a data payload. Endpoints MUST NOT forward the payload
      and transit devices MUST NOT attempt to interpret or process it.
      Since these are infrequent control messages, it is RECOMMENDED
      that endpoints direct these packets to a high priority control
      queue (for example, to direct the packet to a general purpose CPU
      from a forwarding ASIC or to separate out control traffic on a

   NIC). Transit devices MUST NOT alter forwarding behavior on the
   basis of this bit, such as ECMP link selection.

   C (1 bit): Critical options present. One or more options has the
      critical bit set (see Section 3.5). If this bit is set then
      tunnel endpoints MUST parse the options list to interpret any
      critical options. On endpoints where option parsing is not
      supported the frame MUST be dropped on the basis of the 'C' bit in
      the base header. If the bit is not set tunnel endpoints MAY strip
      all options using 'Opt Len' and forward the decapsulated frame.
      Transit devices MUST NOT drop or modify packets on the basis of
      this bit.

   Rsvd. (6 bits): Reserved field which MUST be zero on transmission
      and ignored on receipt.

   Protocol Type (16 bits): The type of the protocol data unit
      appearing after the Geneve header. This follows the EtherType
      [ETYPES] convention with Ethernet itself being represented by the
      value 0x6558.

   Virtual Network Identifier (VNI) (24 bits): An identifier for a
      unique element of a virtual network. In many situations this may
      represent an L2 segment, however, the control plane defines the
      forwarding semantics of decapsulated packets. The VNI MAY be used
      as part of ECMP forwarding decisions or MAY be used as a mechanism
      to distinguish between overlapping address spaces contained in the
      encapsulated packet when load balancing across CPUs.

   Reserved (8 bits): Reserved field which MUST be zero on transmission
      and ignored on receipt.

   Transit devices MUST maintain consistent forwarding behavior
   irrespective of the value of 'Opt Len', including ECMP link
   selection. These devices SHOULD be able to forward packets
   containing options without resorting to a slow path.

3.5. Tunnel Options

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Class         |      Type     |R|R|R| Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Variable Option Data                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                         Geneve Option

The base Geneve header is followed by zero or more options in Type-
Length-Value format. Each option consists of a four byte option
header and a variable amount of option data interpreted according to
the type.

Option Class (16 bits): Namespace for the 'Type' field. IANA will
   be requested to create a "Geneve Option Class" registry to
   allocate identifiers for organizations, technologies, and vendors
   that have an interest in creating types for options. Each
   organization may allocate types independently to allow
   experimentation and rapid innovation. It is expected that over
   time certain options will become well known and a given
   implementation may use option types from a variety of sources. In
   addition, IANA will be requested to reserve specific ranges for
   standardized and experimental options.

Type (8 bits): Type indicating the format of the data contained in
   this option. Options are primarily designed to encourage future
   extensibility and innovation and so standardized forms of these
   options will be defined in a separate document.

   The high order bit of the option type indicates that this is a
   critical option. If the receiving endpoint does not recognize
   this option and this bit is set then the frame MUST be dropped.
   If the critical bit is set in any option then the 'C' bit in the
   Geneve base header MUST also be set. Transit devices MUST NOT
   drop packets on the basis of this bit. The following figure shows
   the location of the 'C' bit in the 'Type' field:

   0 1 2 3 4 5 6 7 8
   +-+-+-+-+-+-+-+-+
   |C| Type |
   +-+-+-+-+-+-+-+-+

   The requirement to drop a packet with an unknown critical option
   applies to the entire tunnel endpoint system and not a particular
   component of the implementation. For example, in a system
   comprised of a forwarding ASIC and a general purpose CPU, this
   does not mean that the packet must be dropped in the ASIC. An
   implementation may send the packet to the CPU using a rate-limited
   control channel for slow-path exception handling.

R (3 bits): Option control flags reserved for future use. MUST be
   zero on transmission and ignored on receipt.

Length (5 bits): Length of the option, expressed in four byte
   multiples excluding the option header. The total length of each
   option may be between 4 and 128 bytes. Packets in which the total

length of all options is not equal to the 'Opt Len' in the base
header are invalid and MUST be silently dropped if received by an
endpoint.

Variable Option Data: Option data interpreted according to 'Type'.

3.5.1. Options Processing

Geneve options are primarily intended to be originated and processed
by tunnel endpoints. However, options MAY be processed by transit
devices along the tunnel path as well. Transit devices not
processing Geneve options SHOULD process Geneve frame as any other
UDP frame and maintain consistent forwarding behavior.

In tunnel endpoints, the generation and interpretation of options is
determined by the control plane, which is out of the scope of this
document. However, to ensure interoperability between heterogeneous
devices two requirements are imposed on endpoint devices:

o Receiving endpoints MUST drop packets containing unknown options
   with the 'C' bit set in the option type.

o Sending endpoints MUST NOT assume that options will be processed
   sequentially by the receiver in the order they were transmitted.

When designing a Geneve option, it is important to consider how the
option will evolve in the future. Once an option is defined it is
reasonable to expect that implementations may come to depend on a
specific behavior. As a result, the scope of any future changes must
be carefully described upfront.

Unexpectedly significant interoperability issues may result from
changing the length of an option that was defined to be a certain
size. A particular option is specified to have either a fixed
length, which is constant, or a variable length, which may change
over time or for different use cases. This property is part of the
definition of the option and conveyed by the 'Type'. For fixed
length options, some implementations may choose to ignore the length
field in the option header and instead parse based on the well known
length associated with the type. In this case, redefining the length
will impact not only parsing of the option in question but also any
options that follow. Therefore, options that are defined to be fixed
length in size MUST NOT be redefined to a different length. Instead,
a new 'Type' should be allocated.

4. **Implementation and Deployment Considerations**

4.1. **Encapsulation of Geneve in IP**

   As an IP-based tunnel protocol, Geneve shares many properties and
   techniques with existing protocols. The application of some of these
   are described in further detail, although in general most concepts
   applicable to the IP layer or to IP tunnels generally also function
   in the context of Geneve.

4.1.1. **IP Fragmentation**

   To prevent fragmentation and maximize performance, the best practice
   when using Geneve is to ensure that the MTU of the physical network
   is greater than or equal to the MTU of the encapsulated network plus
   tunnel headers. Manual or upper layer (such as TCP MSS clamping)
   configuration can be used to ensure that fragmentation never takes
   place, however, in some situations this may not be feasible.

   It is strongly RECOMMENDED that Path MTU Discovery ([RFC1191],
   [RFC1981]) be used by setting the DF bit in the IP header when Geneve
   packets are transmitted over IPv4 (this is the default with IPv6).
   The use of Path MTU Discovery on the transit network provides the
   encapsulating endpoint with soft-state about the link that it may use
   to prevent or minimize fragmentation depending on its role in the
   virtualized network.

   Note that some implementations may not be capable of supporting
   fragmentation or other less common features of the IP header, such as
   options and extension headers.

4.1.2. **DSCP and ECN**

   When encapsulating IP (including over Ethernet) frames in Geneve,
   there are several options for propagating DSCP and ECN bits from the
   inner header to the tunnel on transmission and the reverse on
   reception.

   [RFC2983] lists considerations for mapping DSCP between inner and
   outer IP headers. Network virtualization is typically more closely
   aligned with the Pipe model described, where the DSCP value on the
   tunnel header is set based on a policy (which may be a fixed value,
   one based on the inner traffic class, or some other mechanism for
   grouping traffic). Aspects of the Uniform model (which treats the
   inner and outer DSCP value as a single field by copying on ingress
   and egress) may also apply, such as the ability to remark the inner
   header on tunnel egress based on transit marking. However, the
   Uniform model is not conceptually consistent with network

virtualization, which seeks to provide strong isolation between
encapsulated traffic and the physical network.

[RFC6040] describes the mechanism for exposing ECN capabilities on IP
tunnels and propagating congestion markers to the inner packets.
This behavior SHOULD be followed for IP packets encapsulated in
Geneve.

4.1.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two
endpoints or may utilize broadcast or multicast addressing. It is
not required that inner and outer addressing match in this respect.
For example, in physical networks that do not support multicast,
encapsulated multicast traffic may be replicated into multiple
unicast tunnels or forwarded by policy to a unicast location
(possibly to be replicated there).

With physical networks that do support multicast it may be desirable
to use this capability to take advantage of hardware replication for
encapsulated packets. In this case, multicast addresses may be
allocated in the physical network corresponding to tenants,
encapsulated multicast groups, or some other factor. The allocation
of these groups is a component of the control plane and therefore
outside of the scope of this document. When physical multicast is in
use, the 'C' bit in the Geneve header may be used with groups of
devices with heterogeneous capabilities as each device can interpret
only the options that are significant to it if they are not critical.

4.1.4. Unidirectional Tunnels

Generally speaking, a Geneve tunnel is a unidirectional concept. IP
is not a connection oriented protocol and it is possible for two
endpoints to communicate with each other using different paths or to
have one side not transmit anything at all. As Geneve is an IP-based
protocol, the tunnel layer inherits these same characteristics.

It is possible for a tunnel to encapsulate a protocol, such as TCP,
which is connection oriented and maintains session state at that
layer. In addition, implementations MAY model Geneve tunnels as
connected, bidirectional links, such as to provide the abstraction of
a virtual port. In both of these cases, bidirectionality of the
tunnel is handled at a higher layer and does not affect the operation
of Geneve itself.

4.2. Constraints on Protocol Features

Geneve is intended to be flexible to a wide range of current and
future applications. As a result, certain constraints may be placed
on the use of metadata or other aspects of the protocol in order to
optimize for a particular use case. For example, some applications
may limit the types of options which are supported or enforce a
maximum number or length of options. Other applications may only
handle certain encapsulated payload types, such as Ethernet or IP.
This could be either globally throughout the system or, for example,
restricted to certain classes of devices or network paths.

These constraints may be communicated to tunnel endpoints either
explicitly through a control plane or implicitly by the nature of the
application. As Geneve is defined as a data plane protocol that is
control plane agnostic, the exact mechanism is not defined in this
document.

4.3. NIC Offloads

Modern NICs currently provide a variety of offloads to enable the
efficient processing of packets. The implementation of many of these
offloads requires only that the encapsulated packet be easily parsed
(for example, checksum offload). However, optimizations such as LSO
and LRO involve some processing of the options themselves since they
must be replicated/merged across multiple packets. In these
situations, it is desirable to not require changes to the offload
logic to handle the introduction of new options. To enable this,
some constraints are placed on the definitions of options to allow
for simple processing rules:

o When performing LSO, a NIC MUST replicate the entire Geneve header
   and all options, including those unknown to the device, onto each
   resulting segment. However, a given option definition may
   override this rule and specify different behavior in supporting
   devices. Conversely, when performing LRO, a NIC MAY assume that a
   binary comparison of the options (including unknown options) is
   sufficient to ensure equality and MAY merge packets with equal
   Geneve headers.

o Option ordering is not significant and packets with the same
   options in a different order MAY be processed alike.

o NICs performing offloads MUST NOT drop packets with unknown
   options, including those marked as critical.

There is no requirement that a given implementation of Geneve employ
the offloads listed as examples above. However, as these offloads

are currently widely deployed in commercially available NICs, the
rules described here are intended to enable efficient handling of
current and future options across a variety of devices.

4.4. Inner VLAN Handling

Geneve is capable of encapsulating a wide range of protocols and
therefore a given implementation is likely to support only a small
subset of the possibilities. However, as Ethernet is expected to be
widely deployed, it is useful to describe the behavior of VLANs
inside encapsulated Ethernet frames.

As with any protocol, support for inner VLAN headers is OPTIONAL. In
many cases, the use of encapsulated VLANs may be disallowed due to
security or implementation considerations. However, in other cases
trunking of VLAN frames across a Geneve tunnel can prove useful. As
a result, the processing of inner VLAN tags upon ingress or egress
from a tunnel endpoint is based upon the configuration of the
endpoint and/or control plane and not explicitly defined as part of
the data format.

5. Interoperability Issues

Viewed exclusively from the data plane, Geneve does not introduce any
interoperability issues as it appears to most devices as UDP frames.
However, as there are already a number of tunnel protocols deployed
in network virtualization environments, there is a practical question
of transition and coexistence.

Since Geneve is a superset of the functionality of the three most
common protocols used for network virtualization (VXLAN, NVGRE, and
STT) it should be straightforward to port an existing control plane
to run on top of it with minimal effort. With both the old and new
frame formats supporting the same set of capabilities, there is no
need for a hard transition - endpoints directly communicating with
each other use any common protocol, which may be different even
within a single overall system. As transit devices are primarily
forwarding frames on the basis of the IP header, all protocols appear
similar and these devices do not introduce additional
interoperability concerns.

To assist with this transition, it is strongly suggested that
implementations support simultaneous operation of both Geneve and
existing tunnel protocols as it is expected to be common for a single
node to communicate with a mixture of other nodes. Eventually, older
protocols may be phased out as they are no longer in use.

6. Security Considerations

   As UDP/IP packets, Geneve does not have any inherent security
   mechanisms. As a result, an attacker with access to the underlay
   network transporting the IP frames has the ability to snoop or inject
   packets. Legitimate but malicious tunnel endpoints may also spoof
   identifiers in the tunnel header to gain access to networks owned by
   other tenants.

   Within a particular security domain, such as a data center operated
   by a single provider, the most common and highest performing security
   mechanism is isolation of trusted components. Tunnel traffic can be
   carried over a separate VLAN and filtered at any untrusted
   boundaries. In addition, tunnel endpoints should only be operated in
   environments controlled by the service provider, such as the
   hypervisor itself rather than within a customer VM.

   When crossing an untrusted link, such as the public Internet, IPsec
   [RFC4301] may be used to provide authentication and/or encryption of
   the IP packets formed as part of Geneve encapsulation. If the remote
   tunnel endpoint is not completely trusted, for example it resides on
   a customer premises, then it may also be necessary to sanitize any
   tunnel metadata to prevent tenant-hopping attacks.

   Geneve does not otherwise affect the security of the encapsulated
   packets.

7. IANA Considerations

   IANA has allocated UDP port 6081 as the well-known destination port
   for Geneve. Upon publication, the registry should be updated to cite
   this document. The original request was:

   Service Name: geneve
   Transport Protocol(s): UDP
   Assignee: Jesse Gross <jgross@vmware.com>
   Contact: Jesse Gross <jgross@vmware.com>
   Description: Generic Network Virtualization Encapsulation (Geneve)
   Reference: This document
   Port Number: 6081

   In addition, IANA is requested to create a "Geneve Option Class"
   registry to allocate Option Classes. This shall be a registry of
   16-bit hexadecimal values along with descriptive strings. The
   identifiers 0x0-0xFF are to be reserved for standardized options for
   allocation by IETF Review [RFC5226] and 0xFFFF for Experimental Use.
   Otherwise, identifiers are to be assigned to any organization with an
   interest in creating Geneve options on a First Come First Served

basis. The registry is to be populated with the following initial
values:

```
+----------------+------------------------------------+
| Option Class   | Description                        |
+----------------+------------------------------------+
| 0x0000..0x00FF | Unassigned - IETF Review           |
| 0x0100         | Linux                              |
| 0x0101         | Open vSwitch                       |
| 0x0102         | Open Virtual Networking (OVN)      |
| 0x0103..0xFFFE | Unassigned - First Come First Served |
| 0xFFFF         | Experimental                       |
+----------------+------------------------------------+
```

8. Contributors

The following individuals were authors of an earlier version of this
document and made significant contributions:

T. Sridhar
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
USA

Email: tsridhar@vmware.com

Pankaj Garg
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
USA

Email: pankajg@microsoft.com

Chris Wright
Red Hat Inc.
1801 Varsity Drive
Raleigh, NC 27606
USA

Email: chrisw@redhat.com

Puneet Agarwal
Innovium, Inc.

      6001 America Center Drive
      San Jose, CA 95002
      USA

      Email: puneet@innovium.com

      Kenneth Duda
      Arista Networks
      5453 Great America Parkway
      Santa Clara, CA 95054
      USA

      Email: kduda@arista.com

      Dinesh G. Dutt
      Cumulus Networks
      140C S. Whisman Road
      Mountain View, CA 94041
      USA

      Email: ddutt@cumulusnetworks.com

      Jon Hudson
      Brocade Communications Systems, Inc.
      130 Holger Way
      San Jose, CA 95134
      USA

      Email: jon.hudson@gmail.com

      Ariel Hendel
      Broadcom Corporation
      3151 Zanker Road
      San Jose, CA 95134
      USA

      Email: ahendel@broadcom.com

9. Acknowledgements

   The authors wish to thank Martin Casado, Bruce Davie and Dave Thaler
   for their input, feedback, and helpful suggestions.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
           DOI 10.17487/RFC0768, August 1980,
           <http://www.rfc-editor.org/info/rfc768>.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
           IANA Considerations Section in RFCs", BCP 26, RFC 5226,
           DOI 10.17487/RFC5226, May 2008,
           <http://www.rfc-editor.org/info/rfc5226>.

10.2. Informative References

[ETYPES]   The IEEE Registration Authority, "IEEE 802 Numbers", 2013,
           <http://www.iana.org/assignments/ieee-802-numbers/
           ieee-802-numbers.xml>.

[I-D.davie-stt]
           Davie, B. and J. Gross, "A Stateless Transport Tunneling
           Protocol for Network Virtualization (STT)", draft-davie-
           stt-06 (work in progress), April 2014.

[I-D.ietf-nvo3-dataplane-requirements]
           Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L.,
           and B. Khasnabish, "NVO3 Data Plane Requirements", draft-
           ietf-nvo3-dataplane-requirements-03 (work in progress),
           April 2014.

[IEEE.802.1Q-2014]
           IEEE, "IEEE Standard for Local and metropolitan area
           networks -- Bridges and Bridged Networks", IEEE
           Std 802.1Q, 2014.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
           DOI 10.17487/RFC1191, November 1990,
           <http://www.rfc-editor.org/info/rfc1191>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
           for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August
           1996, <http://www.rfc-editor.org/info/rfc1981>.

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

   [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol
              Label Switching Architecture", RFC 3031,
              DOI 10.17487/RFC3031, January 2001,
              <http://www.rfc-editor.org/info/rfc3031>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <http://www.rfc-editor.org/info/rfc4301>.

   [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion
              Notification", RFC 6040, DOI 10.17487/RFC6040, November
              2010, <http://www.rfc-editor.org/info/rfc6040>.

   [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and
              UDP Checksums for Tunneled Packets", RFC 6935,
              DOI 10.17487/RFC6935, April 2013,
              <http://www.rfc-editor.org/info/rfc6935>.

   [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>.

   [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network
              Virtualization Using Generic Routing Encapsulation",
              RFC 7637, DOI 10.17487/RFC7637, September 2015,
              <http://www.rfc-editor.org/info/rfc7637>.

   [VL2]   Greenberg et al, , "VL2: A Scalable and Flexible Data
              Center Network", 2009.

              Proc. ACM SIGCOMM 2009

Authors' Addresses

    Jesse Gross (editor)
    VMware, Inc.
    3401 Hillview Ave.
    Palo Alto, CA 94304
    USA

    Email: jgross@vmware.com

    Ilango Ganga (editor)
    Intel Corporation
    2200 Mission College Blvd.
    Santa Clara, CA 95054
    USA

    Email: ilango.s.ganga@intel.com