

NVO3 WG
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2019

Ran. Chen
Fangwei. Hu
Yubao. wang
ZTE Corporation
Yufeng. liu
Volta Networks
March 24, 2019

YANG Data Model for NVO3 Protocol
draft-chen-nvo3-yang-02.txt

Abstract

This document defines a YANG data model for NVO3 configuration and operation. This YANG model covers two types of encapsulations: Geneve, and VXLAN-GPE

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 25, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Design of the Data Model	2
3. Configuration	4
4. Control plane configuration	4
5. States	4
6. NVO3 YANG Data Model	4
7. Security Considerations	13
8. IANA Considerations	13
9. Normative references	13
Authors' Addresses	14

1. Introduction

This document defines a YANG data model for NVO3 configuration and operation. This YANG model covers two types of encapsulations: Geneve, and VXLAN-GPE.

2. Design of the Data Model

```

module: ietf-nvo3
+--rw nvo3
|   +--rw vxlan-enable?          boolean
|   +--rw geneve-enable?        boolean
|   +--rw nvo3-instance* [vni]
|       +--rw vni                vni
|       +--rw protocol-type?    enumeration
|       +--rw vtep-ipv4?        inet:ipv4-address-no-zone
|       +--rw vtep-ipv6?        inet:ipv6-address-no-zone
|       +--rw bridge-interface? if:interface-ref
|       +--rw (control-plane)?
|           +--:(evpn)
|           |   +--rw evpn-instance?    evpn-instance-ref
|           +--:(static-config)
|           |   +--rw ac-name?          ac-name
|           |   +--rw l2interface-name? if:interface-ref
|       +--rw unicast-tunnel* [unicast-tunnel-name]
|           |   +--rw unicast-tunnel-name    unicast-tunnel-name
|       +--rw multicast-tunnel* [multicast-tunnel-name]
|           |   +--rw multicast-tunnel-name    multicast-tunnel-name
|       +--rw unicast-tunnel* [unicast-tunnel-name]
|           |   +--rw unicast-tunnel-name    unicast-tunnel-name
|           +--rw encaptype?                enumeration
|           +--rw tunnel-source-ipv4?      inet:ipv4-prefix

```

```

|   +---rw tunnel-source-ipv6?          inet:ipv6-prefix
|   +---rw tunnel-destination-ipv4?     inet:ipv4-prefix
|   +---rw tunnel-destination-ipv6?     inet:ipv6-prefix
+---rw multicast-tunnel* [multicast-tunnel-name]
|   +---rw multicast-tunnel-name        multicast-tunnel-name
|   +---rw encaps-type?                 enumeration
|   +---rw tunnel-source-ipv4?          inet:ipv4-prefix
|   +---rw tunnel-source-ipv6?          inet:ipv6-prefix
|   +---rw tunnel-destination-ipv4* [destination-ipv4]
|   |   +---rw destination-ipv4        inet:ipv4-prefix
+---rw tunnel-destination-ipv6* [destination-ipv6]
|   +---rw destination-ipv6            inet:ipv6-prefix
+---ro nvo3-state
+---ro nvo3-instance* [vni]
|   +---ro vni                          vni
|   +---ro protocol-type?               enumeration
|   +---ro vtep-ipv4?                   inet:ipv4-address-no-zone
|   +---ro vtep-ipv6?                   inet:ipv6-address-no-zone
|   +---ro bridge-interface?            if:interface-ref
|   +---ro (control-plane)?
|   |   +---:(evpn)
|   |   |   +---ro evpn-instance?      evpn-instance-ref
|   |   +---:(static-config)
|   |   |   +---ro ac-name?             ac-name
|   |   |   +---ro l2interface-name?    if:interface-ref
+---ro unicast-tunnel* [unicast-tunnel-name]
|   +---ro unicast-tunnel-name          unicast-tunnel-name
+---ro multicast-tunnel* [multicast-tunnel-name]
|   +---ro multicast-tunnel-name        multicast-tunnel-name
+---ro unicast-tunnel* [unicast-tunnel-name]
|   +---ro unicast-tunnel-name          unicast-tunnel-name
|   +---ro encaps-type?                 enumeration
|   +---ro tunnel-source-ipv4?          inet:ipv4-prefix
|   +---ro tunnel-source-ipv6?          inet:ipv6-prefix
|   +---ro tunnel-destination-ipv4?     inet:ipv4-prefix
|   +---ro tunnel-destination-ipv6?     inet:ipv6-prefix
+---ro multicast-tunnel* [multicast-tunnel-name]
|   +---ro multicast-tunnel-name        multicast-tunnel-name
|   +---ro encaps-type?                 enumeration
|   +---ro tunnel-source-ipv4?          inet:ipv4-prefix
|   +---ro tunnel-source-ipv6?          inet:ipv6-prefix
|   +---ro tunnel-destination-ipv4* [destination-ipv4]
|   |   +---ro destination-ipv4        inet:ipv4-prefix
+---ro tunnel-destination-ipv6* [destination-ipv6]
|   +---ro destination-ipv6            inet:ipv6-prefix

```

3. Configuration

This Container defines the configuration parameters related to NVO3.

The configuration includes vxlan enable, geneve enable, parameters associated with nvo3-instance, parameters associated with unicast-tunnel and multicast-tunnel.

In this document, we contains two types of encapsulation: GENEVE[I-D.ietf-nvo3-geneve] and VXLAN-GPE[I-D.ietf-nvo3-vxlan-gpe].

4. Control plane configuration

This Module will be defined in the next version.

5. States

The operational states contains basic parameters associated with nvo3, such as parameters associated with nvo3-instance, unicast-tunnel and multicast-tunnel.

6. NVO3 YANG Data Model

```
<CODE BEGINS> file "ietf-NVO3@2019-03-11.yang"
module ietf-nvo3{
    namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3";
    prefix "nvo3";

    import ietf-inet-types {
    prefix "inet";
    }

    import ietf-interfaces {
    prefix "if";
    }

    organization
        "IETF Nvo3( Network Virtualization Overlays) Working Group";

    contact
        "WG List: <mailto:nvo3@ietf.org>
        WG Chair: Matthew Bocci
                <mailto:matthew.bocci@nokia.com>
        WG Chair: Sam Aldrin
                <mailto:aldrin.ietf@gmail.com>

        Editor:   Ran Chen
```

```

                                <mailto:chen.ran@zte.com.cn>
Editor:   Fangwei Hu
          <mailto:hu.fangwei@zte.com.cn>
Editor:   yubao wang
          <mailto:wang.yubao@zte.com.cn>
Editor:   xufeng liu
          <mailto:xufeng.liu.ietf@gmail.com>
";

    description
    "The YANG module defines a generic configuration model for nvo3
yang module.";
    revision 2019-03-20{
        description
        "02 version";
        reference "draft-chen-nvo3-yang-02";
    }

    revision 2019-03-11{
        description
        "01 version";
        reference "draft-chen-nvo3-yang-01";
    }

    revision 2018-10-31{
        description
        "Initial version";
        reference "draft-chen-nvo3-yang-00";
    }

/*Typedefs*/

typedef vni{
    type uint32;
    description
    "Virtual Network Identifier";
}

typedef unicast-tunnel-name{
    type string;
    description
    "the name for unicast tunnel";
}
typedef multicast-tunnel-name{
    type string;
    description
    "the name for multicast tunnel";
}
```

```
typedef evpn-instance-ref {
type leafref {
  path "/evpn/evpn-instances/evpn-instance/name";
}
description "A leafref type to an EVPN instance";
}

typedef ac-name{
  type string;
  description
    "the name for ac";
}

typedef interface-name{
  type string;
  description
    "the name for interface";
}

grouping unicast-tunnel-cfg{
  leaf encaps-type{
    type enumeration{
      enum "vxlan"{
        description
          "vxlan type";
      }
      enum "geneve"{
        description
          "geneve type";
      }
    }
    description "the type for encapsulation.";
  }
  leaf tunnel-source-ipv4{
    type inet:ipv4-prefix;
    description
      "tunnel source ipv4 prefix.";
  }
  leaf tunnel-source-ipv6{
    type inet:ipv6-prefix;
    description
      "tunnel source ipv6 prefix.";
  }
  leaf tunnel-destination-ipv4{
    type inet:ipv4-prefix;
    description
      "tunnel destination ipv4 prefix.";
  }
}
```

```
    }
    leaf tunnel-destination-ipv6{
        type inet:ipv6-prefix;
        description
            "tunnel destination ipv6 prefix.";
    }
    description
        "defines the unicast tunnel configuration.";
    }
    grouping multicast-tunnel-cfg{
leaf encapsype{
    type enumeration{
        enum "vxlan"{
            description
                "vxlan type";
        }
        enum "geneve"{
            description
                "geneve type";
        }
    }
    description "the type for encapsulation.";
}
leaf tunnel-source-ipv4{
    type inet:ipv4-prefix;
    description
        "tunnel source ipv4 prefix.";
}
leaf tunnel-source-ipv6{
    type inet:ipv6-prefix;
    description
        "tunnel source ipv6 prefix.";
}
    list tunnel-destination-ipv4{
key "destination-ipv4";
    description
        "the list of destination ipv4 prefix.";
        leaf destination-ipv4{
            type inet:ipv4-prefix;
            description
                "tunnel destination ipv4 prefix.";
        }
    }
    list tunnel-destination-ipv6{
key "destination-ipv6";
    description
        "the list of destination ipv6 prefix.";
        leaf destination-ipv6{
```

```
        type inet:ipv6-prefix;
        description
            "tunnel destination ipv6 prefix.";
    }
    }
    description
        "defines the multicast tunnel configuration.";
}

container nvo3{
    leaf vxlan-enable{
        type boolean;
        default false;
        description
            "Enables vxlan protocol.";
    }
    leaf geneve-enable{
        type boolean;
        default false;
        description
            "Enables geneve protocol.";
    }
    list nvo3-instance {
        key "vni";
        leaf vni {
            type vni;
            description "Virtual Network Identifier.";
        }
    }
    leaf protocol-type{
        type enumeration{
            enum "ipv4"{
                description
                    "ipv4 protocol";
            }
            enum "ipv6"{
                description
                    "ipv6 protocol";
            }
            enum "ethernet"{
                description
                    "ethernet protocol";
            }
            enum "mpls"{
                description
                    "mpls protocol";
            }
            enum "GBP"{
                description
```



```
        "gbp";
    }
    enum "vBNG"{
        description
            "vbng";
    }
}
description "the next protocol type";
}
    leaf vtep-ipv4 {
        type inet:ipv4-address-no-zone;
        description
            "NVO3 tunnel source address";
    }
leaf vtep-ipv6 {
    type inet:ipv6-address-no-zone;
    description
        "ipv6 NVO3 tunnel source address";
}

leaf bridge-interface {
    type if:interface-ref;
    description "bridge interface.";
}

    choice control-plane {
        case evpn{
            leaf evpn-instance{
                type evpn-instance-ref;
                description "Reference to an EVPN instance";
            }
        }
        case static-config{
leaf ac-name {
    type ac-name;
    description "the name for ac.";
}

        leaf l2interface-name{
            type if:interface-ref;
            description "L2 interface.";
        }
description
                    "static-config.";
        }
description "the control-plane.";
    }
        list unicast-tunnel{
            key "unicast-tunnel-name";
```

```
        leaf unicast-tunnel-name {
            type unicast-tunnel-name;
            description "the name for unicast tunnel.";
        }
    description
        "the information for the unicast tunnel configur
ation.";
    }

    list multicast-tunnel{
        key "multicast-tunnel-name";
        leaf multicast-tunnel-name {
            type multicast-tunnel-name;
            description "the name for multicast tunnel.";
        }
    description
        "the information for the multicast tunnel.";
    }
    description
        "defines the nvo3 instance configuration.";
}

    list unicast-tunnel{
        key "unicast-tunnel-name";
        leaf unicast-tunnel-name {
            type unicast-tunnel-name;
            description "the name for unicast tunnel.";
        }
        uses nvo3:unicast-tunnel-cfg;
    description
        "defines the unicast tunnel configuration.";
    }

    list multicast-tunnel{
        key "multicast-tunnel-name";
        leaf multicast-tunnel-name {
            type multicast-tunnel-name;
            description "the name for multicast tunnel.";
        }
        uses nvo3:multicast-tunnel-cfg;
    description
        "defines the multicast tunnel configuration.";
    }
    description
        "defines the nvo3 configuration.";
    }

    container nvo3-state{
```

```
        config false;
        description
            "nvo3 operational state.";
        list nvo3-instance {
key "vni";
leaf vni {
    type vni;
    description "Virtual Network Identifier.";
}
leaf protocol-type{
    type enumeration{
        enum "ipv4"{
            description
                "ipv4 protocol";
        }
        enum "ipv6"{
            description
                "ipv6 protocol";
        }
        enum "ethernet"{
            description
                "ethernet protocol";
        }
        enum "mpls"{
            description
                "mpls protocol";
        }
        enum "GBP"{
            description
                "gbp";
        }
        enum "vBNG"{
            description
                "vbng";
        }
    }
    description "the next protocol type";
}
leaf vtep-ipv4 {
    type inet:ipv4-address-no-zone;
    description
        "NVO3 tunnel source address";
}
leaf vtep-ipv6 {
    type inet:ipv6-address-no-zone;
    description
        "ipv6 NVO3 tunnel source address";
}
```

```
leaf bridge-interface {
  type if:interface-ref;
  description "bridge interface.";
}
choice control-plane {
  case evpn{
    leaf evpn-instance{
      type evpn-instance-ref;
      description "Reference to an EVPN instance";
    }
  }
  case static-config{
    leaf ac-name {
      type ac-name;
      description "the name for ac.";
    }
    leaf l2interface-name{
      type if:interface-ref;
      description "L2 interface.";
    }
  }
  description
    "static-config.";
}
description "the control-plane.";
}
list unicast-tunnel{
  key "unicast-tunnel-name";
  leaf unicast-tunnel-name {
    type unicast-tunnel-name;
    description "the name for unicast tunnel.";
  }
  description
    "the information for the unicast tunnel.";
}

list multicast-tunnel{
  key "multicast-tunnel-name";
  leaf multicast-tunnel-name {
    type multicast-tunnel-name;
    description "the name for multicast tunnel.";
  }
  description
    "the state for multicast tunnel.";
}
description
  "the state for nvo3 instance.";
}
```

```
        list unicast-tunnel{
        key "unicast-tunnel-name";
        leaf unicast-tunnel-name {
            type unicast-tunnel-name;
            description "the name for unicast tunnel.";
        }
        uses nvo3:unicast-tunnel-cfg;
    description
        "the state for the unicast tunnel.";
    }

    list multicast-tunnel{
    key "multicast-tunnel-name";
    leaf multicast-tunnel-name {
        type multicast-tunnel-name;
        description "the name for multicast tunnel.";
    }
    uses nvo3:multicast-tunnel-cfg;
description
    "the state for the multicast tunnel.";
    }
    }
    }
```

<CODE ENDS>

7. Security Considerations

TBD.

8. IANA Considerations

This document requires no IANA Actions. Please remove this section before RFC publication.

9. Normative references

[I-D.ietf-nvo3-encap]

Boutros, S., "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-02 (work in progress), September 2018.

[I-D.ietf-nvo3-geneve]

Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-12 (work in progress), March 2019.

- [I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <<https://www.rfc-editor.org/info/rfc7223>>.
- [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, <<https://www.rfc-editor.org/info/rfc7348>>.

Authors' Addresses

Ran Chen
ZTE Corporation

Email: chen.ran@zte.com.cn

Fangwei Hu
ZTE Corporation

Email: hu.fangwei@zte.com.cn

Yubao wang
ZTE Corporation

Email: wang.yubao@zte.com.cn

Yufeng liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 28, 2019

X. de Foy
U. Olvera-Hernandez
InterDigital Communications
Jan 24, 2019

5G-Datacenter Interconnection Use Case
draft-defoy-nvo3-5g-datacenter-interconnection-00

Abstract

Interconnection between 5G networks and datacenter networks provide a new use case for NVO3 and for the 3rd Generation Partnership Project (3GPP) "5GLAN" feature. This document describes how layer-2 and layer-3 datacenter VPN technology can interoperate with anchor User Plane Functions (UPF) to interconnect 5G devices and datacenter servers over a virtual LAN.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. About the 5GLAN Feature	2
1.2. Interaction between 5GLAN and Virtual Networks	3
1.3. Goals of this Document	3
2. New Use Case for NVO3 and 5GLAN: 5G/Datacenter Interconnection	3
3. Architecture Overview	4
4. Major Features of a 5G/Datacenter Interconnection	6
5. IANA Considerations	9
6. Security Considerations	9
7. Next Steps	9
8. Informative References	9
Appendix A. 5GLAN Background Information	10
Authors' Addresses	11

1. Introduction

1.1. About the 5GLAN Feature

In an ongoing work, 3GPP is seeking to enable LAN-like virtual networking between groups of end devices. Appendix A provides references and additional details relative to the 5GLAN work.

5GLAN requirements, defined in [_3GPP.22.261] can be shortly summarized as:

A selected set of end devices can communicate with each other as if over a LAN, including over Ethernet or over IP, using a private addressing scheme.

Unicast and multicast/broadcast communication is enabled between end devices, in some cases with a required latency (e.g. 180ms), or more generally a given consistent QoE.

End device mobility is supported.

General concepts and use cases were described in [_3GPP.23.734].

In home and enterprise deployments, a goal of 5GLAN is to facilitate interworking with, and sometimes replacing, existing infrastructure, composed of fixed and wireless LANs.

In enterprise deployments, 5GLAN will also provide a VPN-like service integrating mobile devices with enterprise networks.

In industrial automation deployments, 5GLAN can enable low-latency, reliable and deterministic LAN traffic in manufacturing, machine control, packaging and printing use cases.

1.2. Interaction between 5GLAN and Virtual Networks

5GLAN connectivity does not have to be confined entirely within a 5G network domain. The conclusion of the 5GLAN study [3GPP.23.734] states that the standardized solution will include interconnection with (external) data networks. Data networks can be physical or virtual networks.

Within the context of edge computing, 5GLAN will make it possible to have 5G devices share a common IP address space with servers deployed in a mini/micro-datacenter. While a similar result could be achieved using an overlay solution terminated on the 5G device, using 5GLAN in this case makes it possible to benefit from 5G network features such as session continuity support, fine-grained QoS support, user and device authentication, and to make a more efficient use of the air interface.

1.3. Goals of this Document

The goals of this document are to describe:

New use cases for NVO3 and 5GLAN, related to interconnections between 5G networks and datacenters.

A more detailed discussion of the architecture and requirements of this interconnection.

Our primary scenario will be a virtual network domain located outside of the 5G domain (a "data network" in 3GPP terminology), that can be joined by 5G devices. It is NOT a goal of the present document to cover inter-UPF connectivity inside the 5G domain, which 3GPP intends to standardize in 2019.

2. New Use Case for NVO3 and 5GLAN: 5G/Datacenter Interconnection

In addition to already known base 5GLAN use cases (see Section 1.1), interconnection of 5GLAN with datacenters will enable new scenarios.

Support for Virtualization on 5G End Devices: 5G devices may be used as servers in a "mobile data center", or to extend a traditional/fixed data center. This involves VM hosting on 5G devices, and VM

mobility between 5G devices, or between 5G devices and DC servers and enables, for example:

Transparent mobility of Fog RAN [I-D.bernardos-sfc-fog-ran] components between 5G devices and micro-datacenters,

Transparent offloading of application tasks towards the distant or edge cloud, or towards other 5G devices.

As discussed in Section 4, VM hosting on 5G devices under the control of a NVO3 network operator can bring new requirements on 5G networks interface with the datacenter data networks, including exposing a "tenant system interface" identity and state information, supporting adding/removing addresses, and supporting hot VM migration.

End-to-End Redundancy: 5G devices may connect to a 5GLAN virtual network over several paths, using active-active or active-passive configurations. For example, a 5G device running a critical application may use both WLAN and a cellular link to increase availability. Today, this type of connection are defined in 5G but are using a same anchor UPF for both links, which limits the scope of redundancy to the 5G network. Instead, path redundancy could be prolonged beyond the 5G network (e.g. using a different anchor for each path), into the datacenter.

3. Architecture Overview

A high level architecture view of the system is represented in Figure 1 (based on our interpretation of the conclusions of [3GPP.23.734]).

In the data plane, the end device (user equipment) is connected point-to-point to an anchor gateway (UPF), through the radio access network and possibly through intermediate UPFs (not shown here). This point-to-point connection is called PDU session in 5G. In usual non-5GLAN communication use cases, IP or Ethernet packets are carried over a tunnel between the end device and the anchor UPF, decapsulated by the UPF and forwarded over a data network. In the 5GLAN case, the decapsulated packet should be tunneled/forwarded from the anchor UPF towards a remote virtualization edge, or another anchor UPF, which decapsulates and forwards the packet towards its destination end device. (Except in the simpler case where source and destination end devices are served by the same UPF.)

The section of network between anchor UPFs in the diagram is a datacenter VPN domain ("L2/L3 VPN domain"), with its own control and data plane. Anchor UPFs may be directly interconnected inside the 5G

network as well, for internal 5GLAN traffic (although it is not represented here).

In the control plane, 5G end device connectivity is today supported by the Access and Mobility Management Function (AMF) and Session Management Function (SMF). 5GLAN specific control plane support for a given 5GLAN network (e.g. to configure UPFs, and perform access control) will be implemented inside a single SMF.

There should be an interconnection between the 5G network and the L2/L3 VPN domain, in the control and/or management plane.

In the data plane, an edge function collocated or interconnected with the UPF is acting as a gateway between the 3GPP and L2/L3 VPN domain. This edge function corresponds to "provider edge" device in VPN terminology.

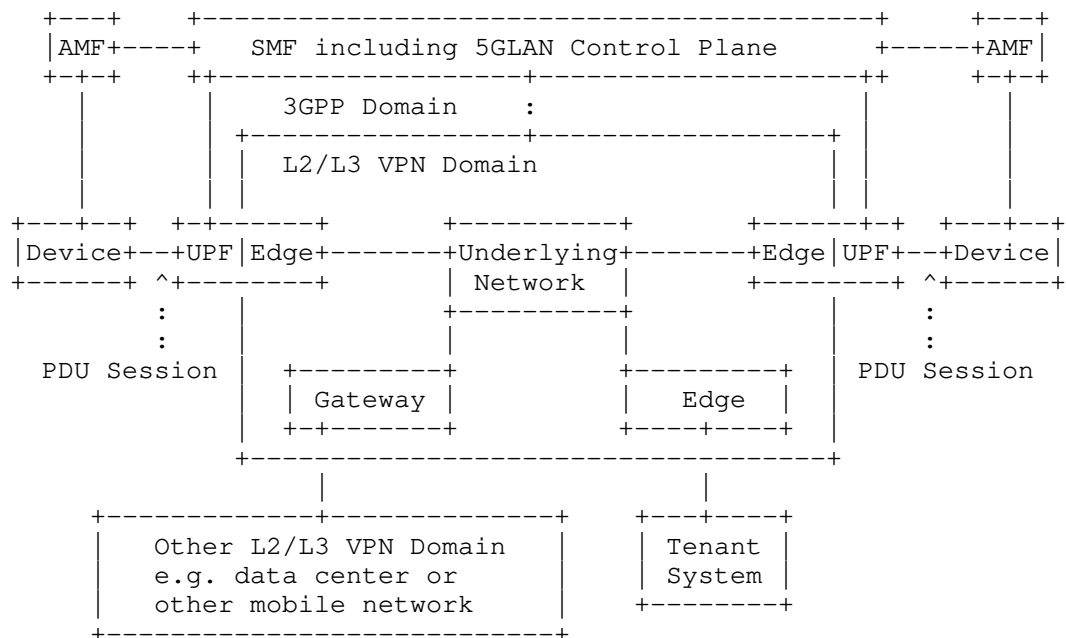


Figure 1: 5GLAN Network Interconnected with a L2/L3 VPN Domain

We will focus on NVO3 as the datacenter VPN technology. Nevertheless, applicability of other virtualization technologies to 5GLAN may be studied as well in future revisions of this document.

The 5GLAN architecture can be made to integrate with the NVO3 architecture [RFC8014], where:

A 5G end device corresponds to an end device in NV03.

A 5G edge/UPF corresponds to an external NVE in NVO3 (the edge/UPF can encapsulate packets in network virtualization headers, as does the external NVE in NVO3, to avoid carrying those extra headers over the wireless link).

The PDU session in 5G corresponds to the VLAN connection between an end device and an external NVE (i.e. both are point-to-point connections).

An overview of the integration of NVO3 and the 5G network for 5GLAN is displayed in Figure 2

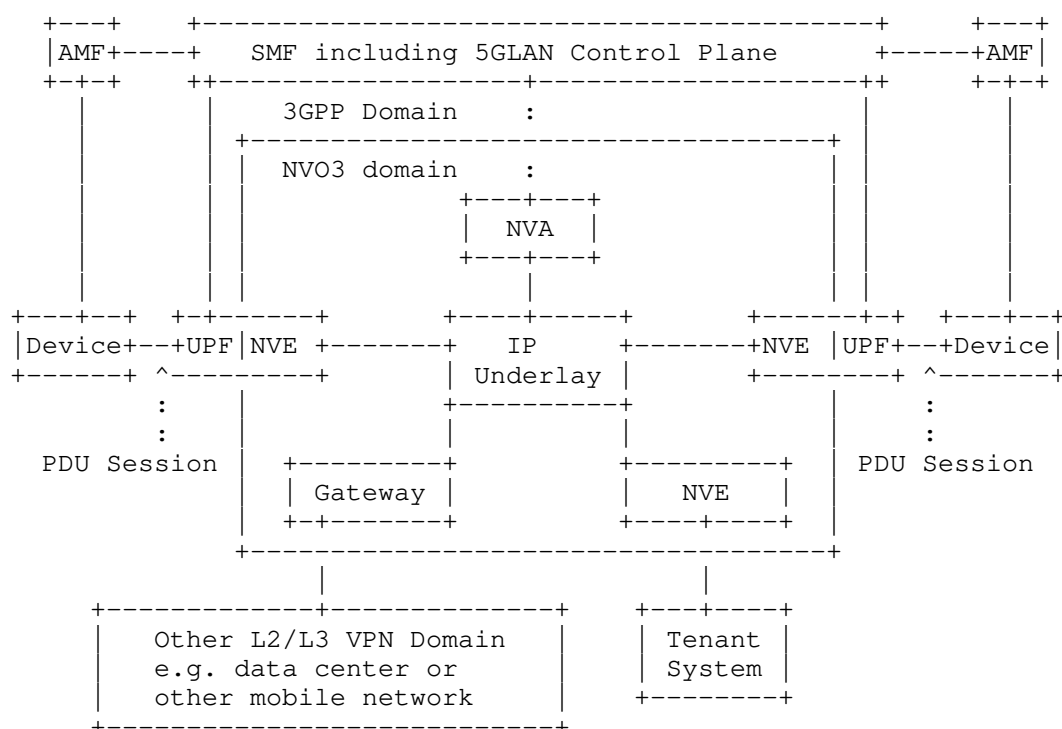


Figure 2: 5GLAN Network Interconnected with a NV03 Domain

4. Major Features of a 5G/Datacenter Interconnection

The following discusses VPN-5G interconnection functionalities.

L2/L3 VPN: To support both IP-based and Ethernet-based 5GLANs, the L2/L3 VPN domain should provide L2 and L3 VPN services between

provider edges. NVO3 can support L2 and L3 VPNs over an IP overlay. For example, EVPN may be used in L2 case, as described in [I-D.ietf-nvo3-evpn-applicability].

VM Hosting and VM Mobility: The goal of this feature is to have the NVO3 network operator control connectivity for all VMs, including VMs hosted on 5G devices. Based on an analysis of End Device-to-NVE Control Protocol requirements [RFC8394] in the context of 5G LANs, here is a summary of potential requirements on 5G-NVO3 interfaces:

The concept of tenant system interface (TSI) identifies the connection to a single VM (e.g. it corresponds to a single VLAN tag on hypervisor-NVE connection in NVO3). This concept may also be useful to support VM migration. 5GLAN could for example restrict traffic to/from a single tenant system (e.g. a VM) to a single PDU session, and associate a TSI identifier to the connection, exposed to the L2/L3 VPN domain.

End Devices can communicate tenant system interface state information (associated and activated), which corresponds to different phases of a VM lifetime. Such information may therefore be carried over a PDU session to enable similar operations in 5GLAN.

A tenant system (e.g. VM) can add/remove IP/MAC addresses dynamically even after End Device-to-NVE connection is made for this tenant system.

The external NVE (i.e. edge/UPF in 5GLAN context) can dynamically initiate the deactivation or de-association of a MAC/IP address.

Hot and cold VM mobility may be supported. The 5G network should indicate when an event is caused by a hot VM migration event.

Active-active and active-passive redundant path to a 5G device (through multiple edge/UPFs) may be supported, to provide end-to-end path redundancy. To support this, the 5G network should expose reachability information towards a given IP or MAC address through multiple UPFs. Priority information may be exposed as well, to enable active-passive redundancy.

End Device Mobility and Session Continuity: End device mobility between anchor UPFs may be similar to hot VM migration events, although they occur more often and affect all hosted VMs on a device. They may also have more stringent requirements in term of packet loss and latency since, as opposed to the VM migration case, end device mobility requires no transfer of state.

Mobility requirements can vary: for example 5G devices using a fixed anchor ("session continuity mode (SSC) 1") will not expose any mobility event to the L2/L3 VPN domain. Nevertheless, in cases where mobility and low latency are required, break-before-make (in SSC mode 2) or make-before-break mobility (in SSC mode 3) events may be exposed to the L2/L3 VPN domain.

QoS: 5GLAN networks can be applied a wide range of QoS inside the 5G network, including ultra-low latency. Nevertheless, the level of QoS depends on the application, and therefore the level of QoS to apply to traffic to/from 5G devices in the L2/L3 VPN is not known at this time.

QoS mechanisms to be supported in the L2/L3 VPN domain can include best-effort, differentiated services, traffic engineered links, deterministic networking. Some form of coordination may be therefore needed between 5G and the L2/L3 VPN domain in control and/or management plane, e.g. to setup proper traffic engineering associated with NVO3 overlay networks (e.g. create/modify/release underlying TE paths when an end device changes its attachment point from one edge/UPF to another).

Privacy: Privacy is required for communication between end devices. The L2/L3 VPN, including the edge/UPF, should therefore support a secure protocol over the VPN domain (e.g. including encryption). Encryption of NVO3 traffic over the underlying network (e.g. using IPSec between NVEs) is mentioned in [RFC8014].

Access Control: Access control of end devices can be performed by the 3GPP domain (e.g. at network registration and later when creating the PDU session). Some form of cooperation between the 5G network and the L2/L3 VPN domain may be needed to authenticate the 5G subscription in the L2/L3 VPN domain (e.g. through the exposition of a subscription identifier).

Other Remarks: As already mentioned, UPFs can be directly interconnected with each other for internal 5GLAN communication. LAN communication between 5G devices may therefore entirely bypass the L2/L3 VPN.

Similarly, although device-to-device communication is currently not defined in 3GPP for 5GLAN, in the future it may also be leveraged to bypass the L2/L3 VPN for direct communication between devices.

A 5G device can become inactive, and may be paged/awaken when there is outstanding traffic for this device. This will be

handled entirely in the 3GPP system (traffic will be buffered at UPF and device will be paged).

5. IANA Considerations

This document requests no IANA actions.

6. Security Considerations

From the 5G operator perspective, traffic sent over the L2/L3 VPN domain should be secured against being misdelivered, being modified, or having its content exposed to an inappropriate third party. This requirement is also found in NVO3.

Additionally, 5G devices wishing to join a virtual network deployed in the L2/L3 VPN domain will need to be authenticated and authorized for joining. Mutual authentication and authorization between 5G devices and virtual networks may be needed and may be supported through coordination between the 5G network, which authenticated the 5G device, and the L2/L3 VPN domains.

7. Next Steps

We would like to propose this use case for further discussion and possibly adoption in a RTG working group such as NVO3 or RTGWG, as a new use case for datacenter networking.

At this time we do not expect a change in NVO3 protocols. On the other side, discussions at the IETF can provide valuable input to justify and drive any future enhancement to 5G networks, and align with IETF datacenter protocols (e.g. what information and operations should be made available to datacenter networks).

8. Informative References

[_3GPP.22.261]

3GPP, "Service requirements for next generation new services and markets", 3GPP TS 22.261,
<<http://www.3gpp.org/ftp/Specs/html-info/22261.htm>>.

[_3GPP.22.821]

3GPP, "Feasibility Study on LAN Support in 5G", 3GPP TR 22.821,
<<http://www.3gpp.org/ftp/Specs/html-info/22821.htm>>.

- [_3GPP.23.501]
3GPP, "System Architecture for the 5G System", 3GPP TS 23.501,
<<http://www.3gpp.org/ftp/Specs/html-info/23501.htm>>.
- [_3GPP.23.734]
3GPP, "Study on enhancement of 5GS for Vertical and LAN Services", 3GPP TR 23.734,
<<http://www.3gpp.org/ftp/Specs/html-info/23734.htm>>.
- [_3GPP.33.819]
3GPP, "Study on security enhancements of 5GS for vertical and Local Area Network (LAN) services", 3GPP TR 33.819,
<<http://www.3gpp.org/ftp/Specs/html-info/33819.htm>>.
- [I-D.bernardos-sfc-fog-ran]
Bernardos, C., Rahman, A., and A. Mourad, "Service Function Chaining Use Cases in Fog RAN", draft-bernardos-sfc-fog-ran-04 (work in progress), September 2018.
- [I-D.ietf-nvo3-evpn-applicability]
Rabadan, J., Bocci, M., Boutros, S., and A. Sajassi, "Applicability of EVPN to NVO3 Networks", draft-ietf-nvo3-evpn-applicability-01 (work in progress), October 2018.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016,
<<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8394] Li, Y., Eastlake 3rd, D., Kreeger, L., Narten, T., and D. Black, "Split Network Virtualization Edge (Split-NVE) Control-Plane Requirements", RFC 8394, DOI 10.17487/RFC8394, May 2018,
<<https://www.rfc-editor.org/info/rfc8394>>.

Appendix A. 5GLAN Background Information

The 5G architecture is defined by 3GPP in [_3GPP.23.501], currently as part of release 15.

5GLAN is a new feature developed as part of release 16. Its requirements are currently being specified in [_3GPP.22.261] (based on results from an earlier study on requirements in [_3GPP.22.821]).

The architecture of 5GLAN has been studied in [_3GPP.23.734], along with other subjects. A specification phase for the 5GLAN

architecture will likely follow. Conclusions of the study included the following:

5GLAN traffic (IP or Ethernet traffic between a restricted set of "5GLAN group member" devices) will be transported between 5G devices and their anchor UPF. Anchor UPFs will forward this traffic, as applicable, to/from (1) a data network, (2) another anchor UPF, or (3) other devices using local forwarding through the anchor UPF, when devices are connected to the same anchor. In case (2), direct traffic between anchor UPFs will be encapsulated in per-5GLAN group tunnels.

In the control plane, a single SMF will handle connectivity of all devices connected to the same 5GLAN.

The user plane can be centralized (single anchor UPF involved in a single 5GLAN) or distributed (multiple anchor UPFs involved in a single 5GLAN).

Security aspects related to 5GLAN are currently studied in [3GPP.33.819].

Authors' Addresses

Xavier de Foy
InterDigital Communications, LLC
1000 Sherbrooke West
Montreal H3A 3G4
Canada

Email: Xavier.Defoy@InterDigital.com

Ulises Olvera-Hernandez
InterDigital Communications, LLC
64 Great Eastern Street
London EC2A 3QR
England

Email: Ulises.Olvera-Hernandez@InterDigital.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

J. Gross, Ed.
I. Ganga, Ed.
Intel
T. Sridhar, Ed.
VMware
March 11, 2019

Geneve: Generic Network Virtualization Encapsulation
draft-ietf-nvo3-geneve-12

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 document describes Geneve, an encapsulation protocol designed to recognize and accommodate these changing capabilities and needs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Requirements Language	4
1.2.	Terminology	4
2.	Design Requirements	5
2.1.	Control Plane Independence	6
2.2.	Data Plane Extensibility	7
2.2.1.	Efficient Implementation	7
2.3.	Use of Standard IP Fabrics	8
3.	Geneve Encapsulation Details	9
3.1.	Geneve Packet Format Over IPv4	9
3.2.	Geneve Packet Format Over IPv6	10
3.3.	UDP Header	12
3.4.	Tunnel Header Fields	13
3.5.	Tunnel Options	14
3.5.1.	Options Processing	16
4.	Implementation and Deployment Considerations	17
4.1.	Applicability Statement	17
4.2.	Congestion Control Functionality	18
4.3.	UDP Checksum	18
4.3.1.	UDP Zero Checksum Handling with IPv6	19
4.4.	Encapsulation of Geneve in IP	20
4.4.1.	IP Fragmentation	20
4.4.2.	DSCP, ECN and TTL	21
4.4.3.	Broadcast and Multicast	22
4.4.4.	Unidirectional Tunnels	22
4.5.	Constraints on Protocol Features	23
4.5.1.	Constraints on Options	23
4.6.	NIC Offloads	24
4.7.	Inner VLAN Handling	24
5.	Interoperability Issues	25
6.	Security Considerations	25
6.1.	Data Confidentiality	26
6.1.1.	Inter-Data Center Traffic	26
6.2.	Data Integrity	27
6.3.	Authentication of NVE peers	27
6.4.	Options Interpretation by Transit Devices	28

6.5. Multicast/Broadcast	28
6.6. Control Plane Communications	28
7. IANA Considerations	28
8. Contributors	29
9. Acknowledgements	30
10. References	31
10.1. Normative References	31
10.2. Informative References	32
Authors' Addresses	35

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] (Virtual eXtensible Local Area Network) and NVGRE [RFC7637] (Network Virtualization Using Generic Routing Encapsulation), 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] (A Scalable and Flexible Data Center Network) and the NVO3 Data Plane Requirements [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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 (Network Interface Controller) 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 of packets within a flow.

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

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 Controller. Also called as Network Interface Card or Network Adapter. 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.

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 packet 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, tunnel 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. Tunnel endpoints are frequently a component of an NVE (Network Virtualization Edge) 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. Many of the concepts of network virtualization overlays over Layer 3 IP networks are described in NVO3 Framework framework [RFC7365]. 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.

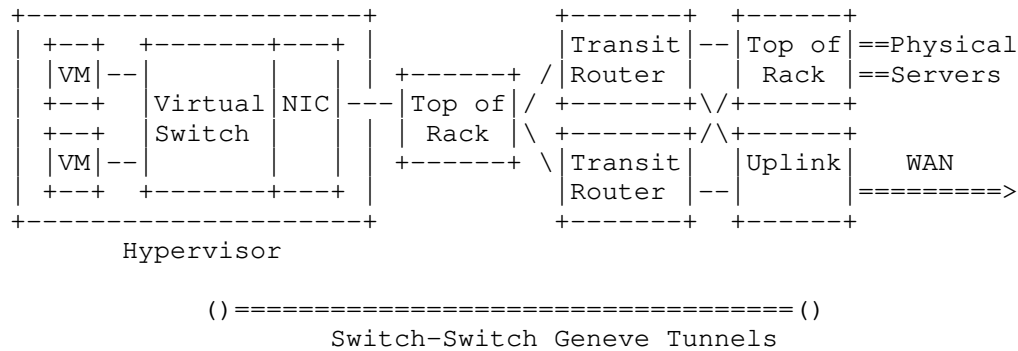


Figure 1: Sample Geneve Deployment

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 VXLAN spec prescribed a multicast learning-based control plane), these specifications have largely been treated as describing only the data format. The VXLAN packet 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 is 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 reduces the chance of obsolescence 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 packets, they are equally important on data packets 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 packets 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 a desired 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.

Tunnel 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 tunnel 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 tunnel endpoints, supporting options can be designed using ordering restrictions and other techniques to ease parsing.

Options, if present in the packet, MUST only be generated and terminated by tunnel endpoints. Transit devices MAY be able to interpret the options, however, as non-terminating devices, transit devices do not originate or terminate the Geneve packet, hence MUST NOT modify Geneve headers and MUST NOT insert or delete options, which is the responsibility of tunnel endpoints. The participation of transit devices in interpreting options 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 tunnel endpoint addresses are available for hashing.

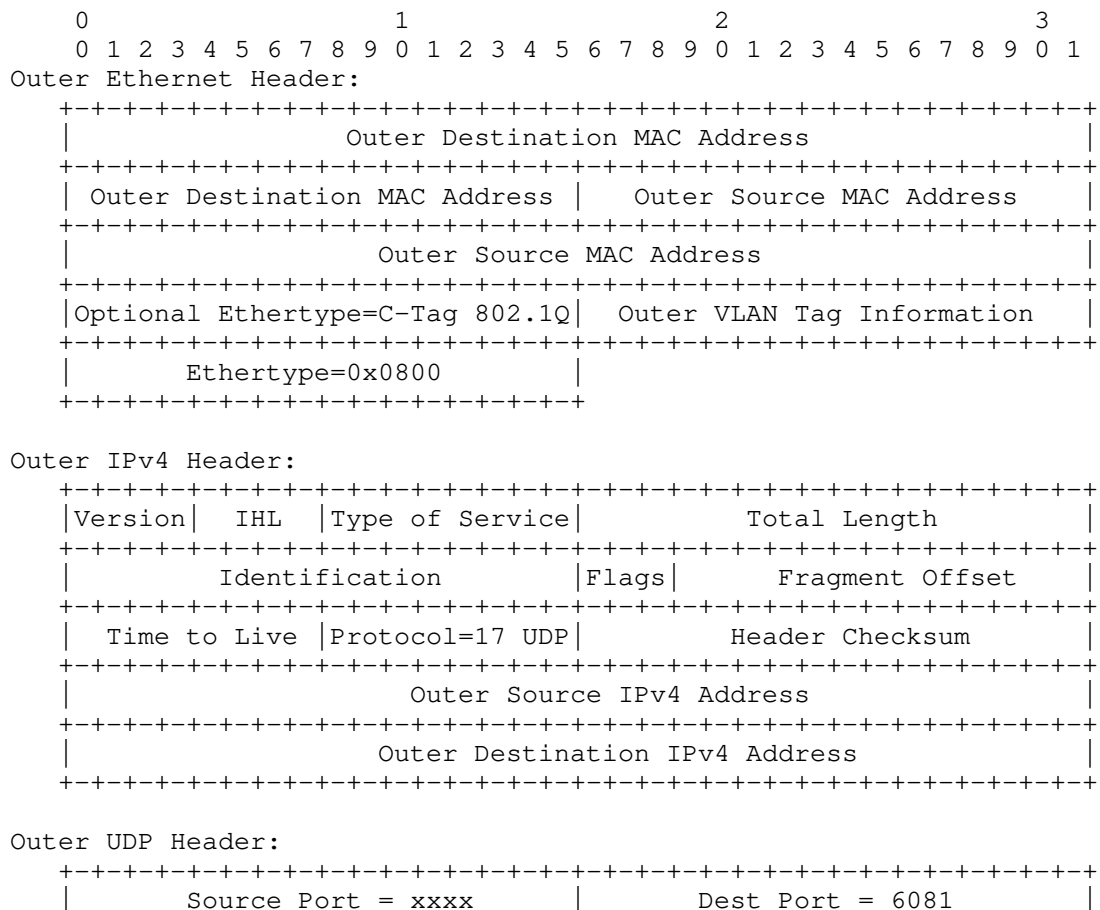
Since it is desirable for Geneve to perform well on these existing fabrics, it is necessary for entropy from encapsulated packets to be exposed in the tunnel header. The most common technique for this is

to use the UDP source port, which is discussed further in Section 3.3.

3. Geneve Encapsulation Details

The Geneve packet 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 packet format transported (for example) over Ethernet along with an Ethernet payload.

3.1. Geneve Packet Format Over IPv4



```

+-----+-----+
|                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 Packet 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: In order to protect the Geneve header, options and payload from potential data corruption, UDP checksum SHOULD be generated as specified in [RFC0768] and [RFC1112] when Geneve is encapsulated in IPv4. To protect the IP header, Geneve header, options and payload from potential data corruption, the UDP checksum MUST be generated by default as specified in [RFC0768] and [RFC2460] when Geneve is encapsulated in IPv6. Upon receiving such packets with non-zero UDP checksum, the receiving tunnel endpoints MUST validate the checksum. If the checksum is not correct, the packet MUST be dropped, otherwise the packet MUST be accepted for decapsulation.

Under certain conditions, the UDP checksum MAY be set to zero on transmit for packets encapsulated in both IPv4 and IPv6 [RFC6935]. See Section 4.3 for additional requirements that apply for using zero UDP checksum with IPv4 and IPv6. Disabling the use of UDP checksums is an operational consideration that should take into account the risks and effects of packet corruption.

3.4. Tunnel Header Fields

Ver (2 bits): The current version number is 0. Packets received by a tunnel endpoint with an unknown version MUST be dropped. Transit devices interpreting 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): Control packet. This packet contains a control message. Control messages are sent between tunnel endpoints. Tunnel Endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret it. Since these are infrequent control messages, it is RECOMMENDED that tunnel 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 tunnel endpoints where option parsing is not supported the packet 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 packet. Transit devices MUST NOT drop packets on the basis of this bit.

The critical bit allows hardware implementations the flexibility to handle options processing in the hardware fastpath or in the exception (slow) path without the need to process all the options. For example, a critical option such as secure hash to provide Geneve header integrity check must be processed by tunnel endpoints and typically processed in the hardware fastpath.

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

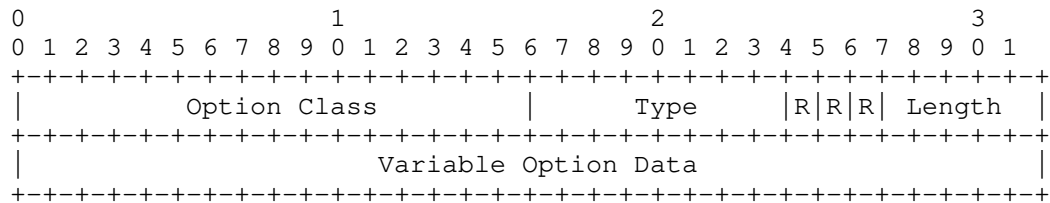
Protocol Type (16 bits): The type of the protocol data unit appearing after the Geneve header. This follows the EtherType [ETYPES] convention with Ethernet itself being represented by the value 0x6558.

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



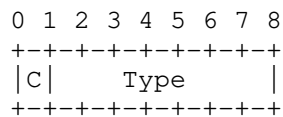
Geneve Option

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 tunnel endpoint does not recognize this option and this bit is set then the packet MUST be dropped. If the 'C' bit (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:



The requirement to drop a packet with an unknown option with the 'C' bit set 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. A value of 0 in the Length field implies an option with only the option header without the variable option data. 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 a tunnel endpoint that processes the options.

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

3.5.1. Options Processing

Geneve options are intended to be originated and processed by tunnel endpoints. However, options MAY be interpreted by transit devices along the tunnel path. Transit devices not interpreting Geneve headers (that may or may not include options) MUST handle Geneve packets as any other UDP packet 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 some requirements are imposed on options and the devices that process them:

- o Receiving tunnel endpoints MUST drop packets containing unknown options with the 'C' bit set in the option type. Conversely, transit devices MUST NOT drop packets as a result of encountering unknown options, including those with the 'C' bit set.
- o Some options may be defined in such a way that the position in the option list is significant. Options MUST NOT be changed by transit devices.
- o An option SHOULD NOT be dependent upon any other option in the packet, i.e., options can be processed independent of one another. An option MUST NOT affect the parsing or interpretation of any other option. However, option processing by tunnel endpoints may

result in the packet being dropped. Options may also be used in conjunction with each other or combined with packet data but this processing is done above the encapsulation layer.

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.

Options may be processed by NIC hardware utilizing offloads (e.g. LSO and LRO) as described in Section 4.6. Careful consideration should be given to how the offload capabilities outlined in Section 4.6 impact an option's design.

4. Implementation and Deployment Considerations

4.1. Applicability Statement

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It is intended for use in public or private data center environments, for deploying multi-tenant overlay networks over an existing IP underlay network.

Geneve is a UDP based encapsulation protocol transported over existing IPv4 and IPv6 networks. Hence, as a UDP based protocol, Geneve adheres to the UDP usage guidelines as specified in [RFC8085]. The applicability of these guidelines are dependent on the underlay IP network and the nature of Geneve payload protocol (example TCP/IP, IP/Ethernet).

[RFC8085] outlines two applicability scenarios for UDP applications, 1) general Internet and 2) controlled environment. The controlled

environment means a single administrative domain or adjacent set of cooperating domains. A network in a controlled environment can be managed to operate under certain conditions whereas in general Internet this cannot be done. Hence requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements of general internet.

Geneve is intended to be deployed in a data center network environment operated by a single operator or adjacent set of cooperating network operators that fits with the definition of controlled environments in [RFC8085].

For the purpose of this document, a traffic-managed controlled environment (TMCE) is defined as an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion. The concept of TMCE is outlined in [RFC8086]. Significant portions of text in Section 4.1 through Section 4.3 are based on [RFC8086] as applicable to Geneve.

It is the responsibility of the operator to ensure that the guidelines/requirements in this section are followed as applicable to their Geneve deployment(s).

4.2. Congestion Control Functionality

Geneve does not natively provide congestion control functionality and relies on the payload protocol traffic for congestion control. As such Geneve MUST be used with congestion controlled traffic or within a network that is traffic managed to avoid congestion (TMCE). An operator of a traffic managed network (TMCE) may avoid congestion by careful provisioning of their networks, rate-limiting of user data traffic and traffic engineering according to path capacity.

4.3. UDP Checksum

In order to provide integrity of Geneve headers, options and payload, for example to avoid mis-delivery of payload to different tenant systems in case of data corruption, outer UDP checksum SHOULD be used with Geneve when transported over IPv4. An operator MAY choose to disable UDP checksum and use zero checksum if Geneve packet integrity is provided by other data integrity mechanisms such as IPsec or additional checksums or if one of the conditions in Section 4.3.1 a, b, c are met.

By default, UDP checksum MUST be used when Geneve is transported over IPv6. A tunnel endpoint MAY be configured for use with zero UDP checksum if additional requirements in Section 4.3.1 are met.

4.3.1. UDP Zero Checksum Handling with IPv6

When Geneve is used over IPv6, UDP checksum is used to protect IPv6 headers, UDP headers and Geneve headers, options and payload from potential data corruption. As such by default Geneve MUST use UDP checksum when transported over IPv6. An operator MAY choose to configure to operate with zero UDP checksum if operating in a traffic managed controlled environment as stated in Section 4.1 if one of the following conditions are met.

- a. It is known that the packet corruption is exceptionally unlikely (perhaps based on knowledge of equipment types in their underlay network) and the operator is willing to take a risk of undetected packet corruption
- b. It is judged through observational measurements (perhaps through historic or current traffic flows that use non zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected corruption.
- c. Geneve payload is carrying applications that are tolerant of misdelivered or corrupted packets (perhaps through higher layer checksum validation and/or reliability through retransmission)

In addition Geneve tunnel implementations using Zero UDP checksum MUST meet the following requirements:

1. Use of UDP checksum over IPv6 MUST be the default configuration for all Geneve tunnels.
2. If Geneve is used with zero UDP checksum over IPv6 then such tunnel endpoint implementation MUST meet all the requirements specified in section 4 of [RFC6936] and requirements 1 as specified in section 5 of [RFC6936].
3. The Geneve tunnel endpoint that decapsulates the tunnel SHOULD check the source and destination IPv6 addresses are valid for the Geneve tunnel that is configured to receive Zero UDP checksum and discard other packets for which such check fails.
4. The Geneve tunnel endpoint that encapsulates the tunnel MAY use different IPv6 source addresses for each Geneve tunnel that uses Zero UDP checksum mode in order to strengthen the decapsulator's check of the IPv6 source address (i.e the same IPv6 source address is not to be used with more than one IPv6 destination address, irrespective of whether that destination address is a unicast or multicast address). When this is not possible, it is

RECOMMENDED to use each source address for as few Geneve tunnels that use zero UDP checksum as is feasible.

5. Measures SHOULD be taken to prevent Geneve traffic over IPv6 with zero UDP checksum from escaping into the general Internet. Examples of such measures include employing packet filters at the Gateways or edge of Geneve network and/or keeping logical or physical separation of Geneve network from networks carrying General Internet.

The above requirements do not change either the requirements specified in [RFC2460] as modified by [RFC6935] or the requirements specified in [RFC6936].

The requirement to check the source IPv6 address in addition to the destination IPv6 address, plus the recommendation against reuse of source IPv6 addresses among Geneve tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

Editorial Note (The following paragraph to be removed by the RFC Editor before publication)

It was discussed during TSVART early review if the level of requirement for using different IPv6 source addresses for different tunnel destinations would need to be "MAY" or "SHOULD". The discussion concluded that it was appropriate to keep this as "MAY", since it was considered not realistic for control planes having to maintain a high level of state on a per tunnel destination basis. In addition, the text above provides sufficient guidance to operators and implementors on possible mitigations.

4.4. 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.

4.4.1. IP Fragmentation

It is strongly RECOMMENDED that Path MTU Discovery ([RFC1191], [RFC8201]) 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 tunnel endpoint with soft-state about the link that it may use to prevent or minimize fragmentation depending on its role in the virtualized network. The NVE control plane MAY use configuration mechanism or path discovery information to maintain the MTU size of the tunnel link(s) associated with the tunnel endpoint, so if a tenant system sends large packets that when encapsulated exceed the MTU size of the tunnel link, the tunnel endpoint can discard such packets and send exception messages to the tenant system(s). If the tunnel endpoint is associated with a routing or forwarding function and/or has the capability to send ICMP messages, the encapsulating tunnel endpoint MAY send ICMP fragmentation needed [RFC0792] or Packet Too Big [RFC4443] messages to the tenant system(s). For example, recommendations/guidance for handling fragmentation in similar overlay encapsulation services like PWE3 are provided in section 5.3 of [RFC3985].

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. For example, some of the issues associated with MTU size and fragmentation in IP tunneling and use of ICMP messages is outlined in section 4.2 of [I-D.ietf-intarea-tunnels].

Editorial Note (The following paragraph to be removed by the RFC Editor before publication)

It was discussed during TSVART early review if the level of requirement for maintaining tunnel MTU at the ingress has to be "MAY" or "SHOULD". The discussion concluded that it was appropriate to leave this as "MAY", considering the high level of state to be maintained.

4.4.2. DSCP, ECN and TTL

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

[RFC2983] provides guidance 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 MUST be followed for IP packets encapsulated in Geneve.

Though Uniform or Pipe models could be used for TTL (or Hop Limit in case of IPv6) handling when tunneling IP packets, Pipe model is more aligned with network virtualization. [RFC2003] provides guidance on handling TTL between inner IP header and outer IP tunnels; this model is more aligned with the Pipe model and is recommended for use with Geneve for network virtualization applications.

4.4.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two tunnel 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.

In addition, [RFC8293] provides examples of various mechanisms that can be used for multicast handling in network virtualization overlay networks.

4.4.4. Unidirectional Tunnels

Generally speaking, a Geneve tunnel is a unidirectional concept. IP is not a connection oriented protocol and it is possible for two tunnel 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.

4.5. 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.

4.5.1. Constraints on Options

While Geneve options are more flexible, a control plane may restrict the number of option TLVs as well as the order and size of the TLVs, between tunnel endpoints, to make it simpler for a data plane implementation in software or hardware to handle [I-D.ietf-nvo3-encap]. For example, there may be some critical information such as a secure hash that must be processed in a certain order to provide lowest latency.

A control plane may negotiate a subset of option TLVs and certain TLV ordering, as well may limit the total number of option TLVs present in the packet, for example, to accommodate hardware capable of processing fewer options [I-D.ietf-nvo3-encap]. Hence, a control plane needs to have the ability to describe the supported TLVs subset and their order to the tunnel endpoints. In the absence of a control plane, alternative configuration mechanisms may be used for this purpose. The exact mechanism is not defined in this document.

4.6. 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 Options **MUST NOT** be reordered during the course of offload processing, including when merging packets for the purpose of LRO.
- o NICs performing offloads **MUST NOT** drop packets with unknown options, including those marked as critical, unless explicitly configured.

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.

4.7. 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 tunnel 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 packets. 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 most common protocols used for network virtualization (VXLAN,NVGRE) 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 packet formats supporting the same set of capabilities, there is no need for a hard transition - tunnel 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 packets 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 encapsulated within a UDP/IP packet, Geneve does not have any inherent security mechanisms. As a result, an attacker with access to the underlay network transporting the IP packets has the ability to snoop or inject packets. Compromised 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 service 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.

Geneve does not otherwise affect the security of the encapsulated packets. As per the guidelines of BCP 72 [RFC3552], the following sections describe potential security risks that may be applicable to Geneve deployments and approaches to mitigate such risks. It is also noted that not all such risks are applicable to all Geneve deployment scenarios, i.e., only a subset may be applicable to certain deployments. So an operator has to make an assessment based on their network environment and determine the risks that are applicable to their specific environment and use appropriate mitigation approaches as applicable.

6.1. Data Confidentiality

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It can be used to deploy multi-tenant overlay networks over an existing IP underlay network in a public or private data center. The overlay service is typically provided by a service provider, for example a cloud services provider or a private data center operator, this may or not may be the same provider as an underlay service provider. Due to the nature of multi-tenancy in such environments, a tenant system may expect data confidentiality to ensure its packet data is not tampered with (active attack) in transit or a target of unauthorized monitoring (passive attack). A tenant may expect the overlay service provider to provide data confidentiality as part of the service or a tenant may bring its own data confidentiality mechanisms like IPsec or TLS to protect the data end to end between its tenant systems.

If an operator determines data confidentiality is necessary in their environment based on their risk analysis, for example as in multi-tenant environments, then an encryption mechanism SHOULD be used to encrypt the tenant data end to end between the NVEs. The NVEs may use existing well established encryption mechanisms such as IPsec, DTLS, etc.

6.1.1. Inter-Data Center Traffic

A tenant system in a customer premises (private data center) may want to connect to tenant systems on their tenant overlay network in a public cloud data center or a tenant may want to have its tenant systems located in multiple geographically separated data centers for high availability. Geneve data traffic between tenant systems across such separated networks should be protected from threats when

traversing public networks. Any Geneve overlay data leaving the data center network beyond the operator's security domain SHOULD be secured by encryption mechanisms such as IPsec or other VPN mechanisms to protect the communications between the NVEs when they are geographically separated over untrusted network links. Specification of data protection mechanisms employed between data centers is beyond the scope of this document.

6.2. Data Integrity

Geneve encapsulation is used between NVEs to establish overlay tunnels over an existing IP underlay network. In a multi-tenant data center, a rogue or compromised tenant system may try to launch a passive attack such as monitoring the traffic of other tenants, or an active attack such as trying to inject unauthorized Geneve encapsulated traffic such as spoofing, replay, etc., into the network. To prevent such attacks, an NVE MUST NOT propagate Geneve packets beyond the NVE to tenant systems and SHOULD employ packet filtering mechanisms so as not to forward unauthorized traffic between TSs in different tenant networks.

A compromised network node or a transit device within a data center may launch an active attack trying to tamper with the Geneve packet data between NVEs. Malicious tampering of Geneve header fields may cause the packet from one tenant to be forwarded to a different tenant network. If an operator determines the possibility of such threat in their environment, the operator may choose to employ data integrity mechanisms between NVEs. In order to prevent such risks, a data integrity mechanism SHOULD be used in such environments to protect the integrity of Geneve packets including packet headers, options and payload on communications between NVE pairs. A cryptographic data protection mechanism such as IPsec may be used to provide data integrity protection. A data center operator may choose to deploy any other data integrity mechanisms as applicable and supported in their underlay networks.

6.3. Authentication of NVE peers

A rogue network device or a compromised NVE in a data center environment might be able to spoof Geneve packets as if it came from a legitimate NVE. In order to mitigate such a risk, an operator SHOULD use an authentication mechanism, such as IPsec to ensure that the Geneve packet originated from the intended NVE peer, in environments where the operator determines spoofing or rogue devices is a potential threat. Other simpler source checks such as ingress filtering for VLAN/MAC/IP address, reverse path forwarding checks, etc., may be used in certain trusted environments to ensure Geneve packets originated from the intended NVE peer.

6.4. Options Interpretation by Transit Devices

Options, if present in the packet, are generated and terminated by tunnel endpoints. As indicated in Section 2.2.1, transit devices may interpret the options. However, if the packet is protected by tunnel endpoint to tunnel endpoint encryption, for example through IPsec, transit devices will not have visibility into the Geneve header or options in the packet. In such cases transit devices MUST handle Geneve packets as any other IP packet and maintain consistent forwarding behavior. In cases where options are interpreted by transit devices, the operator MUST ensure that transit devices are trusted and not compromised. Implementation of a mechanism to ensure this trust is beyond the scope of this document.

6.5. Multicast/Broadcast

In typical data center networks where IP multicasting is not supported in the underlay network, multicasting may be supported using multiple unicast tunnels. The same security requirements as described in the above sections can be used to protect Geneve communications between NVE peers. If IP multicasting is supported in the underlay network and the operator chooses to use it for multicast traffic among tunnel endpoints, then the operator in such environments may use data protection mechanisms such as IPsec with Multicast extensions [RFC5374] to protect multicast traffic among Geneve NVE groups.

6.6. Control Plane Communications

A Network Virtualization Authority (NVA) as outlined in [RFC8014] may be used as a control plane for configuring and managing the Geneve NVEs. The data center operator is expected to use security mechanisms to protect the communications between the NVA to NVEs and use authentication mechanisms to detect any rogue or compromised NVEs within their administrative domain. Data protection mechanisms for control plane communication or authentication mechanisms between the NVA and the NVEs is beyond the scope of this document.

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 <jesse@kernel.org>
Contact: Jesse Gross <jesse@kernel.org>
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 [RFC8126] and 0xFFFF0-0xFFFF 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 (OVS)
0x0102	Open Virtual Networking (OVN)
0x0103	In-band Network Telemetry (INT)
0x0104	VMware, Inc.
0x0105	Amazon.com, Inc.
0x0106	Cisco Systems, Inc.
0x0107	Oracle Corporation
0x0108..0x110	Amazon.com, Inc.
0x0111..0xFFEF	Unassigned - First Come First Served
0xFFFF0..FFFF	Experimental

8. Contributors

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

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

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

Email: kduda@arista.com

Dinesh G. Dutt
Independent

Email: didutt@gmail.com

Jon Hudson
Independent

Email: jon.hudson@gmail.com

Ariel Hendel
Facebook, Inc.
1 Hacker Way
Menlo Park, CA 94025
USA

Email: ahendel@fb.com

9. Acknowledgements

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

The authors would like to thank Magnus Nystrom for his reviews and feedback.

Thanks to Daniel Migault, Anoop Ghanwani, Greg Mirksy, Puneet Agarwal, and Tal Mizrahi for their reviews, comments and feedback.

The authors would like to thank David Black for his detailed reviews and valuable inputs.

Thanks to Sami Boutros for his inputs and helpful feedback.

The authors would like to thank Matthew Bocci, Sam Aldrin, Benson Schliesser, Martin Vigoureux, and Alia Atlas for their guidance throughout the process.

10. References

10.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.

- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

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.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-09 (work in progress), July 2018.
- [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.
- [I-D.ietf-nvo3-encap]
Boutros, S., "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-02 (work in progress), September 2018.
- [IEEE.802.1Q_2014]
IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks", IEEE 802.1Q-2014, DOI 10.1109/ieeestd.2014.6991462, December 2014, <<http://ieeexplore.ieee.org/servlet/opac?punumber=6991460>>.

- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://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, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <<https://www.rfc-editor.org/info/rfc3985>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <<https://www.rfc-editor.org/info/rfc5374>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.

- [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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086, March 2017, <<https://www.rfc-editor.org/info/rfc8086>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.
- [RFC8293] Ghanwani, A., Dunbar, L., McBride, M., Bannai, V., and R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC 8293, DOI 10.17487/RFC8293, January 2018, <<https://www.rfc-editor.org/info/rfc8293>>.
- [VL2] "VL2: A Scalable and Flexible Data Center Network", ACM SIGCOMM Computer Communication Review, DOI 10.1145/1594977.1592576, 2009, <<http://www.sigcomm.org/sites/default/files/ccr/papers/2009/October/1594977-1592576.pdf>>.

Authors' Addresses

Jesse Gross (editor)

Email: jesse@kernel.org

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

Email: ilango.s.ganga@intel.com

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

Email: tsridhar@vmware.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2020

J. Gross, Ed.
I. Ganga, Ed.
Intel
T. Sridhar, Ed.
VMware
March 07, 2020

Geneve: Generic Network Virtualization Encapsulation
draft-ietf-nvo3-geneve-16

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 document describes Geneve, an encapsulation protocol designed to recognize and accommodate these changing capabilities and needs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 8, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
1.2. Terminology	4
2. Design Requirements	6
2.1. Control Plane Independence	7
2.2. Data Plane Extensibility	7
2.2.1. Efficient Implementation	8
2.3. Use of Standard IP Fabrics	8
3. Geneve Encapsulation Details	9
3.1. Geneve Packet Format Over IPv4	9
3.2. Geneve Packet Format Over IPv6	11
3.3. UDP Header	13
3.4. Tunnel Header Fields	14
3.5. Tunnel Options	15
3.5.1. Options Processing	17
4. Implementation and Deployment Considerations	18
4.1. Applicability Statement	18
4.2. Congestion Control Functionality	19
4.3. UDP Checksum	19
4.3.1. UDP Zero Checksum Handling with IPv6	19
4.4. Encapsulation of Geneve in IP	21
4.4.1. IP Fragmentation	21
4.4.2. DSCP, ECN and TTL	22
4.4.3. Broadcast and Multicast	23
4.4.4. Unidirectional Tunnels	23
4.5. Constraints on Protocol Features	24
4.5.1. Constraints on Options	24
4.6. NIC Offloads	25
4.7. Inner VLAN Handling	25
5. Transition Considerations	26
6. Security Considerations	26
6.1. Data Confidentiality	27
6.1.1. Inter-Data Center Traffic	27
6.2. Data Integrity	28
6.3. Authentication of NVE peers	29
6.4. Options Interpretation by Transit Devices	29

6.5. Multicast/Broadcast	29
6.6. Control Plane Communications	29
7. IANA Considerations	30
8. Contributors	31
9. Acknowledgements	32
10. References	33
10.1. Normative References	33
10.2. Informative References	34
Authors' Addresses	37

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, for example, ranging all the way from VLANs [IEEE.802.1Q_2018] and MPLS [RFC3031] through the more recent VXLAN [RFC7348] (Virtual eXtensible Local Area Network) and NVGRE [RFC7637] (Network Virtualization Using Generic Routing Encapsulation), 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. Note that the list of protocols presented above is non-exhaustive.

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.

Work such as [VL2] (A Scalable and Flexible Data Center Network) and the NVO3 Data Plane Requirements [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 metadata (e.g. system state) along with the packet data; example use cases of metadata are noted below. The use of some metadata is certainly not a foreign concept - nearly all protocols used for network 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 port identifiers for simple security policies to sending service based context for advanced middlebox applications that terminate and re-encapsulate Geneve traffic.

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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 (Network Interface Controller) 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 of packets within a flow.

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

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

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).

Middlebox. The term middlebox in the context of this document refers to network service functions or appliances for service interposition that would typically implement NVE functionality, which terminate or re-encapsulate Geneve traffic.

NIC. Network Interface Controller. Also called as Network Interface Card or Network Adapter. 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.

Transit device. A forwarding element (e.g. router or switch) along the path of the tunnel making up part of the Underlay Network. A transit device may be capable of understanding the Geneve packet format but does not originate or terminate Geneve packets.

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, tunnel 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. Tunnel endpoints are frequently a component of an NVE (Network Virtualization Edge) 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 for data center environments, 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. Many of the concepts of network virtualization overlays over Layer 3 IP networks are described in the NVO3 Framework [RFC7365]. 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.

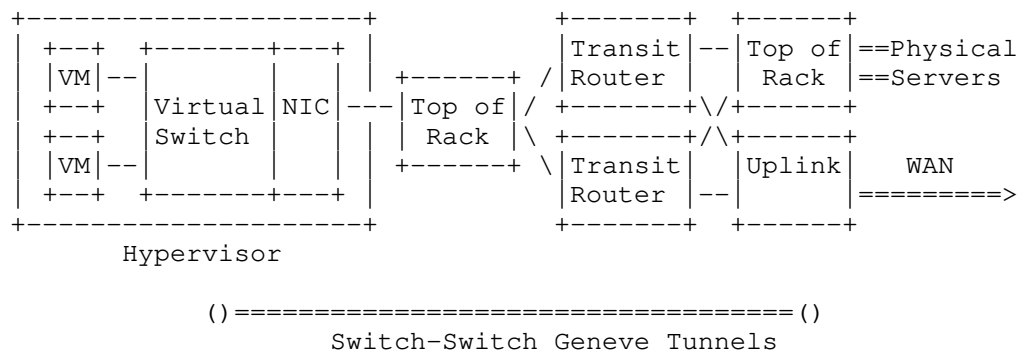


Figure 1: Sample Geneve Deployment

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, VXLAN [RFC7348] prescribed a multicast learning-based control plane), these specifications have largely been treated as describing only the data format. The VXLAN packet 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 is 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 reduces the chance of obsolescence 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 packets, they are equally important on data packets as well to segregate and direct forwarding (for instance, the examples given before of input port based security policies and terminating/re-encapsulating service interposition both require tags to be placed on data packets). Therefore, while it would be desirable to limit the extensibility to only control packets 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 a desired 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.

Tunnel 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 tunnel endpoints are the ultimate consumers of the data, transmitters can tailor their output to the capabilities of the recipient.

Transit devices may be able to interpret the options, however, as non-terminating devices, transit devices do not originate or terminate the Geneve packet, hence MUST NOT modify Geneve headers and MUST NOT insert or delete options, which is the responsibility of tunnel endpoints. Options, if present in the packet, MUST only be generated and terminated by tunnel endpoints. The participation of transit devices in interpreting options 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 tunnel 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 packet 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 packet format transported (for example) over Ethernet along with an Ethernet payload.

3.1. Geneve Packet 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 Preamble, Start Frame
| Delimiter(SFD) & Frame Check Sequence(FCS) are not included
| and the Ethernet Payload need not be 4-byte aligned)
|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

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

```

3.2. Geneve Packet 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 Preamble, Start Frame |
| Delimiter(SFD) & Frame Check Sequence(FCS) are not included   |
| and the Ethernet Payload need not be 4-byte aligned)          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Frame Check Sequence:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|          New Frame Check Sequence (FCS) 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. In addition to setting the source port, for IPv6, flow label MAY also be used for providing entropy. For an example of using IPv6 flow label for tunnel use cases, see [RFC6438].

If Geneve traffic is shared with other UDP listeners on the same IP address, tunnel endpoints SHOULD implement a mechanism to ensure ICMP return traffic arising from network errors is directed to the correct listener. The definition of such a mechanism is beyond the scope of this document.

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. It is the responsibility of the control plane for any reconfiguration of the assigned port and its interpretation by respective devices. The definition of the control plane is beyond the scope of this document.

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

UDP checksum: In order to protect the Geneve header, options and payload from potential data corruption, UDP checksum SHOULD be generated as specified in [RFC0768] and [RFC1112] when Geneve is encapsulated in IPv4. To protect the IP header, Geneve header, options and payload from potential data corruption, the UDP checksum MUST be generated by default as specified in [RFC0768] and [RFC8200] when Geneve is encapsulated in IPv6, except for certain conditions, which are outlined in the next paragraph. Upon receiving such packets with non-zero UDP checksum, the

receiving tunnel endpoints MUST validate the checksum. If the checksum is not correct, the packet MUST be dropped, otherwise the packet MUST be accepted for decapsulation.

Under certain conditions, the UDP checksum MAY be set to zero on transmit for packets encapsulated in both IPv4 and IPv6 [RFC8200]. See Section 4.3 for additional requirements that apply when using zero UDP checksum with IPv4 and IPv6. Disabling the use of UDP checksums is an operational consideration that should take into account the risks and effects of packet corruption.

3.4. Tunnel Header Fields

Ver (2 bits): The current version number is 0. Packets received by a tunnel endpoint with an unknown version MUST be dropped. Transit devices interpreting 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.

Transit devices MUST maintain consistent forwarding behavior irrespective of the value of 'Opt Len', including ECMP link selection.

O (1 bit): Control packet. This packet contains a control message. Control messages are sent between tunnel endpoints. Tunnel endpoints MUST NOT forward the payload and transit devices MUST NOT attempt to interpret it. Since control messages are less frequent, it is RECOMMENDED that tunnel 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 tunnel endpoints where option parsing is not supported the packet 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

packet. Transit devices MUST NOT drop packets on the basis of this bit.

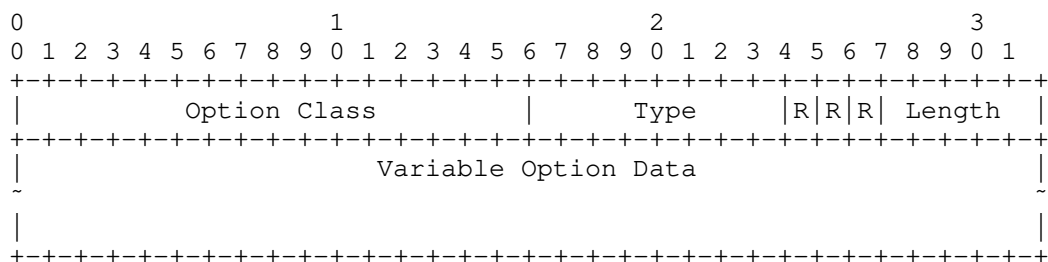
Rsvd. (6 bits): Reserved field, which MUST be zero on transmission and MUST be ignored on receipt.

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.

3.5. Tunnel Options



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 allocation by IETF Review and for Experimental Use (see Section 7).

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 separate documents.

The high order bit of the option type indicates that this is a critical option. If the receiving tunnel endpoint does not recognize this option and this bit is set then the packet **MUST** be dropped. If this 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 option with the 'C' bit set 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. These bits **MUST** be zero on transmission and **MUST** be 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. A value of 0 in the Length field implies an option with only an option header and no variable option data. 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 a tunnel endpoint that processes the options.

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

3.5.1. Options Processing

Geneve options are intended to be originated and processed by tunnel endpoints. However, options MAY be interpreted by transit devices along the tunnel path. Transit devices not interpreting Geneve headers (which may or may not include options) MUST handle Geneve packets as any other UDP packet and maintain consistent forwarding behavior.

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

- o Receiving tunnel endpoints MUST drop packets containing unknown options with the 'C' bit set in the option type. Conversely, transit devices MUST NOT drop packets as a result of encountering unknown options, including those with the 'C' bit set.
- o The contents of the options and their ordering MUST NOT be modified by transit devices.
- o If a tunnel endpoint receives a Geneve packet with 'Opt Len' (total length of all options) that exceeds the options processing capability of the tunnel endpoint then the tunnel endpoint MUST drop such packets. An implementation may raise an exception to the control plane of such an event. It is the responsibility of the control plane to ensure the communicating peer tunnel endpoints have the processing capability to handle the total length of options. The definition of the control plane is beyond the scope of this document.

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.

Architecturally, options are intended to be self-descriptive and independent. This enables parallelism in option processing and reduces implementation complexity. However, the control plane may impose certain ordering restrictions as described in Section 4.5.1.

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. Actual definition of the option type is beyond the scope of this document. The option type and its interpretation should be defined by the entity that owns the option class.

Options may be processed by NIC hardware utilizing offloads (e.g. LSO and LRO) as described in Section 4.6. Careful consideration should be given to how the offload capabilities outlined in Section 4.6 impact an option's design.

4. Implementation and Deployment Considerations

4.1. Applicability Statement

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It is intended for use in public or private data center environments, for deploying multi-tenant overlay networks over an existing IP underlay network.

Geneve is a UDP based encapsulation protocol transported over existing IPv4 and IPv6 networks. Hence, as a UDP based protocol, Geneve adheres to the UDP usage guidelines as specified in [RFC8085]. The applicability of these guidelines are dependent on the underlay IP network and the nature of Geneve payload protocol (example TCP/IP, IP/Ethernet).

Geneve is intended to be deployed in a data center network environment operated by a single operator or adjacent set of cooperating network operators that fits with the definition of controlled environments in [RFC8085]. A network in a controlled environment can be managed to operate under certain conditions whereas in the general Internet this cannot be done. Hence requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements of the general Internet.

For the purpose of this document, a traffic-managed controlled environment (TMCE) is defined as an IP network that is traffic-engineered and/or otherwise managed (e.g., via use of traffic rate

limiters) to avoid congestion. The concept of TMCE is outlined in [RFC8086]. Significant portions of the text in Section 4.1 through Section 4.3 are based on [RFC8086] as applicable to Geneve.

It is the responsibility of the operator to ensure that the guidelines/requirements in this section are followed as applicable to their Geneve deployment(s).

4.2. Congestion Control Functionality

Geneve does not natively provide congestion control functionality and relies on the payload protocol traffic for congestion control. As such Geneve **MUST** be used with congestion controlled traffic or within a network that is traffic managed to avoid congestion (TMCE). An operator of a traffic managed network (TMCE) may avoid congestion by careful provisioning of their networks, rate-limiting of user data traffic and traffic engineering according to path capacity.

4.3. UDP Checksum

In order to provide integrity of Geneve headers, options and payload, (for example to avoid misdelivery of payload to different tenant systems) in case of data corruption, the outer UDP checksum **SHOULD** be used with Geneve when transported over IPv4. The UDP checksum provides a statistical guarantee that a payload was not corrupted in transit. These integrity checks are not strong from a coding or cryptographic perspective and are not designed to detect physical-layer errors or malicious modification of the datagram (see Section 3.4 of [RFC8085]). In deployments where such a risk exists, an operator **SHOULD** use additional data integrity mechanisms such as offered by IPsec (see Section 6.2).

An operator **MAY** choose to disable UDP checksums and use zero checksums if Geneve packet integrity is provided by other data integrity mechanisms such as IPsec or additional checksums or if one of the conditions in Section 4.3.1 a, b, c are met.

By default, UDP checksums **MUST** be used when Geneve is transported over IPv6. A tunnel endpoint **MAY** be configured for use with zero UDP checksum if additional requirements in Section 4.3.1 are met.

4.3.1. UDP Zero Checksum Handling with IPv6

When Geneve is used over IPv6, the UDP checksum is used to protect IPv6 headers, UDP headers and Geneve headers, options and payload from potential data corruption. As such by default Geneve **MUST** use UDP checksums when transported over IPv6. An operator **MAY** choose to configure to operate with zero UDP checksum if operating in a traffic

managed controlled environment as stated in Section 4.1 if one of the following conditions are met.

- a. It is known that the packet corruption is exceptionally unlikely (perhaps based on knowledge of equipment types in their underlay network) and the operator is willing to take a risk of undetected packet corruption
- b. It is judged through observational measurements (perhaps through historic or current traffic flows that use non zero checksum) that the level of packet corruption is tolerably low and where the operator is willing to take the risk of undetected corruption.
- c. Geneve payload is carrying applications that are tolerant of misdelivered or corrupted packets (perhaps through higher layer checksum validation and/or reliability through retransmission)

In addition Geneve tunnel implementations using zero UDP checksum MUST meet the following requirements:

1. Use of UDP checksum over IPv6 MUST be the default configuration for all Geneve tunnels.
2. If Geneve is used with zero UDP checksum over IPv6 then such tunnel endpoint implementation MUST meet all the requirements specified in Section 4 of [RFC6936] and requirement 1 as specified in Section 5 of [RFC6936] as that is relevant to Geneve.
3. The Geneve tunnel endpoint that decapsulates the tunnel SHOULD check the source and destination IPv6 addresses are valid for the Geneve tunnel that is configured to receive zero UDP checksum and discard other packets for which such check fails.
4. The Geneve tunnel endpoint that encapsulates the tunnel MAY use different IPv6 source addresses for each Geneve tunnel that uses zero UDP checksum mode in order to strengthen the decapsulator's check of the IPv6 source address (i.e the same IPv6 source address is not to be used with more than one IPv6 destination address, irrespective of whether that destination address is a unicast or multicast address). When this is not possible, it is RECOMMENDED to use each source address for as few Geneve tunnels that use zero UDP checksum as is feasible.

Note that (for requirements 3 and 4) the receiving tunnel endpoint can apply these checks only if it has out-of-band knowledge that the encapsulating tunnel endpoint is applying the

indicated behavior. One possibility to obtain this out-of-band knowledge is through signaling by the control plane. The definition of the control plane is beyond the scope of this document.

5. Measures SHOULD be taken to prevent Geneve traffic over IPv6 with zero UDP checksum from escaping into the general Internet. Examples of such measures include employing packet filters at the gateways or edge of Geneve network and/or keeping logical or physical separation of the Geneve network from networks carrying the general Internet traffic.

The above requirements do not change either the requirements specified in [RFC8200] or the requirements specified in [RFC6936].

The use of the source IPv6 address in addition to the destination IPv6 address, plus the recommendation against reuse of source IPv6 addresses among Geneve tunnels collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

Editorial Note (The following paragraph to be removed by the RFC Editor before publication)

It was discussed during TSVART early review if the level of requirement for using different IPv6 source addresses for different tunnel destinations would need to be "MAY" or "SHOULD". The discussion concluded that it was appropriate to keep this as "MAY", since it was considered not realistic for control planes having to maintain a high level of state on a per tunnel destination basis. In addition, the text above provides sufficient guidance to operators and implementors on possible mitigations.

4.4. 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.

4.4.1. IP Fragmentation

It is strongly RECOMMENDED that Path MTU Discovery ([RFC1191], [RFC8201]) be used to prevent or minimize fragmentation. The use of Path MTU Discovery on the transit network provides the encapsulating

tunnel endpoint with soft-state about the link that it may use to prevent or minimize fragmentation depending on its role in the virtualized network. The NVE can maintain this state (the MTU size of the tunnel link(s) associated with the tunnel endpoint), so if a tenant system sends large packets that when encapsulated exceed the MTU size of the tunnel link, the tunnel endpoint can discard such packets and send exception messages to the tenant system(s). If the tunnel endpoint is associated with a routing or forwarding function and/or has the capability to send ICMP messages, the encapsulating tunnel endpoint MAY send ICMP fragmentation needed [RFC0792] or Packet Too Big [RFC4443] messages to the tenant system(s). When determining the MTU size of a tunnel link, maximum length of options MUST be assumed as options may vary on a per-packet basis. For example, recommendations/guidance for handling fragmentation in similar overlay encapsulation services like PWE3 are provided in Section 5.3 of [RFC3985].

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. For example, some of the issues associated with MTU size and fragmentation in IP tunneling and use of ICMP messages is outlined in Section 4.2 of [I-D.ietf-intarea-tunnels].

4.4.2. DSCP, ECN and TTL

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

[RFC2983] provides guidance 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 MUST be followed for IP packets encapsulated in Geneve.

Though Uniform or Pipe models could be used for TTL (or Hop Limit in case of IPv6) handling when tunneling IP packets, the Pipe model is more aligned with network virtualization. [RFC2003] provides guidance on handling TTL between inner IP header and outer IP tunnels; this model is more aligned with the Pipe model and is RECOMMENDED for use with Geneve for network virtualization applications.

4.4.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two tunnel 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 is beyond the scope of this document.

When physical multicast is in use, devices with heterogeneous capabilities may be present in the same group. Some options may only be interpretable by a subset of the devices in the group. Other devices can safely ignore such options unless the 'C' bit is set to mark the unknown option as critical. Requirements outlined in Section 3.4 apply for critical options.

In addition, [RFC8293] provides examples of various mechanisms that can be used for multicast handling in network virtualization overlay networks.

4.4.4. Unidirectional Tunnels

Generally speaking, a Geneve tunnel is a unidirectional concept. IP is not a connection oriented protocol and it is possible for two tunnel 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.

4.5. 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, definition of such mechanisms are beyond the scope of this document.

4.5.1. Constraints on Options

While Geneve options are flexible, a control plane may restrict the number of option TLVs as well as the order and size of the TLVs between tunnel endpoints to make it simpler for a data plane implementation in software or hardware to handle [I-D.ietf-nvo3-encap]. For example, there may be some critical information such as a secure hash that must be processed in a certain order to provide lowest latency or there may be other scenarios where the options must be processed in a certain order due to protocol semantics.

A control plane may negotiate a subset of option TLVs and certain TLV ordering, as well may limit the total number of option TLVs present in the packet, for example, to accommodate hardware capable of processing fewer options [I-D.ietf-nvo3-encap]. Hence, a control plane needs to have the ability to describe the supported TLVs subset and their order to the tunnel endpoints. In the absence of a control plane, alternative configuration mechanisms may be used for this purpose. Such mechanisms are beyond the scope of this document.

4.6. 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 unless an option allows an exception. 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 Options **MUST NOT** be reordered during the course of offload processing, including when merging packets for the purpose of LRO.
- o NICs performing offloads **MUST NOT** drop packets with unknown options, including those marked as critical, unless explicitly configured.

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.

4.7. 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 tunnel

endpoint and/or control plane and not explicitly defined as part of the data format.

5. Transition Considerations

Viewed exclusively from the data plane, Geneve is compatible with existing IP networks as it appears to most devices as UDP packets. 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 builds on the base data plane functionality provided by the most common protocols used for network virtualization (VXLAN, NVGRE) 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 packet formats supporting the same set of capabilities, there is no need for a hard transition - tunnel endpoints directly communicating with each other can use any common protocol, which may be different even within a single overall system. As transit devices are primarily forwarding packets 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 encapsulated within a UDP/IP packet, Geneve does not have any inherent security mechanisms. As a result, an attacker with access to the underlay network transporting the IP packets has the ability to snoop, alter or inject packets. Compromised tunnel endpoints or transit devices 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 service 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.

When crossing an untrusted link, such as the general Internet, VPN technologies such as IPsec [RFC4301] should be used to provide authentication and/or encryption of the IP packets formed as part of Geneve encapsulation (See Section 6.1.1).

Geneve does not otherwise affect the security of the encapsulated packets. As per the guidelines of BCP 72 [RFC3552], the following sections describe potential security risks that may be applicable to Geneve deployments and approaches to mitigate such risks. It is also noted that not all such risks are applicable to all Geneve deployment scenarios, i.e., only a subset may be applicable to certain deployments. So an operator has to make an assessment based on their network environment and determine the risks that are applicable to their specific environment and use appropriate mitigation approaches as applicable.

6.1. Data Confidentiality

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It can be used to deploy multi-tenant overlay networks over an existing IP underlay network in a public or private data center. The overlay service is typically provided by a service provider, for example a cloud services provider or a private data center operator, this may or not may be the same provider as an underlay service provider. Due to the nature of multi-tenancy in such environments, a tenant system may expect data confidentiality to ensure its packet data is not tampered with (active attack) in transit or a target of unauthorized monitoring (passive attack) for example by other tenant systems or underlay service provider. A compromised network node or a transit device within a data center may passively monitor Geneve packet data between NVEs; or route traffic for further inspection. A tenant may expect the overlay service provider to provide data confidentiality as part of the service or a tenant may bring its own data confidentiality mechanisms like IPsec or TLS to protect the data end to end between its tenant systems. The overlay provider is expected to provide cryptographic protection in cases where the underlay provider is not the same as the overlay provider to ensure the payload is not exposed to the underlay.

If an operator determines data confidentiality is necessary in their environment based on their risk analysis, for example as in multi-tenant environments, then an encryption mechanism SHOULD be used to encrypt the tenant data end to end between the NVEs. The NVEs may use existing well established encryption mechanisms such as IPsec, DTLS, etc.

6.1.1. Inter-Data Center Traffic

A tenant system in a customer premises (private data center) may want to connect to tenant systems on their tenant overlay network in a public cloud data center or a tenant may want to have its tenant systems located in multiple geographically separated data centers for

high availability. Geneve data traffic between tenant systems across such separated networks should be protected from threats when traversing public networks. Any Geneve overlay data leaving the data center network beyond the operator's security domain SHOULD be secured by encryption mechanisms such as IPsec or other VPN technologies to protect the communications between the NVEs when they are geographically separated over untrusted network links. Specification of data protection mechanisms employed between data centers is beyond the scope of this document.

The principles described in Section 4 regarding controlled environments still apply to the geographically separated data center usage outlined in this section.

6.2. Data Integrity

Geneve encapsulation is used between NVEs to establish overlay tunnels over an existing IP underlay network. In a multi-tenant data center, a rogue or compromised tenant system may try to launch a passive attack such as monitoring the traffic of other tenants, or an active attack such as trying to inject unauthorized Geneve encapsulated traffic such as spoofing, replay, etc., into the network. To prevent such attacks, an NVE MUST NOT propagate Geneve packets beyond the NVE to tenant systems and SHOULD employ packet filtering mechanisms so as not to forward unauthorized traffic between tenant systems in different tenant networks. An NVE MUST NOT interpret Geneve packets from tenant systems other than as frames to be encapsulated.

A compromised network node or a transit device within a data center may launch an active attack trying to tamper with the Geneve packet data between NVEs. Malicious tampering of Geneve header fields may cause the packet from one tenant to be forwarded to a different tenant network. If an operator determines the possibility of such threat in their environment, the operator may choose to employ data integrity mechanisms between NVEs. In order to prevent such risks, a data integrity mechanism SHOULD be used in such environments to protect the integrity of Geneve packets including packet headers, options and payload on communications between NVE pairs. A cryptographic data protection mechanism such as IPsec may be used to provide data integrity protection. A data center operator may choose to deploy any other data integrity mechanisms as applicable and supported in their underlay networks, although non-cryptographic mechanisms may not protect the Geneve portion of the packet from tampering.

6.3. Authentication of NVE peers

A rogue network device or a compromised NVE in a data center environment might be able to spoof Geneve packets as if it came from a legitimate NVE. In order to mitigate such a risk, an operator SHOULD use an authentication mechanism, such as IPsec to ensure that the Geneve packet originated from the intended NVE peer, in environments where the operator determines spoofing or rogue devices is a potential threat. Other simpler source checks such as ingress filtering for VLAN/MAC/IP address, reverse path forwarding checks, etc., may be used in certain trusted environments to ensure Geneve packets originated from the intended NVE peer.

6.4. Options Interpretation by Transit Devices

Options, if present in the packet, are generated and terminated by tunnel endpoints. As indicated in Section 2.2.1, transit devices may interpret the options. However, if the packet is protected by tunnel endpoint to tunnel endpoint encryption, for example through IPsec, transit devices will not have visibility into the Geneve header or options in the packet. In such cases transit devices MUST handle Geneve packets as any other IP packet and maintain consistent forwarding behavior. In cases where options are interpreted by transit devices, the operator MUST ensure that transit devices are trusted and not compromised. The definition of a mechanism to ensure this trust is beyond the scope of this document.

6.5. Multicast/Broadcast

In typical data center networks where IP multicasting is not supported in the underlay network, multicasting may be supported using multiple unicast tunnels. The same security requirements as described in the above sections can be used to protect Geneve communications between NVE peers. If IP multicasting is supported in the underlay network and the operator chooses to use it for multicast traffic among tunnel endpoints, then the operator in such environments may use data protection mechanisms such as IPsec with multicast extensions [RFC5374] to protect multicast traffic among Geneve NVE groups.

6.6. Control Plane Communications

A Network Virtualization Authority (NVA) as outlined in [RFC8014] may be used as a control plane for configuring and managing the Geneve NVEs. The data center operator is expected to use security mechanisms to protect the communications between the NVA to NVEs and use authentication mechanisms to detect any rogue or compromised NVEs within their administrative domain. Data protection mechanisms for

control plane communication or authentication mechanisms between the NVA and the NVEs are beyond the scope of this document.

7. IANA Considerations

IANA has allocated UDP port 6081 in the Service Name and Transport Protocol Port Number Registry [IANA-SN] as the well-known destination port for Geneve based on early registration.

Upon publication of this document, this registration will have its reference changed to cite this document [RFC-to-be] and inline with [RFC6335] the assignee and contact of the port entry should be changed to IESG <iesg@ietf.org> and IETF Chair <chair@ietf.org> respectively:

Service Name: geneve
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: Generic Network Virtualization Encapsulation (Geneve)
Reference: [RFC-to-be]
Port Number: 6081

In addition, IANA is requested to create a new "Geneve Option Class" registry to allocate Option Classes. This registry is to be placed under a new Network Virtualization Overlay (NVO3) protocols page (to be created) in IANA protocol registries [IANA-PR]. The Geneve Option Class registry shall consist of 16-bit hexadecimal values along with descriptive strings, assignee/contact information and references. The registration rules for the new registry are (as defined by [RFC8126]):

Range	Registration Procedures
0x0000..0x00FF	IETF Review
0x0100..0xFEFF	First Come First Served
0xFF00..0xFFFF	Experimental Use

Initial registrations in the new registry are as follows:

Option Class	Description	Assignee/Contact	References
0x0100	Linux		
0x0101	Open vSwitch (OVS)		
0x0102	Open Virtual Networking (OVN)		
0x0103	In-band Network Telemetry (INT)		
0x0104	VMware, Inc.		
0x0105	Amazon.com, Inc.		
0x0106	Cisco Systems, Inc.		
0x0107	Oracle Corporation		
0x0108..0x0110	Amazon.com, Inc.		

8. Contributors

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

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

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

Email: kduda@arista.com

Dinesh G. Dutt
Independent

Email: didutt@gmail.com

Jon Hudson
Independent

Email: jon.hudson@gmail.com

Ariel Hendel
Facebook, Inc.
1 Hacker Way
Menlo Park, CA 94025
USA

Email: ahendel@fb.com

9. Acknowledgements

The authors wish to acknowledge Puneet Agarwal, David Black, Sami Boutros, Scott Bradner, Martin Casado, Alissa Cooper, Roman Danyliw, Bruce Davie, Anoop Ghanwani, Benjamin Kaduk, Suresh Krishnan, Mirja Kuhlewind, Barry Leiba, Daniel Migault, Greg Mirksy, Tal Mizrahi,

Kathleen Moriarty, Magnus Nystrom, Adam Roach, Sabrina Tanamal, Dave Thaler, Eric Vyncke, Magnus Westerlund and many other members of the NVO3 WG for their reviews, comments and suggestions.

The authors would like to thank Sam Aldrin, Alia Atlas, Matthew Bocci, Benson Schliesser, and Martin Vigoureux for their guidance throughout the process.

10. References

10.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.

- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [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, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

10.2. Informative References

- [ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", <<https://www.iana.org/assignments/ieee-802-numbers>>.
- [I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.
- [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.

- [I-D.ietf-nvo3-encap] Boutros, S., "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-05 (work in progress), February 2020.
- [IANA-PR] IANA, "Protocol Registries", <<https://www.iana.org/protocols>>.
- [IANA-SN] IANA, "Service Name and Transport Protocol Port Number Registry", <<https://www.iana.org/assignments/service-names-port-numbers>>.
- [IEEE.802.1Q_2018] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", IEEE 802.1Q-2018, DOI 10.1109/ieeestd.2018.8403927, July 2018, <<http://ieeexplore.ieee.org/servlet/opac?punumber=8403925>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://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, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <<https://www.rfc-editor.org/info/rfc3985>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <<https://www.rfc-editor.org/info/rfc5374>>.

- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [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, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086, March 2017, <<https://www.rfc-editor.org/info/rfc8086>>.
- [RFC8293] Ghanwani, A., Dunbar, L., McBride, M., Bannai, V., and R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC 8293, DOI 10.17487/RFC8293, January 2018, <<https://www.rfc-editor.org/info/rfc8293>>.
- [VL2] "VL2: A Scalable and Flexible Data Center Network", ACM SIGCOMM Computer Communication Review, DOI 10.1145/1594977.1592576, 2009, <<https://www.sigcomm.org/sites/default/files/ccr/papers/2009/October/1594977-1592576.pdf>>.

Authors' Addresses

Jesse Gross (editor)

Email: jesse@kernel.org

Ilango Ganga (editor)

Intel Corporation

2200 Mission College Blvd.

Santa Clara, CA 95054

USA

Email: ilango.s.ganga@intel.com

T. Sridhar (editor)

VMware, Inc.

3401 Hillview Ave.

Palo Alto, CA 94304

USA

Email: tsridhar@vmware.com

NVO3
Internet-Draft
Intended status: Informational
Expires: September 1, 2019

D. Migault
Ericsson
S. Boutros
D. Wings
VMware, Inc.
S. Krishnan
Kaloom
February 28, 2019

Geneve Security Requirements
draft-mglt-nvo3-geneve-security-requirements-06

Abstract

The document defines the security requirements to protect tenants overlay traffic against security threats from the NVO3 network components that are interconnected with tunnels implemented using Generic Network Virtualization Encapsulation (Geneve).

The document provides two sets of security requirements: 1. requirements to evaluate the data plane security of a given deployment of Geneve overlay. Such requirements are intended to evaluate a given deployment. 2. requirement a security mechanism need to fulfill to secure any deployment of Geneve overlay deployment

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 1, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Notation	3
2. Introduction	3
3. Terminology	6
4. Security Threats	6
4.1. Passive Attacks	6
4.2. Active Attacks	7
5. Requirements for Security Mitigations	8
5.1. Protection Against Traffic Sniffing	8
5.1.1. Operational Security Requirements	9
5.1.2. Geneve Security Requirements	10
5.2. Protecting Against Traffic Injection	10
5.2.1. Operational Security Requirements	12
5.2.2. Geneve Security Requirements	13
5.3. Protecting Against Traffic Redirection	13
5.4. Protecting Against Traffic Replay	14
5.4.1. Geneve Security Requirements	14
5.4.2. Geneve Security Requirements	15
5.5. Security Management	15
5.5.1. Operational Security Requirements	15
5.5.2. Geneve Security Requirements	15
6. IANA Considerations	16
7. Security Considerations	16
8. Appendix	16
8.1. DTLS	16
8.1.1. Operational Security Requirements	16
8.1.2. Geneve Security Requirements	18
8.2. IPsec	20
8.2.1. Operational Security Requirements	20
8.2.2. Geneve Security Requirements	21
9. Acknowledgments	23
10. References	23

10.1. Normative References	23
10.2. Informative References	25
Authors' Addresses	25

1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Introduction

The network virtualization overlay over Layer 3 (NVO3) as depicted in Figure 1, allows an overlay cloud provider to provide a logical L2/L3 interconnect for the Tenant Systems TSes that belong to a specific tenant network. A packet received from a TS is encapsulated by the ingress Network Virtualization Edge (NVE). The encapsulated packet is then sent to the remote NVE through a tunnel. When reaching the egress NVE of the tunnel, the packet is decapsulated and forwarded to the target TS. The L2/L3 address mappings to the remote NVE(s) are distributed to the NVEs by a logically centralized Network Virtualization Authority (NVA) or using a distributed control plane such as Ethernet-VPN. In a datacenter, the NVO3 tunnels can be implemented using Generic Network Virtualization Encapsulation (Geneve) [I-D.ietf-nvo3-geneve]. Such Geneve tunnels establish NVE-to-NVE communications, may transit within the data center via Transit device. The Geneve tunnels overlay network enable multiple Virtual Networks to coexist over a shared underlay infrastructure, and a Virtual Network may span a single data center or multiple data centers.

The underlay infrastructure on which the multi-tenancy overlay networks are hosted, can be owned and provided by an underlay provider who may be different from the overlay cloud provider.

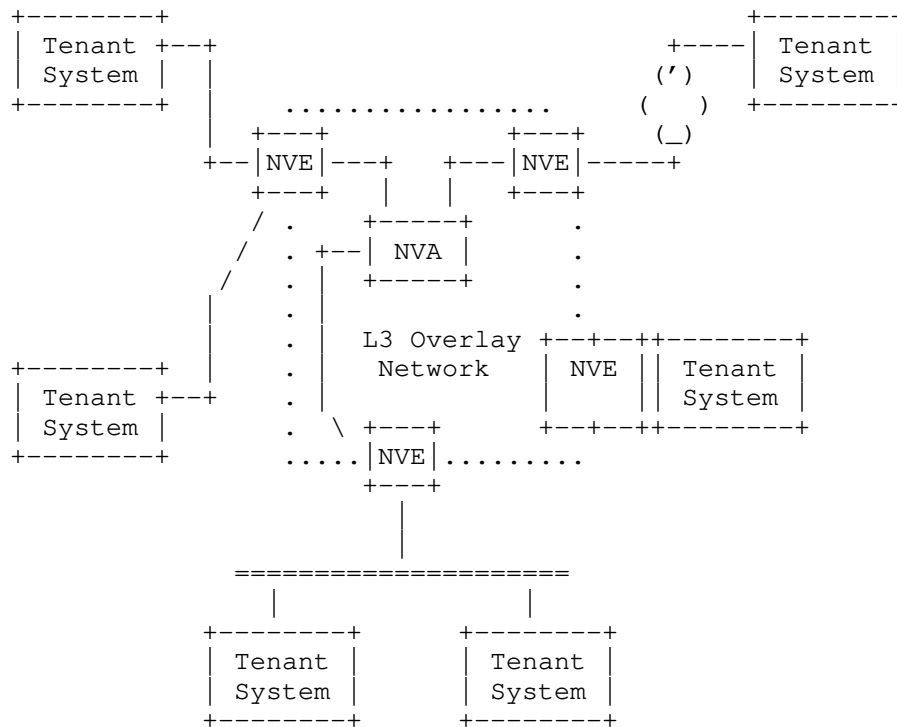


Figure 1: Generic Reference Model for Network Virtualization Overlays [RFC7365]

This document discusses the security risks that a Geneve based NVO3 network may encounter. In addition, this document lists the requirements to protect the Geneve packet components defined in [I-D.ietf-nvo3-geneve] that include the Geneve tunnel IP and UDP header, the Geneve Header, Geneve options, and inner payload.

The document provides two sets of security requirements:

1. SEC-OP: requirements to evaluate a given deployment of Geneve overlay. Such requirements are intended to Geneve overlay provider to evaluate a given deployment. Security of the Geneve packet may be achieved using various mechanisms. Typically, some deployments may use a limited subset of the capabilities provided by Geneve and rely on specific assumptions. Given these specificities, the secure deployment of a given Geneve deployment may be achieved reusing specific mechanisms such as for example DTLS [RFC6347] or IPsec [RFC4301]. On the other hand, the definition of a security mechanisms that enables to secure any Geneve deployment requires the design of a Geneve specific

mechanism. Note that the security is limited to the security of the data plane only. Additional requirements for the control plane MAY be considered in [I-D.ietf-nvo3-security-requirements]. A given Geneve deployment will be considered secured when matching with all SEC-OP requirements does not raise any concern. As such the given deployment will be considered passing SEC-OP requirements that are not applicable.

2. SEC-GEN: requirements a security mechanism need to fulfill to secure any deployment of Geneve overlay deployment. Such mechanism may require the design of a specific solution. In the case new protocol needs to be design, the document strongly recommend to re-use existing security protocols like IP Security (IPsec) [RFC4301] and Datagram Transport Layer Security (DTLS) [RFC6347], and existing encryption algorithms (such as [RFC8221]), and authentication protocols. A given candidate for a security mechanism will be considered as valid when matching with all SEC-GEN requirements does not raise any concern. In other words, at least all MUST status are met.

This document assumes the following roles are involved: - Tenant: designates the entity that connects various systems within a single virtualized network. The various system can typically be containers, VMs implementing a single or various functions.

- Geneve Overlay Provider: provides the Geneve overlay that seamlessly connect the various Tenant Systems over a given virtualized network.

- Infrastructure Provider: provides the infrastructure that runs the Geneve overlay network as well as the Tenant System. A given deployment may consider different infrastructure provider with different level of trust. Typically the Geneve overlay network may use a public cloud to extend the resource of a private cloud. Similarly, a edge computing may extend its resources using resource of the core network.

Tenant, Geneve Overlay Provider and Infrastructure Provider can be implemented by a single or various different entities with different level of trust between each other. The simplest deployment may consists in a single entity running its systems in its data center and using Geneve in order to manage its internal resources. A more complex use case may consider that a Tenant subscribe to the Geneve Overlay Provider which manage the virtualized network over various type of infrastructure. The trust between the Tenant, Geneve Overlay Provider and Infrastructure Provider may be limited.

Given the different relations between Tenant, Geneve Overlay Provider and Infrastructure Provider, this document aims providing requirements to ensure: 1. The Geneve Overlay Provider delivers

tenant payload traffic (Geneve inner payload) and ensuring privacy and integrity. 2. The Geneve Overlay Provider provides the necessary means to prevent injection or redirection of the Tenant traffic from a rogue node in the Geneve overlay network or a rogue node from the infrastructure. 3. The Geneve Overlay Provider can rely on the Geneve overlay in term of robustness and reliability of the signaling associated to the Geneve packets (Geneve tunnel header, Geneve header and Geneve options) in order to appropriately manage its overlay.

3. Terminology

This document uses the terminology of [RFC8014], [RFC7365] and [I-D.ietf-nvo3-geneve].

4. Security Threats

This section considers attacks performed by NVE, network devices or any other devices using Geneve, that is when the attackers knowing the details of the Geneve packets can perform their attacks by changing fields in the Geneve tunnel header, base header, Geneve options and Geneve inner payload. Attacks related to the control plane are outside the scope of this document. The reader is encouraged to read [I-D.ietf-nvo3-security-requirements] for a similar threat analysis of NVO3 overlay networks.

Threats include traffic analysis, sniffing, injection, redirection, and replay. Based on these threats, this document enumerates the security requirements.

Threats are divided into two categories: passive attack and active attack.

Threats are always associated with risks and the evaluation of these risks depend among other things on the environment.

4.1. Passive Attacks

Passive attacks include traffic analysis (noticing which workloads are communicating with which other workloads, how much traffic, and when those communications occur) and sniffing (examining traffic for useful information such as personally-identifiable information or protocol information (e.g., TLS certificate, overlay routing protocols)).

Passive attacks may also consist in inferring information about a virtualized network or some Tenant System from observing the Geneve traffic. This could also involve the correlation between observed

traffic and additional information. For example, a passive network observer can determine two virtual machines are communicating by manipulating activity or network activity of other virtual machines on that same host. For example, the attacker could control (or be otherwise aware of) network activity of the other VMs running on the same host, and deduce other network activity is due to a victim VM.

A rogue element of the overlay Geneve network under the control of an attacker may leak and redirect the traffic from a virtual network to the attacker for passive monitoring [RFC7258].

Avoiding leaking information is hard to enforced. The security requirements provided in section {{sniffing}} expect to mitigate such attacks by lowering the consequences, typically making leaked data unusable to an attacker.

4.2. Active Attacks

Active attacks involve modifying Geneve packets, injecting Geneve packets, or interfering with Geneve packet delivery (such as by corrupting packet checksum). Active attack may target the Tenant System or the Geneve overlay.

There are multiple motivations to inject illegitimate traffic into a tenants network. When the rogue element is on the path of the TS traffic, it may be able to inject and receive the corresponding messages back. On the other hand, if the attacker is not on the path of the TS traffic it may be limited to only inject traffic to a TS without receiving any response back. When rogue element have access to the traffic in both directions, the possibilities are only limited by the capabilities of the other on path elements - Transit device, NVE or TS - to detect and protect against the illegitimate traffic. On the other hand, when the rogue element is not on path, the surface for such attacks remains still quite large. For example, an attacker may target a specific TS or application by crafting a specific packet that can either generate load on the system or crash the system or application. TCP syn flood typically overload the TS while not requiring the ability to receive responses. Note that udp application are privileged target as they do not require the establishment of a session and are expected to treat any incoming packets.

Traffic injection may also be used to flood the virtual network to disrupt the communications between the TS or to introduce additional cost for the tenant, for example when pricing considers the traffic inside the virtual network. The two latest attacks may also take advantage of applications with a large factor of amplification for their responses as well as applications that upon receiving a packet

interact with multiple TS. Similarly, applications running on top of UDP are privileged targets.

Note also that an attacker that is not able to receive the response traffic, may use other channels to evaluate or measure the impact of the attack. Typically, in the case of a service, the attacker may have access, for example, to a user interface that provides indication on the level of disruption and the success of an attack, Such feed backs may also be used by the attacker to discover or scan the network.

Preventing traffic to cross virtual networks, reduce the surface of attack, but rogue element main still perform attacks within a given virtual network by replaying a legitimate packet. Some variant of such attack also includes modification of unprotected parts when available in order for example to increase the payload size.

5. Requirements for Security Mitigations

The document assumes that Security protocols, algorithms, and implementations provide the security properties for which they are designed, an attack caused by a weakness in a cryptographic algorithm is out of scope. The algorithm used MUST follow the cryptographic guidance such as [RFC8247], [RFC8221] or [RFC7525]. In this context, when the document mentions encryption, it assumes authenticated encryption.

Protecting network connecting TSeS and NVEs which could be accessible to outside attackers is out of scope.

An attacker controlling an underlying network device may break the communication of the overlays by discarding or delaying the delivery of the packets passing through it. The security consideration to prevent this type of attack is out of scope of this document.

Securing communication between NVAs and NVEs is out of scope.

Selectively providing integrity / authentication, confidentiality / encryption of only portions of the Geneve packet is in scope. This will be the case if the Tenant Systems uses security protocol to protect its communications.

5.1. Protection Against Traffic Sniffing

The inner payload, unless protection is provided by the Tenant System reveals the content of the communication. This may be mitigate by the Tenant using application level security such as, for example JSON Web Encryption [RFC7516] or transport layer security such as DTLS

[RFC6347] or TLS [RFC8446] or IPsec/ESP [RFC4303]. However none of these security protocols are sufficient to protect the entire inner payload. IPsec/ESP still leave in clear the optional L2 layer information as well as the IP addresses and some IP options. In addition to these pieces of information, the use of TLS or DTLS reveals the transport layer protocol as well as ports. As a result, the confidentiality protection of the inner packet may be handled either entirely by the Geneve Overlay Provider, or partially by the Tenant or handled by both the Tenant and the Geneve Overlay Provider.

The Geneve Header contains information related to the Geneve communications or metadata designated as Geneve Information. Geneve Information is carried on the Geneve Outer Header, the Geneve Header (excluding Geneve Options) as well as in the Geneve Options. Geneve Information needs to be accessed solely by a NVE or transit device while other Geneve Information may need to be accessed by other transit devices. More specifically, a subset of the information contained in the Geneve Header (excluding Geneve Options) as well as a subset of (none, one or multiple Geneve Option) may be accessed by a transit device or the NVE while the others needs to be accessed by other transit devices. The confidentiality protection of the Geneve Information is handled by the Geneve Overlay Provider.

In addition to Geneve Information, the traffic generated for the Geneve overlay may be exposed to traffic volumetry and pattern analysis within a virtualized network. Confidentiality protection against traffic pattern recognition is handled by the Geneve Overlay Provider.

5.1.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-1: A secure deployment of a Geneve overlay SHOULD by default encrypt the inner payload. A Geneve overlay provider MAY disable this capability for example when encryption is performed by the Tenant System and that level of confidentiality is believed to be sufficient. In order to provide additional protection to traffic already encrypted by the Tenant the Geneve network operator MAY partially encrypt the clear part of the inner payload.
- o SEC-OP-2: A secure deployment of a Geneve overlay MUST evaluate the information associated to the leakage of Geneve Information carried by the Geneve Packet. When a risk analysis concludes that the risk of leaking sensitive information is too high, such Geneve Information MUST NOT be transmit in clear text.

- o SEC-OP-3: A secure deployment of a Geneve overlay MUST evaluate the risk associated to traffic pattern recognition. When a risk has been identified, traffic pattern recognition MUST be addressed with padding policies as well as generation of dummy packets.

5.1.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-1: Geneve security mechanism MUST provide the capability to encrypt the inner payload.
- o SEC-GEN-2: Geneve security mechanism SHOULD provide the capability to partially encrypt the inner payload header.
- o SEC-GEN-3: Geneve security mechanism MUST provide means to encrypt a single or a set of zero, one or multiple Geneve Options while leave other Geneve Options in clear. Reversely, a Geneve security mechanism MUST be able to leave a Geneve option in clear, while encrypting the others.
- o SEC-GEN-4: Geneve security mechanism MUST provide means to encrypt the information of Geneve Header (excluding Geneve Options). Reversely, a Geneve security mechanism MUST be able to leave in clear Geneve Header information (Geneve Options excluded) while encrypting the other.
- o SEC-GEN-5: Geneve security mechanisms MUST provide the ability to provide confidentiality protection between multiple nodes, i.e. multiple transit devices and a NVE.
- o SEC-GEN-6: Geneve security mechanism MUST provide the ability to pad a Geneve packet.
- o SEC-GEN-7: Geneve security mechanism MUST provide the ability to send dummy packets.

5.2. Protecting Against Traffic Injection

Traffic injection from a rogue non legitimate NVO3 Geneve overlay device or a rogue underlay transit device can target an NVE, a transit underlay device or a Tenant System. Targeting a Tenant's System requires a valid MAC and IP addresses of the Tenant's System.

When traffic between tenants is not protected, the rogue device may forward the modified packet over a valid (authenticated) Geneve Header. The crafted packet may for example, include a specifically crafted application payload for a specific Tenant Systems

application, with the intention to load the tenant specific application. Tenant's System may provide integrity protection of the inner payload by protect their communications using for example IPsec/ESP, IPsec/AH [RFC4302], TLS or DTLS. Such protection protects at various layers the Tenants from receiving spoofed packets, as any injected packet is expected to be discarded by the destination Tenant's System. Note IPsec/ESP with NULL encryption may be used to authenticate-only the layers above IP in which case the IP header remains unprotected. However IPsec/AH enables the protection of the entire IP packet, including the IP header. As a result, when Geneve encapsulates IP packets the Tenant has the ability to integrity protect the IP packet on its own, without relying on the Geneve overlay network. On the other hand, L2 layers remains unprotected. As encryption is using authenticated encryption, authentication may also be provided via encryption. At the time of writing the document DTLS 1.3 [I-D.ietf-tls-dtls13] is still a draft document and TLS 1.3 does not yet provide the ability for authenticate only the traffic. As such it is likely that the use of DTLS1.3 may not involve authentication-only cipher suites. Similarly to confidentiality protection, integrity protection may be handled either entirely by the Geneve Overlay Provider, or partially by the Tenant or handled by both the Tenant and the Geneve Overlay Provider.

In addition to confidentiality protection of the inner payload, integrity protection also prevents the Tenant System from receiving illegitimate packets that may disrupt the Tenant's System performance. The Geneve overlay network need to prevent the overlay to be used as a vector to spoof packets being steered to the Tenant's system. As a result, the Overlay Network Provider needs to ensure that inner packets steered to the Tenant's network are only originating from one Tenant System and not from an outsider using the Geneve Overlay to inject packets to one virtual network. As such, the destination NVE MUST be able to authenticate the incoming Geneve packets from the source NVE. This may be performed by the NVE authenticating the full Geneve Packet. When the Geneve Overlay wants to take advantage of the authentication performed by the Tenant System, the NVE should be able to perform some checks between the Geneve Header and the inner payload. Suppose two Geneve packets are composed of a Geneve Header (H1, and H2) and a inner payload (P1 and P2). Suppose H1, H2, P1 and P2 are authenticated. The replacement of P2 by P1 by an attacker will be detected by the NVE only if there is a binding between H2 and P2. Such integrity protection is handled by the Geneve Overlay Provider.

While traffic injection may target the Tenant's virtual network or a specific Tenant System, traffic injection may also target the Geneve Overlay Network by injecting Geneve Options that will affect the processing of the Geneve Packet. Updating the Geneve header and

option parameters such as setting an OAM bit, adding bogus option TLVs, or setting a critical bit, may result in different processing behavior, that could greatly impact performance of the overlay network and the underlay infrastructure and thus affect the tenants traffic delivery. As such, the Geneve Overlay should provide integrity protection of the Geneve Information present in the Geneve Header to guarantee Geneve processing is not altered.

The Geneve architecture considers transit devices that may process some Geneve Options. More specifically, a Geneve packet may have A subset of Geneve Information of the Geneve Header (excluding Geneve Options) as well as a set of zero, one or multiple of Geneve Options accessed by one or more transit devices. This information needs to be authenticated by a transit device while other options may be authenticated by other transit devices or the tunnel endpoint. The integrity protection is handled by the Geneve Overlay Provider and authentication MUST be performed prior any processing.

5.2.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-4: A secure deployment of a Geneve overlay MUST provide the capability authenticate the inner payload when encryption is not provided. A Geneve overlay provider MAY disable this capability for example when this is performed by the Tenant System and that level of integrity is believed to be sufficient. In order to provide additional protection to traffic already protected by the Tenant the Geneve network operator MAY partially protect the unprotected part of the inner payload.
- o SEC-OP-5: A secure deployment of a Geneve overlay MUST evaluate the risk associated to a change of the Geneve Outer Header, Geneve Header (excluding Geneve Options) and Geneve Option. When a risk analysis concludes that the risk is too high, this piece of information MUST be authenticated.
- o SEC-OP-6: A secure deployment of a Geneve overlay SHOULD authenticate communications between NVE to protect the Geneve Overlay infrastructure as well as the Tenants System's communications (Geneve Packet). A Geneve overlay provider MAY disable authentication of the inner packet and delegates it to the Tenant Systems when communications between Tenant's System is secured. This is NOT RECOMMENDED. Instead, it is RECOMMENDED that mechanisms binds the inner payload to the Geneve Header. To prevent injection between virtualized network, it is strongly

RECOMMENDED that at least the Geneve Header without Geneve Options is authenticated.

- o SEC-OP-7: A secure deployment of a Geneve overlay SHOULD NOT process data prior authentication. If that is not possible, the Geneve overlay provider SHOULD evaluate its impact.

5.2.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-8: Geneve security mechanism MUST provide the capability to authenticate the inner payload.
- o SEC-GEN-9: Geneve security mechanism SHOULD provide the capability to partially authenticate the inner payload header.
- o SEC-GEN-10: Geneve security mechanism MUST provide the capability to authenticate a single or a set of options while leave other Geneve Option unauthenticated. Reversely, a Geneve security mechanism MUST be able to leave a Geneve option unauthenticated, while encrypting the others.
- o SEC-GEN-11: Geneve security mechanism MUST provide means to authenticate the information of Geneve Header (Geneve Option excluded). Reversely, a Geneve security mechanism MUST be able to leave unauthenticated Geneve header information (Geneve Options excluded) while authenticating the other.
- o SEC-GEN-12: Geneve Security mechanism MUST provide means for a tunnel endpoint (NVE) to authenticate data prior it is being processed.
- o SEC-GEN-13: Geneve Security mechanism MUST provide means for a transit device to authenticate data prior it is being processed.

5.3. Protecting Against Traffic Redirection

A rogue device of the NVO3 overlay Geneve network or the underlay network may redirect the traffic from a virtual network to the attacker for passive or active attacks. If the rogue device is in charge of securing the Geneve packet, then Geneve security mechanisms are not intended to address this threat. More specifically, a rogue source NVE will still be able to redirect the traffic in clear text before protecting (and encrypting the packet). A rogue destination NVE will still be able to redirect the traffic in clear text after decrypting the Geneve packets. The same occurs with a rogue transit that is in charge of encrypting and decrypting a Geneve Option,

Geneve Option or any information. The security mechanisms are intended to protect a Geneve information from any on path node. Note that modern cryptography recommend the use of authenticated encryption. This section assumes such algorithms are used, and as such encrypted packets are also authenticated.

To prevent an attacker located in the middle between the NVEs and modifying the tunnel address information in the data packet header to redirect the data traffic, the solution needs to provide confidentiality protection for data traffic exchanged between NVEs.

Requirements are similar as those provided in section Section 5.1 to mitigate sniffing attacks and those provided in section Section 5.2 to mitigate traffic injection attacks.

5.4. Protecting Against Traffic Replay

A rogue device of the NVO3 overlay Geneve network or the underlay network may replay a Geneve packet, to load the network and/or a specific Tenant System with a modified Geneve payload. In some cases, such attacks may target an increase of the tenants costs.

When traffic between Tenant System is not protected against anti-replay. A packet even authenticated can be replayed. DTLS and IPsec provides anti replay mechanisms, so it is unlikely that authenticated Tenant's traffic is subject to replay attacks.

Similarly to integrity protection, the Geneve Overlay Provider should prevent the overlay to be used to replay packet to the Tenant's System. In addition, similarly to integrity protection, the Geneve Overlay network may also be a target of a replay attack, and NVE as well as transit devices should benefit from the same protection.

Given the proximity between authentication and anti-replay mechanisms and that most authentication mechanisms integrates anti-replay attacks, we RECOMMEND that authentication contains an anti-replay mechanisms.

5.4.1. Geneve Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-8: A secure deployment of a Geneve overlay MUST evaluate the communications subject to replay attacks. Communications that are subject to this attacks MUST be authenticated with an anti replay mechanism. Note that when partial authentication is provided, the part not covered by the authentication remains a

surface of attack. It is strongly RECOMMENDED that the Geneve Header is authenticated with anti replay protection.

5.4.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-14: Geneve Security mechanism MUST provide authentication with anti-replay protection.

5.5. Security Management

5.5.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-9: A secure deployment of a Geneve overlay MUST define the security policies that associates the encryption, and authentication associated to each flow between NVEs.
- o SEC-OP-10: A secure deployment of a Geneve overlay SHOULD define distinct material for each flow. The cryptographic depends on the nature of the flow (multicast, unicast) as well as on the security mechanism enabled to protect the flow.

5.5.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-15: A Geneve security mechanism MUST be managed via security policies associated for each traffic flow to be protected. Geneve overlay provider MUST be able to configure NVEs with different security policies for different flows. A flow MUST be identified at minimum by the Geneve virtual network identifier and the inner IP and transport headers, and optionally additional fields which define a flow (e.g., inner IP DSCP, IPv6 flow id, Geneve options).
- o SEC-GEN-16: A Geneve security mechanism MUST be able to assign different cryptographic keys to protect the unicast tunnels between NVEs respectively.
- o SEC-GEN-17: A Geneve security mechanisms, when multicast is used, packets, MUST be able to assign distinct cryptographic group keys to protect the multicast packets exchanged among the NVEs within different multicast groups. Upon receiving a data packet, an egress Geneve NVE MUST be able to verify whether the packet is

sent from a proper ingress NVE which is authorized to forward that packet.

6. IANA Considerations

There are no IANA consideration for this document.

7. Security Considerations

The whole document is about security.

Limiting the coverage of the authentication / encryption provides some means for an attack to craft special packets.

The current document details security requirements that are related to the Geneve protocol. Instead, [I-D.ietf-nvo3-security-requirements] provides generic architecture security requirement upon the deployment of an NVO3 overlay network. It is strongly recommended to read that document as architecture requirements also apply here. In addition, architecture security requirements go beyond the scope of Geneve communications, and as such are more likely to address the security needs upon deploying an Geneve overlay network.

8. Appendix

8.1. DTLS

This section compares how NVE communications using DTLS meet the security requirements for a secure Geneve overlay deployment. In this example DTLS is used over the Geneve Outer Header and secures the Geneve Header including the Geneve Options and the inner payload.

The use of DTLS MAY fill the security requirements for a secure Geneve deployment. However DTLS cannot be considered as the Geneve security mechanism enabling all Geneve deployments. To ease the reading of the Requirements met by DTLS or IPsec, the requirements list indicates with Y (Yes) when the requirement and N (No) when the requirement is not met. In addition, an explanation is provided on the reasoning. This section is not normative and its purpose is limited to illustrative purpose.

8.1.1. Operational Security Requirements

This section shows how DTLS may secure some Geneve deployments. Some Geneve deployments may not be secured by DTLS, but that does not exclude DTLS from being used.

- o SEC-OP-1 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will provide confidentiality to the full Geneve Packet which contains the inner payload. As such the use of DTLS meets SEC-OP-1. Note that DTLS does not provide partial encryption and as such the Geneve Overlay Provider may not benefit from the encryption performed by the Tenant if performed, which may result in some portion of the payload being encrypted twice.
- o SEC-OP-2 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and associated metadata. Only the UDP port is leaked which could be acceptable. As such, the use of DTLS meets SEC-OP-2.
- o SEC-OP-3 (Y/N): A deployment using DTLS between NVEs will not be able to send dummy packets or pad Geneve Packet unless this is managed by the Geneve packet itself. DTLS does not provide the ability to send dummy traffic, nor to pad. As a result DTLS itself does not meet this requirement. This requirement may be met if handled by the Geneve protocol. As such SEC-OP-3 may not be met for some the deployment. However, it is not a mandatory requirement and as such it is likely that the use of DTLS SEC-OP-3 is met.
- o SEC-OP-4 (Y): Similarly to SEC-OP-1, A deployment using DTLS between NVEs provides integrity protection to the full Geneve Packet which includes the inner payload. As such the use of DTLS meets SEC-OP-4. Note that DTLS 1.2 provides integrity-only cipher suites while DTLS 1.3 does not yet. As a result, the use of DTLS 1.3 may provide integrity protection using authenticated encryption.
- o SEC-OP-5 (Y): Similarly to SEC-OP-2, A deployment using DTLS between NVE authenticates the full Geneve Packet which includes the Geneve Header. Only the UDP port is left unauthenticated. As such, the use of DTLS meets SEC-OP-5.
- o SEC-OP-6 (Y): A deployment using DTLS between NVE authenticates NVE-to-NVE communications and the use of DTLS meets SEC-OP-6.
- o SEC-OP-7 (Y/N): A deployment using DTLS between NVEs is not compatible with a Geneve architecture that includes transit devices. When the DTLS session uses a non NULL encryption cipher suite, the transit device will not be able to access it. When the NULL encryption cipher suite is used, the transit device may be able to access the data, but will not be able to authenticate it prior to processing the packet. As such the use of DTLS only

meets SEC-OP-7 for deployment that do not include any transit devices.

- o SEC-OP-8 (Y): A deployment using DTLS between NVEs provides anti-replay protection and so, the use of DTLS meets SEC-OP-8.
- o SEC-OP-9 (Y/N): DTLS does not define any policies. Instead DTLS process is bound to an UDP socket. As such handling of flow policies is handled outside the scope of DTLS. As such SEC-OP-9 is met outside the scope of DTLS.
- o SEC-OP-10 (N): DTLS session may be established with specific material, as such it is possible to assign different material for each flow. However, the binding between flow and session is performed outside the scope of DTLS. In addition, DTLS does not support multicast. As such, the use of DTLS may only meets SEC-OP-10 in the case of unicast communications.

8.1.2. Geneve Security Requirements

This section shows that DTLS cannot be used as a generic Geneve security mechanism to secure Geneve deployments. A Geneve security mechanism would need to meet all SEC-GEN requirements.

- o SEC-GEN-1 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will provide confidentiality to the full Geneve Packet which contains the inner payload. As such the use of DTLS meets SEC-GEN-1.
- o SEC-GEN-2 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will not be able to partially encrypt the inner payload header. However such requirement is not set a mandatory so the use of DTLS meets SEC-GEN-2
- o SEC-GEN-3 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and all Geneve Options. However DTLS does not provides any means to selectively encrypt or leave in clear text a subset of Geneve Options. As a result the use of DTLS does not meet SEC-GEN-3.
- o SEC-GEN-4 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively encrypt some information of the Geneve Header. As such the use of DTLS does not meet SEC-GEN-5.

- o SEC-GEN-5 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite provides end-to-end security between the NVEs and as such does not permit the interaction of one or multiple on-path transit devices. As such the use of DTLS does not meet SEC-GEN-5.
- o SEC-GEN-6 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite does not provide padding facilities. This requirements is not met by DTLS itself and needs to be handled by Geneve and specific options. As a result, the use of DTLS does not meet SEC-GEN-6
- o SEC-GEN-7 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite does not provide the ability to send dummy packets. This requirements is not met by DTLS itself and needs to be handled by Geneve and specific options. As a result, the use of DTLS does not meet SEC-GEN-7.
- o SEC-GEN-8 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite or a NULL encryption cipher suite provide authentication of the inner payload. As such the use of DTLS meets SEC-GEN-8.
- o SEC-GEN-9 (Y): A deployment using DTLS between NVEs does not provide the ability to partially authenticate the inner payload header. However such requirement is not set a mandatory so the use of DTLS meets SEC-GEN-9
- o SEC-GEN-10 (N): A deployment using DTLS between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively encrypt some information of the Geneve Header. As such the use of DTLS meets SEC-GEN-10.
- o SEC-GEN-11 (N): A deployment using DTLS between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively authenticate some information of the Geneve Header. As such the use of DTLS does not meet SEC-GEN-11.
- o SEC-GEN-12 (Y): A deployment using DTLS between NVEs authenticates the data prior the data is processed by the NVE. As such, the use of DTLS meets SEC-GEN-12.
- o SEC-GEN-13 (N): A deployment using DTLS between NVEs authenticates the data when the tunnel reaches the NVE. As a result the transit device is not able to authenticate the data prior accessing it and the use of DTLS does not meet SEC-GEN-13.

- o SEC-GEN-14 (Y): DTLS provides anti-replay mechanism as such, the use of DTLS meets SEC-GEN-14.
- o SEC-GEN-15 (N): DTLS itself does not have a policy base mechanism. As a result, the classification of the flows needs to be handled by a module outside DTLS. In order to meet SEC-GEN-15 further integration is needed and DTLS in itself cannot be considered as meeting SEC-GEN-15.
- o SEC-GEN-16 (Y): DTLS is able to assign various material to each flows, as such the use of DTLS meets SEC-GEN-16.
- o SEC-GEN-17 (N): DTLS does not handle multicast communications. As such the use of DTLS does not meet SEC-GEN-17.

8.2. IPsec

This section compares how NVE communications using IPsec/ESP or IPsec/AH meet the security requirements for a secure Geneve overlay deployment. In this example secures the Geneve IP packet including Outer IP header, the Geneve Outer Header, the Geneve Header including Geneve Options and the inner payload.

The use of IPsec/ESP or IPsec/AH share most of the analysis performed for DTLS. The main advantages of using IPsec would be that IPsec supports multicast communications and natively supports flow based security policies. However, the use of these security policies in a context of Geneve is not natively supported.

As a result, the use of IPsec MAY fill the security requirements for a secure Geneve deployment. However IPsec cannot be considered as the Geneve security mechanism enabling all Geneve deployments.

8.2.1. Operational Security Requirements

This section shows how IPsec may secure some Geneve deployments. Some Geneve deployments may not be secured by IPsec, but that does not exclude IPsec from being used.

- o SEC-OP-1 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption will provide confidentiality to the full Outer IP payload of the Geneve Packet which contains the inner payload. As a result, such deployments meet SEC-OP-1. Note that IPsec/ESP does not provide partial encryption and as such the Geneve Overlay Provider may not benefit from the encryption performed by the Tenant if performed, which may result in some portion of the payload being encrypted twice.

- o SEC-OP-2 (Y): A deployment using IPsec/ESP between NVEs with a non NULL encryption encrypts the Outer IP payload Geneve IP Packet which includes the Geneve Header and associated information. As such SEC-OP-2 is met.
- o SEC-OP-3 (Y): A deployment using IPsec/ESP between NVEs will be able to send dummy packets or pad Geneve Packet. As such OP-SEC-3 is met.
- o SEC-OP-4 (Y): Similarly to SEC-OP-1, A deployment using IPsec/ESP or IPsec/AH between NVEs provides integrity protection to the full Geneve Packet which includes the inner payload. As such SEC-OP-4 is met.
- o SEC-OP-5 (Y): Similarly to SEC-OP-2, A deployment using IPsec/ESP or IPsec/AH between NVE authenticates the full Geneve Packet which includes the Geneve Header. As such SEC-OP-5 is met as well.
- o SEC-OP-6 (Y): A deployment using IPsec/ESP or IPsec/AH between NVE authenticates NVE-to-NVE communications and SEC-OP-6 is met.
- o SEC-OP-7 (Y/N): A deployment using IPsec between NVEs is not compatible with a Geneve architecture that includes transit devices. When IPsec/ESP with a non NULL encryption is used, the transit device will not be able to access it. When IPsec/AH or IPsec/ESP with the NULL encryption is used, the transit device may be able to access the data, but will not be able to authenticate it prior to processing the packet. As SEC-OP-7 is only met for deployment that do not include any transit devices.
- o SEC-OP-8 (Y): A deployment using IPsec between NVEs provides anti-replay protection and so meets SEC-OP-8.
- o SEC-OP-9 (Y/N): IPsec enables the definition of security policies. As such IPsec is likely to handle a per flow security. However the traffic selector required for Geneve flows may not be provided natively by IPsec. As such Sec-OP-9 is only partially met.
- o SEC-OP-10 (Y): IPsec session may be established with specific material, as such it is possible to assign different material for each flow. In addition IPsec supports multicats communications. As such SEC-OP-10 is met.

8.2.2. Geneve Security Requirements

This section shows that IPsec cannot be used as a generic Geneve security mechanism to secure Geneve deployments. A Geneve security mechanism would need to meet all SEC-GEN requirements.

- o SEC-GEN-1 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption provide confidentiality to the full Geneve Packet which contains the inner payload. As such IPsec/ESP meets SEC-GEN-1.
- o SEC-GEN-2 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption will not be able to partially encrypt the inner payload header. However such requirement is not set a mandatory so IPsec/ESP meets SEC-GEN-2
- o SEC-GEN-3 (N): A deployment using IPsec between NVEs with an non NULL encryption encrypts the Outer IP payload of the Geneve Packet which includes the Geneve Header and all Geneve Options. However IPsec/ESP does not provides any means to selectively encrypt or leave in clear text a subset of Geneve Options. As a result SEC-GEN-3 is not met.
- o SEC-GEN-4 (N): A deployment using IPsec/ESP between NVEs with an non NULL encryption encrypts the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec/ESP does not provides means to selectively encrypt some information of the Geneve Header. As such SEC-GEN-5 is not met.
- o SEC-GEN-5 (N): A deployment using IPsec between NVEs with an non NULL encryption provides end-to-end security between the NVEs and as such does not permit the interaction of one or multiple on-path transit devices. As such IPsec/ESP does not meet SEC-GEN-5.
- o SEC-GEN-6 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption provides padding facilities and as such IPsec/ESP meets SEC-GEN-6.
- o SEC-GEN-7 (Y): A deployment using IPsec between NVEs with an non NULL encryption cipher provides the ability to send dummy packets. As such IPsec/ESP meets SEC-GEN-7.
- o SEC-GEN-8 (Y): A deployment using IPsec/ESP or IPsec/AH authenticates the inner payload. As such SEC-GEN-8 is met.
- o SEC-GEN-9 (Y): A deployment using IPsec/AH or IPsec/ESP between NVEs does not provide the ability to partially authenticate the inner payload header. However such requirement is not set a mandatory so IPsec meets SEC-GEN-9
- o SEC-GEN-10 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec does not provides

means to selectively encrypt some information of the Geneve Header. As such SEC-GEN-10 is not met.

- o SEC-GEN-11 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec does not provides means to selectively authenticate some information of the Geneve Header. As such SEC-GEN-11 is not met.
- o SEC-GEN-12 (Y): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the data prior the data is processed by the NVE. As such SEC-GEN-12 is met.
- o SEC-GEN-13 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the data when the tunnel reaches the NVE. As a result the transit device is not able to authenticate the data prior accessing it and SEC-GEN-13 is not met.
- o SEC-GEN-14 (Y): IPsec/ESP and IPsec/AH provides anti-replay mechanism as such SEC-GEN-14 is met.
- o SEC-GEN-15 (N): IPsec is a policy base architecture. As a result, the classification of the flows needs to be handled by IPsec. However, the traffic selector available are probably not those required by Geneve and further integration is needed. As such SEC-GEN-15 is not met.
- o SEC-GEN-16 (Y): IPsec is able to assign various material to each flows, as such SEC-GEN-16 is met.
- o SEC-GEN-17 (Y): IPsec handles mutlicast communications. As such SEC-GEN-17 is met.

9. Acknowledgments

We would like to thank Ilango S Ganaga, Magnus Nystroem for their useful reviews and clarifications as well as Matthew Bocci, Sam Aldrin and Ignas Bagdona for moving the work forward.

10. References

10.1. Normative References

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

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [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, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

10.2. Informative References

- [I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-09 (work in progress), February 2019.
- [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.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-30 (work in progress), November 2018.

Authors' Addresses

Daniel Migault
Ericsson
8275 Trans Canada Route
Saint Laurent, QC 4S 0B6
Canada

EMail: daniel.migault@ericsson.com

Sami Boutros
VMware, Inc.

EMail: boutros@vmware.com<

Dan Wings
VMware, Inc.

EMail: dwing@vmware.com

Suresh Krishnan
Kaloom

EMail: suresh@kaloom.com

BFD Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 27, 2019

X. Min
G. Mirsky
ZTE
November 23, 2018

BFD for Geneve
draft-xiao-bfd-geneve-00

Abstract

This document describes the use of the Bidirectional Forwarding Detection (BFD) protocol in Generic Network Virtualization Encapsulation (Geneve) overlay networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 27, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	2
1.1.1. Terminology	2
1.1.2. Requirements Language	3
2. BFD Packet Transmission over Geneve Tunnel	3
2.1. BFD Packet Encapsulation in Geneve	3
2.1.1. BFD Encapsulation With IP/UDP Header	3
2.1.2. BFD Encapsulation Without IP/UDP Header	5
3. Reception of BFD packet from Geneve Tunnel	7
3.1. Demultiplexing of the BFD packet	8
4. Security Considerations	8
5. IANA Considerations	9
6. Acknowledgements	9
7. Normative References	9
Authors' Addresses	10

1. Introduction

"Generic Network Virtualization Encapsulation" (Geneve) [I-D.ietf-nvo3-geneve] provides a generic tunneling protocol that is applicable to many scenarios, including an encapsulation scheme that allows virtual machines (VMs) to communicate in a data center network.

This document describes the use of Bidirectional Forwarding Detection (BFD) protocol for Geneve to enable monitoring continuity of the path between Network Virtualization Edges (NVEs) and/or availability of a replicator service node using BFD.

The use cases and the deployment of BFD for Geneve are consistent with what's described in Section 3 and Section 4 of [I-D.ietf-bfd-vxlan]. The main difference between Geneve and "Virtual eXtensible Local Area Network" (VXLAN) [RFC7348] encapsulation is that Geneve supports multi-protocol payload and variable length options.

1.1. Conventions Used in This Document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

Geneve: Generic Network Virtualization Encapsulation

NVE: Network Virtualization Edge

VFI: Virtual Forwarding Instance

VM: Virtual Machine

VNI: Virtual Network Identifier

VXLAN: Virtual eXtensible Local Area Network

1.1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. BFD Packet Transmission over Geneve Tunnel

BFD packet MUST be encapsulated and sent to a remote NVE using one of the options described in Section 2.1. Implementations SHOULD ensure that the BFD packets follow the same lookup path as Geneve data packets within the sender system.

2.1. BFD Packet Encapsulation in Geneve

Concerning whether or not the Geneve data packets include an IP protocol data unit, this document defines three options of BFD packet encapsulation in Geneve.

2.1.1. BFD Encapsulation With IP/UDP Header

If the Protocol Type field (as defined in Section 3.4 of [I-D.ietf-nvo3-geneve]) of data packets indicates that there exists an inner IP header, i.e., the Protocol Type equals to 0x6558 (Ethernet frame), or 0x0800 (IPv4), or 0x86DD (IPv6), or 0x8847 (MPLS), or 0x8848 (MPLS with the upstream-assigned label), then BFD packets are encapsulated in Geneve as described below. The Geneve packet format over IPv4 is defined in Section 3.1 of [I-D.ietf-nvo3-geneve]. The Geneve packet format over IPv6 is defined in Section 3.2 of [I-D.ietf-nvo3-geneve]. The Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve]. Note that the outer IP header and the inner IP header may not be of the same address family, in other words, outer IPv6 header accompanied with inner IPv4 header and outer IPv4 header accompanied with inner IPv6 header are both possible.

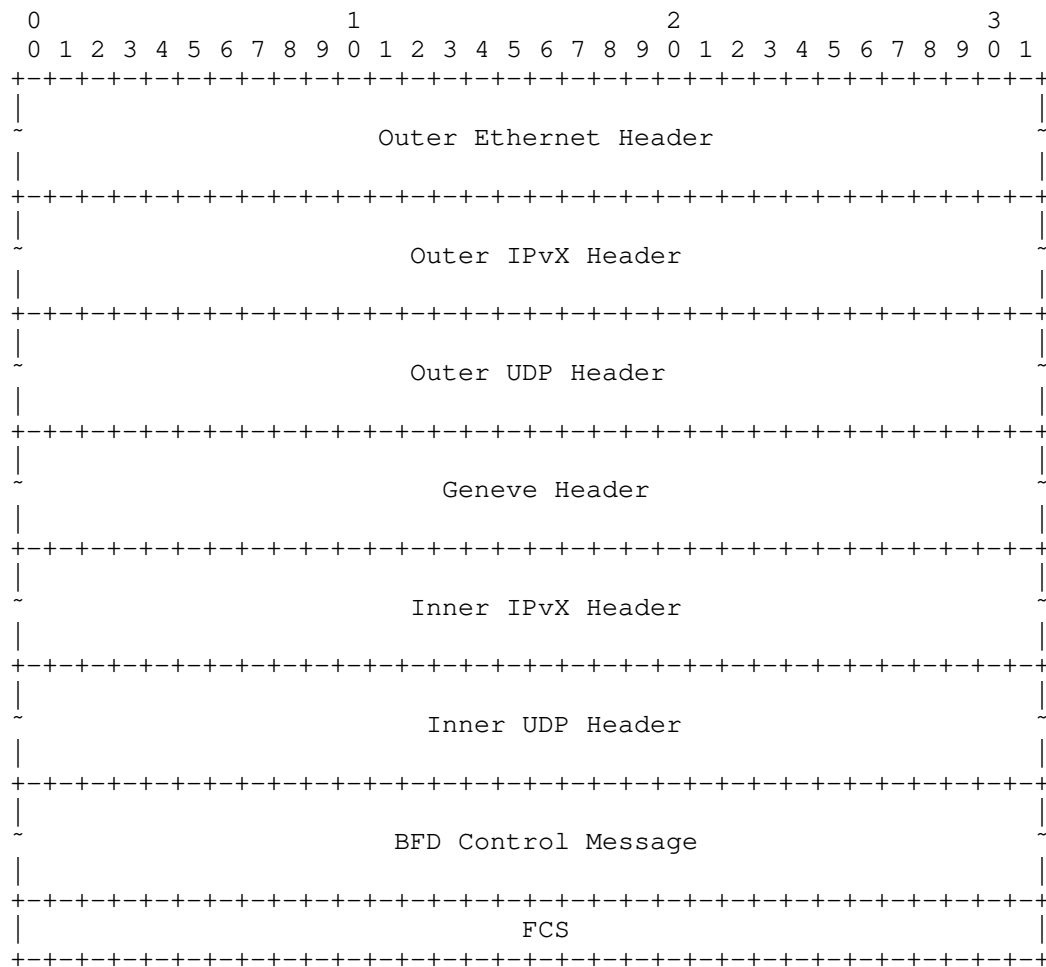


Figure 1: Geneve Encapsulation of BFD Control Message With the Inner IP/UDP Header

When the BFD packets are encapsulated in Geneve in this way, the BFD packet MUST be carried inside the inner IP packet of the Geneve packet. The inner IP packet carrying the BFD payload has the following format:

IP header:

Source IP: IP address of the originating NVE.

Destination IP: IP address of the terminating NVE.

TTL: MUST be set to 1 to ensure that the BFD packet is not routed within the L3 underlay network.

The fields of the UDP header and the BFD control packet are encoded as specified in [RFC5881].

When the BFD packets are encapsulated in Geneve in this way, the Geneve header SHOULD follow the value set below.

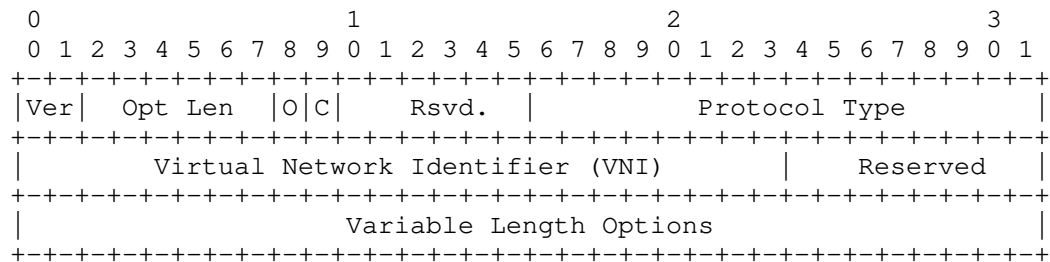


Figure 2: Geneve Header

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

[Ed.Note]: Use of O bit is still being discussed in the NVO3 WG, so the value is undetermined.

C bit SHOULD be set to 0.

Protocol Type field SHOULD be set to 0x0800 (IPv4) or 0x86DD (IPv6).

2.1.2. BFD Encapsulation Without IP/UDP Header

Alternatively to the use of the inner IP/UDP header to demultiplex BFD control packet by the value of the destination UDP port, BFD control packet MAY be encapsulated without the inner IP/UDP header. The BFD control packet MAY be identified directly in the Geneve header or through Geneve OAM shim. In either case, the Outer IP/UDP and Geneve headers MUST be encoded by the sender as defined in [I-D.ietf-nvo3-geneve].

Figure 3 displays the layout of the Ethernet frame with BFD control packet encapsulated in Geneve without the use of IP/UDP header and identified by the value TBA1 (to be assigned by IANA) of the Protocol Type field.

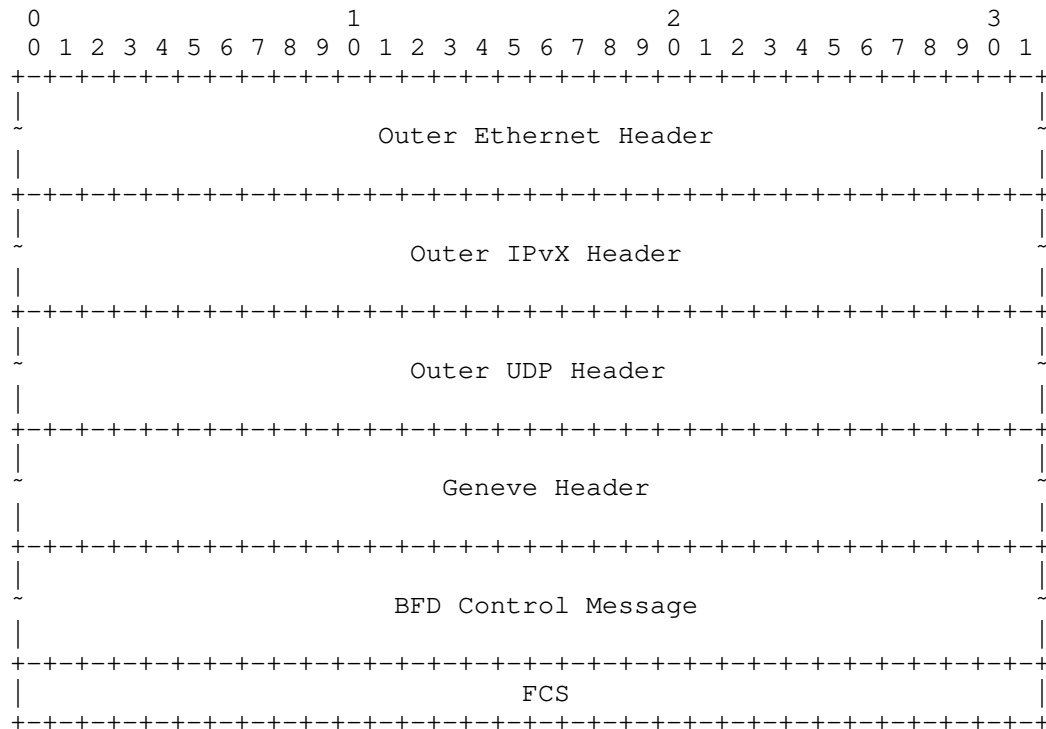


Figure 3: Geneve Encapsulation of BFD Control Message Without the Inner IP/UDP Header

When the BFD packets are encapsulated in Geneve in this way, the BFD packet MUST immediately follow the Geneve header, and the Geneve header SHOULD follow the value set below.

Opt Len field SHOULD be set to 0, which indicates there isn't any variable length option.

[Ed.Note]: Use of O bit is still being discussed in the NVO3 WG, so the value is undetermined.

C bit SHOULD be set to 0.

Also, if BFD control packet is encapsulated in Geneve without the use of IP/UDP header, the BFD control packet MAY be identified through the Geneve OAM shim. The layout of the Ethernet frame is shown in Figure 4. Protocol Type field MUST be set to the value TBA2 (to be assigned by IANA) which indicates a Geneve OAM shim that will have a field to indicate the inner BFD control packet. Definition of the

format of the Geneve OAM shim is outside the scope of this document. The Geneve OAM shim immediately follows the Geneve header, and the BFD control packet immediately follows the Geneve OAM shim.

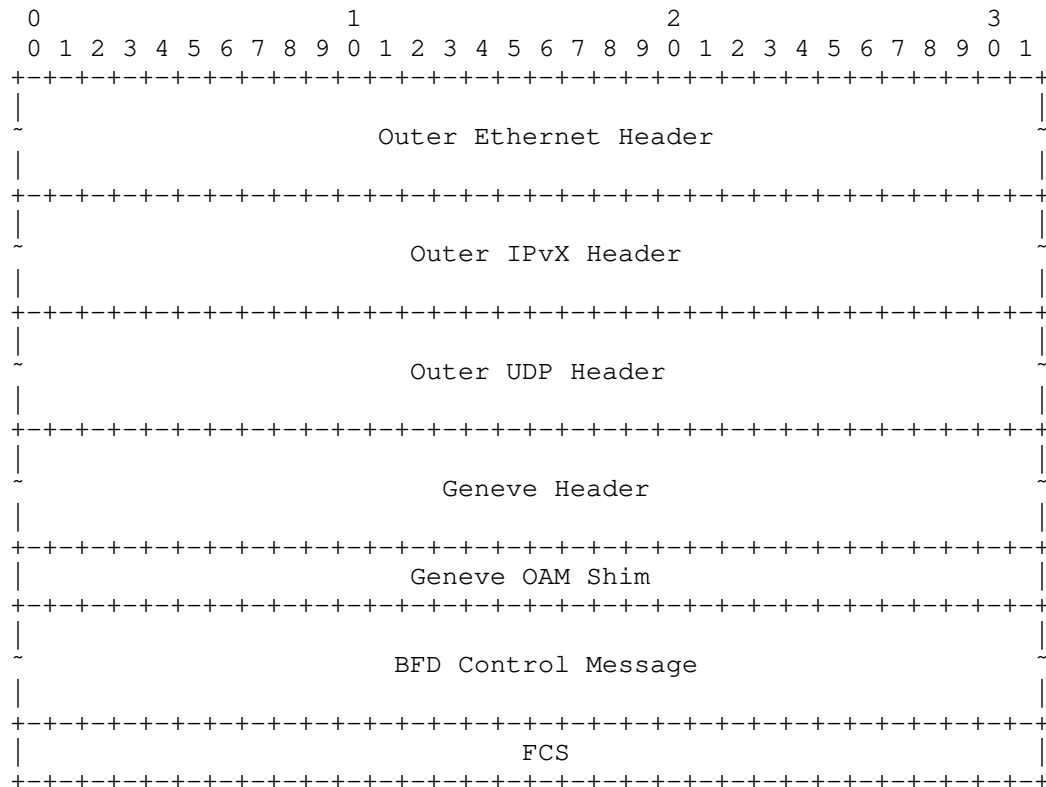


Figure 4: Geneve Encapsulation of BFD Control Message With Geneve OAM Shim

3. Reception of BFD packet from Geneve Tunnel

Once a packet is received, NVE MUST validate the packet as described in [I-D.ietf-nvo3-geneve].

If the Protocol Type field equals 0x0800 (IPv4) or 0x86DD (IPv6), and the Destination IP of the inner IP packet matches the IP address of the NVE, the UDP destination port and the TTL of the inner IP packet MUST be validated to determine whether BFD can process the received packet. BFD packet with inner IP set to NVE MUST NOT be forwarded to VMs.

If the Protocol Type field equals the value TBA1 (to be assigned by IANA) which indicates an inner BFD control message, the received packet MUST be processed by BFD and MUST NOT be forwarded to VMs.

If the Protocol Type field equals the value TBA2 (to be assigned by IANA) which indicates a Geneve OAM shim that will have a field to indicate the inner BFD control message, the received packet MUST be processed by BFD and MUST NOT be forwarded to VMs. This case is for further study.

To ensure BFD detects the proper configuration of Virtual Network Identifier (VNI) in a remote NVE, a lookup SHOULD be performed with the MAC-DA/IP-DA/MPLS-Label and VNI as key in the Virtual Forwarding Instance (VFI) table of the originating/terminating NVE to exercise the VFI associated with the VNI.

3.1. Demultiplexing of the BFD packet

If the Protocol Type field equals 0x0800 (IPv4) or 0x86DD (IPv6), demultiplexing of IP BFD packet has been defined in Section 3 of [RFC5881]. Since multiple BFD sessions may be running between two NVEs, there needs to be a mechanism for demultiplexing received BFD packets to the proper session. The procedure for demultiplexing packets with Your Discriminator equal to 0 is different from [RFC5880]. For such packets, the BFD session MUST be identified using the inner headers, i.e., the source IP and the destination IP present in the IP header carried by the payload of the Geneve encapsulated packet. The VNI of the packet SHOULD be used to derive interface-related information for demultiplexing the packet. If BFD packet is received with non-zero Your Discriminator, then BFD session MUST be demultiplexed only with Your Discriminator as the key.

If the Protocol Type field equals the value TBA1 (to be assigned by IANA) which indicates an inner BFD control message, or the value TBA2 (to be assigned by IANA) which indicates a Geneve OAM shim that will have a field to indicate the inner BFD control message, the VNI of the packet SHOULD be used to derive interface-related information for demultiplexing the packet, demultiplexing of BFD packet MUST rely on non-zero Your Discriminator as the key.

4. Security Considerations

This document does not raise any additional security issues beyond those of the specifications referred to in the list of normative references.

5. IANA Considerations

In the Geneve Protocol Type registry defined in [ETYPES], a new BFD Control Message or Geneve OAM Shim is requested from IANA as follows:

Geneve Protocol Type	Description	Semantics Definition	Reference
TBA1	BFD Control Message	Section 3.1	This Document
TBA2	Geneve OAM Shim	Section 3.1	This Document

Table 1: New BFD Control Message or Geneve OAM shim Ethertype

6. Acknowledgements

To be added.

7. Normative References

- [ETYPES] The IEEE Registration Authority, "IEEE 802 Numbers", 2013, <<http://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xml>>.
- [I-D.ietf-bfd-vxlan] Networks, J., Paragiri, S., Govindan, V., Mudigonda, M., and G. Mirsky, "BFD for VXLAN", draft-ietf-bfd-vxlan-03 (work in progress), October 2018.
- [I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-08 (work in progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.

- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [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, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Authors' Addresses

Xiao Min
ZTE
Nanjing
China

Phone: +86 25 88016574
Email: xiao.min2@zte.com.cn

Greg Mirsky
ZTE
USA

Email: gregimirsky@gmail.com

INTERNET-DRAFT
Intended Status: Standards Track

Mingui Zhang
Yuan Gao
Haibo Wang
Bing Liu
Huawei
Fu Qiao
China Mobile
March 8, 2019

Expires: September 9, 2019

Base YANG Data Model for NVO3 Protocols
draft-zhang-nvo3-yang-cfg-05.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Acronyms and Terminology	3
2.1. Acronyms	3
2.2. Terminology	3
3. The YANG Data Model for NVO3	3
3.1. The Configuration Parameters	4
3.1.1. NVE ID	4
3.1.2. Virtual Network Instance	4
3.1.3. Flags in the Header	4
3.1.4. BUM Mode	4
3.2. Statistics	4
3.3. Model Structure	4
3.4. YANG Module	7
4. Security Considerations	26
5. IANA Considerations	26
Acknowledgements	27
6. References	27
6.1. Normative References	27
6.2. Informative References	27
Author's Addresses	29

1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN, NVGRE, GENEVE and VXLAN-GPE, enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE may require configuration of TLVs at the NVE. The base model can be augmented to accommodate these new solutions.

2. Acronyms and Terminology

2.1. Acronyms

NVO3: Network Virtualization Overlays

VNI: Virtual Network Instance

BUM: Broadcast, Unknown Unicast, Multicast traffic

BUM: Broadcast, Unknown Unicast, Multicast traffic

2.2. Terminology

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 RFC 2119 [RFC2119].

Familiarity with [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

3. The YANG Data Model for NVO3

The NVO3 base YANG model is divided in three containers. The first container contains writable parameters. The second container contains the writable enablers per VNI for the statistical operational states as well as the status of these enablers. The third container contains the statistical operational states.

3.1. The Configuration Parameters

3.1.1. NVE ID

The 'nves' list contains NVEs under the management. The NVE is identified using the 'srcAddr', which is the underlay IP address of the NVE.

3.1.2. Virtual Network Instance

A Virtual Network Instance ('VNI') is a specific VN instance on an NVE [RFC7365]. At each NVE, a Tenant System is connect to VNIs through Virtual Access Points (VAP). VAPs can be physical ports or virtual ports identified by the bridge domain Identifier ('bdId'). The mapping between VNI and bdId is managed by the operator.

3.1.3. Flags in the Header

Flags in the NVO3 header are to be configured. For VxLAN, the I bit of 'flag' MUST be set to 1 while other 7 bits are reserved and MUST be set to zero on transmission and ignored on receipt. For NVGRE, bits in position 0 and 3 MUST be set to zero. The bit in position 2 MUST be set to 1. Bits from position 4 through 12 are reserved and MUST be set to zero on transmission and ignored on receipt.

3.1.4. BUM Mode

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

If ingress replication is used, the receiver addresses are listed in 'peers'. If multicast proxy is used, the proxy's address is given in "flood-proxy". If the choice is point to multipoint tunnels, the multicast address is given as 'multiAddr'.

3.2. Statistics

Operators can determine whether a NVE should gather statistic values on a per-VNI base. The enablers are contained in the 'nvo3Info' list as 'statisticsEnable' leaf.

If the gathering for a VNI is enabled, the statistical information about the local NVEs, the remote NVEs, the flows and the MAC addresses will be collected by the NVEs in this VNI.

3.3. Model Structure

```

module: ietf-nvo3-base
+--rw nov3
  +--rw common
    | +--rw nvo3-enable?    boolean
  +--rw nves
    +--rw nve* [if-name source-interface]
      +--rw if-name          ifName
      +--rw vtep-ip?         inet:ipv4-address-no-zone
      +--rw ipv6-vtep-ip?    inet:ipv6-address-no-zone
      +--rw source-interface ifName
      +--rw mac-address?     yang:mac-address
      +--rw bypass-vtep-ip?  inet:ipv4-address-no-zone
      +--rw vni-members
        +--rw vni-member* [vni-id]
          +--rw vni-id      uint32
          +--rw protocol-bgp? protocolType
          +--rw peers
            +--rw peer* [peer-ip]
              +--rw peer-ip      inet:ipv4-address-no-zone
              +--rw out-vni-id?   uint32
              +--rw split-horizon-group? string
          +--rw ipv6-peers
            +--rw ipv6-peer* [ipv6-peer-ip]
              +--rw ipv6-peer-ip  inet:ipv6-address-no-zone
          +--rw flood-proxys
            +--rw flood-proxy* [peer-ip]
              +--rw peer-ip      inet:ipv4-address-no-zone
          +--rw mcast-group* [mcast-group]
            +--rw mcast-group    inet:ipv4-address-no-zone
      +--rw vni-map-l2vpns
        +--rw vni-map-l2vpn* [vni-id]
          +--rw l2vpn-id      uint32
          +--rw l2vpn-name?   string
          +--rw vni-id        uint32
          +--rw split-horizon-group? string
      +--rw vni-map-l3vpns
        +--rw vni-map-l3vpn* [l3vpn-name]
          +--rw l3vpn-name    vrfName
          +--rw vni-id?       uint32
      +--ro vni-statistics-infos
        +--ro vni-statistic-info* [vni-id]
          +--ro vni-id        uint32
          +--ro vni-statistics
            +--ro rx-bits-persec?    uint64
            +--ro rx-pkts-persec?    uint64
            +--ro tx-bits-persec?    uint64
            +--ro tx-pkts-persec?    uint64
            +--ro rx-pkts?           uint64

```

```

    +---ro rx-bytes?                uint64
    +---ro tx-pkts?                uint64
    +---ro tx-bytes?                uint64
    +---ro rx-unicast-pkts?        uint64
    +---ro rx-multicast-pkts?      uint64
    +---ro rx-broadcast-pkts?      uint64
    +---ro drop-unicast-pkts?      uint64
    +---ro drop-multicast-pkts?    uint64
    +---ro drop-broadcast-pkts?    uint64
    +---ro tx-unicast-pkts?        uint64
    +---ro tx-multicast-pkts?      uint64
    +---ro tx-broadcast-pkts?      uint64
+---rw vnipeer-statistics-cfgs
|   +---rw vnipeer-satistics-cfg* [vni-id peer-ip]
|   |   +---rw vni-id            uint32
|   |   +---rw peer-ip          inet:ipv4-address-no-zone
+---ro vnipeer-statistics-infos
|   +---ro vnipeer-satistics-info* [vni-id source-ip peer-ip]
|   |   +---ro vni-id            uint32
|   |   +---ro source-ip        inet:ipv4-address-no-zone
|   |   +---ro peer-ip          inet:ipv4-address-no-zone
|   +---ro vnipeer_statistics
|   |   +---ro rx-bits-persec?    uint64
|   |   +---ro rx-pkts-persec?    uint64
|   |   +---ro tx-bits-persec?    uint64
|   |   +---ro tx-pkts-persec?    uint64
|   |   +---ro rx-pkts?          uint64
|   |   +---ro rx-bytes?          uint64
|   |   +---ro tx-pkts?          uint64
|   |   +---ro tx-bytes?          uint64
|   |   +---ro rx-unicast-pkts?    uint64
|   |   +---ro rx-multicast-pkts?  uint64
|   |   +---ro rx-broadcast-pkts?  uint64
|   |   +---ro drop-unicast-pkts?  uint64
|   |   +---ro drop-multicast-pkts? uint64
|   |   +---ro drop-broadcast-pkts? uint64
|   |   +---ro tx-unicast-pkts?    uint64
|   |   +---ro tx-multicast-pkts?  uint64
|   |   +---ro tx-broadcast-pkts?  uint64
+---rw ipv6-vnipeer-statistics-cfgs
|   +---rw ipv6-vnipeer-statistics-cfg* [vni-id ipv6-source-ip ipv6-pe
er-ip]
|   |   +---rw vni-id            uint32
|   |   +---rw ipv6-source-ip    inet:ipv6-address-no-zone
|   |   +---rw ipv6-peer-ip      inet:ipv6-address-no-zone
+---ro ipv6-vnipeer-statistics-infos
|   +---ro ipv6-vnipeer-statistics-info* [vniId ipv6-source-ip ipv6-pe
er-ip]
|   |   +---ro vniId            uint32
|   |   +---ro ipv6-source-ip    inet:ipv6-address-no-zone

```

```

    +--ro ipv6-peer-ip                inet:ipv6-address-no-zone
    +--ro ipv6_vnipeer_statistics
      +--ro rx-bits-persec?            uint64
      +--ro rx-pkts-persec?           uint64
      +--ro tx-bits-persec?           uint64
      +--ro tx-pkts-persec?           uint64
      +--ro rx-pkts?                  uint64
      +--ro rx-bytes?                 uint64
      +--ro tx-pkts?                  uint64
      +--ro tx-bytes?                 uint64
      +--ro rx-unicast-pkts?          uint64
      +--ro rx-multicast-pkts?        uint64
      +--ro rx-broadcast-pkts?        uint64
      +--ro drop-unicast-pkts?        uint64
      +--ro drop-multicast-pkts?      uint64
      +--ro drop-broadcast-pkts?      uint64
      +--ro tx-unicast-pkts?          uint64
      +--ro tx-multicast-pkts?        uint64
      +--ro tx-broadcast-pkts?        uint64
+--ro vnipeer-infos
  +--ro vnipeer-info* [vni-id source-ip peer-ip]
    +--ro vni-id                    uint32
    +--ro source-ip                 inet:ip-address-no-zone
    +--ro peer-ip                   inet:ip-address-no-zone
    +--ro type?                     peerType
    +--ro out-vni-id?               uint32
+--rw vni-infos
  +--rw vni_info* [vni-id]
    +--rw vni-id                    uint32
    +--rw statistics-enable?        vniStatisticsEnable
    +--ro status?                   vniStatus
+--ro tunnel-infos
  +--ro tunnel-infos* [tunnel-id]
    +--ro tunnel-id                uint32
    +--ro source-ip?               inet:ip-address-no-zone
    +--ro peer-ip?                 inet:ip-address-no-zone
    +--ro status?                  tunnelStatus
    +--ro type?                    tunnelType
    +--ro up-time?                 string
    +--ro vrf-name?                vrfName

```

Figure 3.1. The tree structure of YANG module for NVO3 configuration

3.4. YANG Module

<CODE BEGINS> file "ietf-nvo3-base@2019-03-01.yang"

```
module ietf-nvo3-base {
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3-base";
  //namespace need to be assigned by IANA
  prefix "nvo3";
  import ietf-inet-types {
    prefix "inet";
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization "IETF NVO3 Working Group";
  contact "zhangmingui@huawei.com
    sean.gao@huawei.com";
  description "nvo3 yang module";
  revision "2019-03-01" {
    description
      "Initial version";
    reference "RFC 7348 RFC 7637";
  }
  typedef vrfName {
    type string {
      length "1..31";
    }
    description
      "vrfName is a string type with length constraints";
  }
  typedef ifName {
    type string {
      length "1..63";
    }
    description
      "ifName is a string type with length constraints";
  }
  typedef vniStatus {
    type enumeration {
      enum up {
        description
          "The VNI is up";
      }
      enum down {
        description
          "The VNI is down";
      }
    }
    description
      "The status of a VNI can be either up or down";
  }
```

```
}
typedef protocolType {
  type enumeration {
    enum null {
      description
        "No specific protocol is used";
    }
    enum evpn {
      description
        "EVPN is used as the control plane protocol";
    }
  }
  description
    "The protocol type being used as control plane";
}
typedef peerType {
  type enumeration {
    enum static {
      description
        "Static peer NVE";
    }
    enum dynamic {
      description
        "Dynamic peer NVE";
    }
  }
  description
    "The type of the peer NVE";
}
typedef tunnelStatus {
  type enumeration {
    enum up {
      description
        "The tunnel is up";
    }
    enum down {
      description
        "The tunnel is down";
    }
  }
  description
    "The status of a tunnel can be either up or down";
}
typedef vniStatisticsEnable {
  type enumeration {
    enum enable {
      description
        "The statistics is enabled";
    }
  }
}
```

```
    }
    enum disable {
      description
        "The statistics is disabled";
    }
  }
  description
    "The statistics of a VNI can be enabled or disabled";
}
typedef tunnelType {
  type enumeration {
    enum dynamic {
      description
        "The tunnel is built in a dynamic way";
    }
    enum static {
      description
        "The tunnel is built by static configuration";
    }
  }
}
description
  "The type of a tunnel";
}
container nvo3{
  description
    "NVO3 base YANG main part";
  container common {
    description
      "Common part";
    leaf nvo3-enable {
      type boolean;
      default "false";
      description
        "Enable/Disable NVO3 featrues";
    }
  }
}
container nves {
  description
    "Parameters of NVEs";
  list nve {
    key "if-name source-interface";
    description
      "The list of NVEs";
    leaf if-name {
      type ifName;
      description
        "nve interface name";
    }
  }
}
```



```
    leaf vtep-ip {
      type inet:ipv4-address-no-zone;
      description
        "NVO3 tunnel source address";
    }
    leaf ipv6-vtep-ip {
      type inet:ipv6-address-no-zone;
      description
        "ipv6 NVO3 tunnel source address";
    }
    leaf source-interface {
      type ifName;
      description
        "source interface";
    }
    leaf mac-address {
      type yang:mac-address;
      description
        "mac address";
    }
    leaf bypass-vtep-ip {
      type inet:ipv4-address-no-zone;
      description
        "bypass NVO3 tunnel source address";
    }
    container vni-members {
      description
        "The VNI members on the NVE";
      list vni-member {
        key "vni-id";
        description
          "nve vni member";
        leaf vni-id {
          type uint32 {
            range "1..16777215";
          }
          description
            "Associate NVO3 VNIs (Virtual Network Identifiers) with the N
VE interface.";
        }
        leaf protocol-bgp {
          type protocolType;
          default "null";
          description
            "Enables BGP EVPN with ingress replication for the VNI.";
        }
        container peers {
          description
            "The peers in this VNI";
        }
      }
    }
  }
}
```

```
list peer {
  key "peer-ip";
  description
    "A peer NVE in this VNI";
  leaf peer-ip {
    type inet:ipv4-address-no-zone;
    description
      "peer ip address";
  }
  leaf out-vni-id {
    type uint32 {
      range "1..16777215";
    }
    description
      "out vni id";
  }
  leaf split-horizon-group {
    type string {
      length "1..31";
    }
    description
      "split group name";
  }
}
}
container ipv6-peers {
  description
    "The IPv6 peers in this VNI";
  list ipv6-peer {
    key "ipv6-peer-ip";
    description
      "An IPv6 peer NVE in this VNI";
    leaf ipv6-peer-ip {
      type inet:ipv6-address-no-zone;
      description
        "peer ipv6 address";
    }
  }
}
container flood-proxys {
  description
    "The flood proxys for this VNI";
  list flood-proxy {
    key "peer-ip";
    leaf peer-ip {
      type inet:ipv4-address-no-zone;
      description
        "peer ip address";
    }
  }
}
```

```
        }
        description
            "List of the flood proxys";
    }
}
list mcast-group {
    key "mcast-group";
    description
        "The multicast groups in this VNI";
    leaf mcast-group {
        type inet:ipv4-address-no-zone;
        description
            "mcast ip address";
    }
}
}
}
container vni-map-l2vpns {
description
    "Mapping VNIs to L2VPNs";
list vni-map-l2vpn {
    key "vni-id";
    description
        "L2VPN";
    leaf l2vpn-id {
        type uint32 {
            range "1..16777215";
        }
        mandatory true;
        description
            "l2vpn id";
    }
    leaf l2vpn-name {
        type string {
            length "1..31";
        }
        description
            "l2vpn name";
    }
}
leaf vni-id {
    type uint32 {
        range "1..16777215";
    }
    description
        "vni id";
}
leaf split-horizon-group {
    type string {
```

```
        length "1..31";
      }
      description
        "split group name";
    }
  }
}
container vni-map-l3vpns {
  description
    "Mapping VNIs to L3VPNs";
  list vni-map-l3vpn {
    key "l3vpn-name";
    description
      "L3VPN";
    leaf l3vpn-name {
      type vrfName;
      description
        "l3vpn name";
    }
    leaf vni-id {
      type uint32 {
        range "1..16777215";
      }
      description
        "vni id";
    }
  }
}
container vni-statistics-infos {
  config false;
  description
    "The statistics information for VNIs";
  list vni-statistic-info {
    key "vni-id";
    config false;
    description
      "The statistics information for a VNI";
    leaf vni-id {
      type uint32 {
        range "1..16777215";
      }
      config false;
      description
        "vni id";
    }
    container vni-statistics {
      config false;
      description
```

```
    "The statistics information items for a VNI";
leaf rx-bits-persec {
    type uint64;
    config false;
    description
        "Received bits per second";
}
leaf rx-pkts-persec {
    type uint64;
    config false;
    description
        "Received packets per second";
}
leaf tx-bits-persec {
    type uint64;
    config false;
    description
        "Transmitted bits per second";
}
leaf tx-pkts-persec {
    type uint64;
    config false;
    description
        "Transmitted packets per second";
}
leaf rx-pkts {
    type uint64;
    config false;
    description
        "Received packets";
}
leaf rx-bytes {
    type uint64;
    config false;
    description
        "Received bytes";
}
leaf tx-pkts {
    type uint64;
    config false;
    description
        "Transmitted packets";
}
leaf tx-bytes {
    type uint64;
    config false;
    description
        "Transmitted bytes";
}
```

```
}
leaf rx-unicast-pkts {
    type uint64;
    config false;
    description
        "Received unicast packets";
}
leaf rx-multicast-pkts {
    type uint64;
    config false;
    description
        "Received multicast packets";
}
leaf rx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Received broadcast packets";
}
leaf drop-unicast-pkts {
    type uint64;
    config false;
    description
        "Dropped unicast packets";
}
leaf drop-multicast-pkts {
    type uint64;
    config false;
    description
        "Dropped multicast packets";
}
leaf drop-broadcast-pkts {
    type uint64;
    config false;
    description
        "Dropped broadcast packets";
}
leaf tx-unicast-pkts {
    type uint64;
    config false;
    description
        "Transmitted unicast packets";
}
leaf tx-multicast-pkts {
    type uint64;
    config false;
    description
        "Transmitted multicast packets";
}
```

```
    }
    leaf tx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Transmitted broadcast packets";
    }
}
}
container vnipeer-statistics-cfgs {
    description
        "Statistics configuration of peers in the VNI.";
    list vnipeer-satistics-cfg {
        key "vni-id peer-ip";
        description
            "Statistics configuration of a peer in the VNI.";
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "vni id";
        }
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "The Ipv4 address of the peer";
        }
    }
}
container vnipeer-statistics-infos {
    config false;
    description
        "Statistics configuration of the peers in a VNI";
    list vnipeer-satistics-info {
        key "vni-id source-ip peer-ip";
        config false;
        description
            "Statistics information of a peer in the VNI.";
        leaf vni-id {
            type uint32 {
                range "0..16777215";
            }
            config false;
            description
                "vni id";
        }
    }
}
```

```
leaf source-ip {
  type inet:ipv4-address-no-zone;
  config false;
  description
    "Source address";
}
leaf peer-ip {
  type inet:ipv4-address-no-zone;
  config false;
  description
    "Address of the peer";
}
container vnipeer_statistics {
  config false;
  description
    "Statistics information items of a peer in the
e VNI";

  leaf rx-bits-persec {
    type uint64;
    config false;
    description
      "Received bits per second";
  }
  leaf rx-pkts-persec {
    type uint64;
    config false;
    description
      "Received packets per second";
  }
  leaf tx-bits-persec {
    type uint64;
    config false;
    description
      "Transmitted bits per second";
  }
  leaf tx-pkts-persec {
    type uint64;
    config false;
    description
      "Transmitted packets per second";
  }
  leaf rx-pkts {
    type uint64;
    config false;
    description
      "Received packets";
  }
  leaf rx-bytes {
    type uint64;
```



```
        config false;
        description
            "Received bytes";
    }
    leaf tx-pkts {
        type uint64;
        config false;
        description
            "Transmitted packets";
    }
    leaf tx-bytes {
        type uint64;
        config false;
        description
            "Transmitted bytes";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Received unicast packets";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Received multicast packets";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Received broadcast packets";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Dropped unicast packets";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Dropped multicast packets";
    }
    leaf drop-broadcast-pkts {
        type uint64;
```

```

        config false;
                description
                "Dropped broadcast packets";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
                description
                "Transmitted unicast packets";
    }
    leaf tx-multicast-pkts {
        type uint64;
        config false;
                description
                "Transmitted multicast packets";
    }
    leaf tx-broadcast-pkts {
        type uint64;
        config false;
                description
                "Transmitted broadcast packets";
    }
}

}

}

container ipv6-vnipeer-statistics-cfgs {
    description
    "Statistics configuration of IPv6 peers in the VNI.";
    list ipv6-vnipeer-statistics-cfg {
        key "vni-id ipv6-source-ip ipv6-peer-ip";
            description
            "Statistics configuration of an IPv6 peer in the VNI.";
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
            "vni id";
        }
        leaf ipv6-source-ip {
            type inet:ipv6-address-no-zone;
                description
                "Source IPv6 address";
        }
        leaf ipv6-peer-ip {
            type inet:ipv6-address-no-zone;
                description
                "IPv6 address of the peer";
        }
    }
}

```

```
    }
  }
}
container ipv6-vnipeer-statistics-infos {
  config false;
  description
    "Statistics information of IPv6 peers in the VNI.";
  list ipv6-vnipeer-statistics-info {
    key "vniId ipv6-source-ip ipv6-peer-ip";
    config false;
    description
      "Statistics information of an IPv6 peer in the VNI.";
    leaf vniId {
      type uint32 {
        range "1..16777215";
      }
      config false;
      description
        "vni id";
    }
    leaf ipv6-source-ip {
      type inet:ipv6-address-no-zone;
      config false;
      description
        "Source IPv6 address";
    }
    leaf ipv6-peer-ip {
      type inet:ipv6-address-no-zone;
      config false;
      description
        "IPv6 address of the peer";
    }
  }
  container ipv6_vnipeer_statistics {
    config false;
    description
      "Statistics information items of an IPv6 peer in the VNI.";
    leaf rx-bits-persec {
      type uint64;
      config false;
      description
        "Received bits per second";
    }
    leaf rx-pkts-persec {
      type uint64;
      config false;
      description
        "Received packets per second";
    }
  }
}
```

```
leaf tx-bits-persec {
    type uint64;
    config false;
    description
        "Transmitted bits per second";
}
leaf tx-pkts-persec {
    type uint64;
    config false;
    description
        "Transmitted packets per second";
}
leaf rx-pkts {
    type uint64;
    config false;
    description
        "Received packets";
}
leaf rx-bytes {
    type uint64;
    config false;
    description
        "Received bytes";
}
leaf tx-pkts {
    type uint64;
    config false;
    description
        "Transmitted packets";
}
leaf tx-bytes {
    type uint64;
    config false;
    description
        "Transmitted bytes";
}
leaf rx-unicast-pkts {
    type uint64;
    config false;
    description
        "Received unicast packets";
}
leaf rx-multicast-pkts {
    type uint64;
    config false;
    description
        "Received multicast packets";
}
```

```
leaf rx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Received broadcast packets";
}
leaf drop-unicast-pkts {
    type uint64;
    config false;
    description
        "Dropped unicast packets";
}
leaf drop-multicast-pkts {
    type uint64;
    config false;
    description
        "Dropped multicast packets";
}
leaf drop-broadcast-pkts {
    type uint64;
    config false;
    description
        "Dropped broadcast packets";
}
leaf tx-unicast-pkts {
    type uint64;
    config false;
    description
        "Transmitted unicast packets";
}
leaf tx-multicast-pkts {
    type uint64;
    config false;
    description
        "Transmitted multicast packets";
}
leaf tx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Transmitted broadcast packets";
}
}
}
}
}
}
container vnipeer-infos {
```

```
    config false;
    description
      "Information of the peers";
  list vnipeer-info {
    key "vni-id source-ip peer-ip";
    config false;
    description
      "Information of a peer";
    leaf vni-id {
      type uint32 {
        range "1..16777215";
      }
      config false;
      description
        "vni-id";
    }
    leaf source-ip {
      type inet:ip-address-no-zone;
      config false;
      description
        "source ip";
    }
    leaf peer-ip {
      type inet:ip-address-no-zone;
      config false;
      description
        "peer ip";
    }
    leaf type {
      type peerType;
      config false;
      description
        "peer type";
    }
    leaf out-vni-id {
      type uint32 {
        range "1..16777215";
      }
      config false;
      description
        "out vni id";
    }
  }
}
container vni-infos {
  description
    "Information of the VNIs";
  list vni_info {
```

```
    key "vni-id";
      description
        "Information of a VNI";
    leaf vni-id {
      type uint32 {
        range "1..16777215";
      }
      description
        "vni id";
    }
    leaf statistics-enable {
      type vniStatisticsEnable;
      default "disable";
      description
        "Statistics is enabled or not";
    }
    leaf status {
      type vniStatus;
      config false;
      description
        "The status of the VNI";
    }
  }
}
container tunnel-infos {
  config false;
  description
    "Information of tunnels";
  list tunnel-infos {
    key "tunnel-id";
    config false;
    description
      "Information of a tunnel";
    leaf tunnel-id {
      type uint32 {
        range "1..4294967295";
      }
      config false;
      description
        "tunnel id";
    }
    leaf source-ip {
      type inet:ip-address-no-zone;
      config false;
      description
        "source";
    }
    leaf peer-ip {
```

```

        type inet:ip-address-no-zone;
        config false;
        description
            "peer ip";
    }
    leaf status {
        type tunnelStatus;
        config false;
        description
            "tunnel status";
    }
    leaf type {
        type tunnelType;
        config false;
        description
            "tunnel type";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "tunnel up time";
    }
    leaf vrf-name {
        type vrfName;
        default "_public_";
        config false;
        description
            "vrf";
    }
}
}
}
}
<CODE ENDS>
```

4. Security Considerations

This document raises no new security issues.

5. IANA Considerations

The namespace URI defined in Section 3.3 need be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in

the YANG Module Names registry [RFC6020].

Acknowledgements

Authors would like to thank the comments and suggestions from Tao Han, Weilian Jiang.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7364] T. Narten, E. Gray, et al, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement, working in progress.
- [RFC7365] Marc Lasserre, Florin Balus, et al, "Framework for DC Network Virtualization", draft-ietf-nvo3-framework, working in progress.
- [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.
- [I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-10 (work in progress), March 2019.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

6.2. Informative References

- [RFC7637] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC7637, September 2015.
- [I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur,

- "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten, An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3), RFC8014, December 2016.
- [I-D.ietf-nvo3-encap-02] Boutros, S., "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-02 (work in progress), September, 2018.

Author's Addresses

Mingui Zhang
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China

EMail: zhangmingui@huawei.com

Yuan Gao
Huawei Technologies
No. 101 Nanjing Rd. Yuhua District,
Nanjing 210012
P.R. China
EMail: sean.gao@huawei.com

Haibo Wang
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China
EMail: rainsword.wang@huawei.com

Bing Liu
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China
EMail: remy.liubing@huawei.com

Fu Qiao
China Mobile

EMail: fuqiao@chinamobile.com

INTERNET-DRAFT
Intended Status: Standards Track

B. Liu, Ed.
Huawei
R. Chen
ZTE
F. Qin
China Mobile
R. Rahman
Cisco

Expires: March 9, 2020

September 6, 2019

Base YANG Data Model for NVO3 Protocols
draft-zhang-nvo3-yang-cfg-07.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Acronyms and Terminology	3
2.1. Acronyms	3
2.2. Terminology	3
3. The YANG Data Model for NVO3	3
3.1 Mapping to the NVO3 architecture	4
3.2. The Configuration Parameters	4
3.2.1. NVE as an interface	4
3.2.2. Virtual Network Instance	5
3.2.3. BUM Mode	5
3.3. Statistics	5
3.3. Model Structure	5
3.4. YANG Module	8
4. Security Considerations	24
5. IANA Considerations	24
6. Contributors	24
7. Acknowledgements	25
8. References	25
8.1. Normative References	25
8.2. Informative References	26
Author's Addresses	27

1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN, NVGRE, GENEVE and VXLAN-GPE, enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE may require configuration of TLVs at the NVE. The base module can be augmented to accommodate these new solutions.

2. Acronyms and Terminology

2.1. Acronyms

NVO3: Network Virtualization Overlays
VNI: Virtual Network Instance
BUM: Broadcast, Unknown Unicast, Multicast traffic

2.2. Terminology

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 RFC 2119 [RFC2119].

Familiarity with [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

3. The YANG Data Model for NVO3

The NVO3 base YANG model defined in this document is used to configure the NVEs. It is divided into three containers. The first container contains the configuration of the virtual network instances, e.g. the VNI, the NVE that the instance is mounted, the peer NVEs which can be determined dynamically via a control plane or given statically, and the statistical states of the instance. The other two containers are separately the statistical states of the

peer NVEs and the tunnels.

3.1 Mapping to the NVO3 architecture

The NVO3 base YANG model is defined according to the NVO3 architecture [RFC8014]. As shown in Figure 3.1, the reference model of the NVE defined in [RFC8014], multiple instances can be mounted under a NVE. The key of the instance is VNI. The source NVE of the instance is the NVE configured by the base YANG. An instance can have several peer NVEs. A NVO3 tunnel can be determined by the VNI, the source NVE and the peer NVE. The tunnel can be built statically by manually indicate the addresses of the peer NVEs, or dynamically via a control plane, e.g. EVPN [RFC8365]. An enabler is defined in the NVO3 base YANG to choose from these two modes.

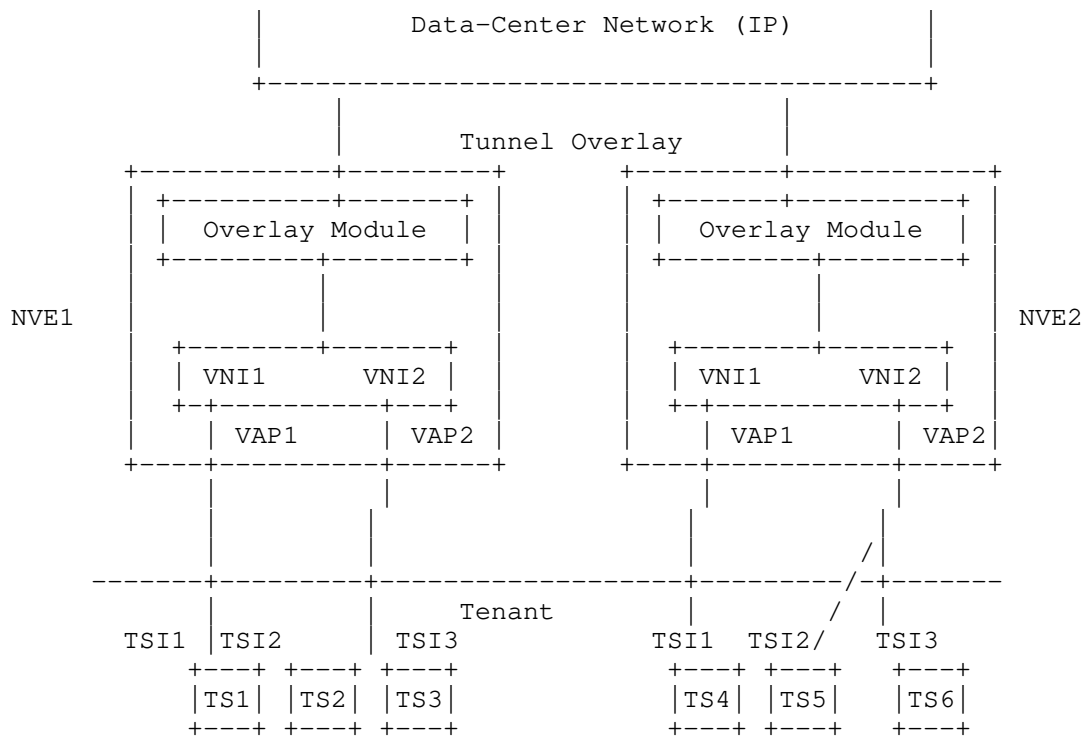


Figure 3.1. NVE Reference model in RFC 8014

3.2. The Configuration Parameters

3.2.1. NVE as an interface

A NVE in the NVO3 base YANG is defined via augmenting the IETF

3.2.2. Virtual Network Instance

As defined in [draft-ietf-bess-evpn-inter-subnet-forwarding], a tenant can have multiple bridge domains, and each domain has its own VNI. Thus these VNIs are used as L2VPN. Besides, a dedicated VNI can be used for routing between the bridge domains, i.e. used as L3VPN. The mapping relationship between VNI and L2VPN (respectively, L3VPN) is given by augmenting the IETF YANG of L2VPN (respectively L3VPN).

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

3.3. Statistics

3.3. Model Structure

Bing Liu, et al


```

+--rw source-nve                    if:interface-ref
+--rw protocol-bgp?                 boolean
+--ro status?                       vni-status-type
+--rw static-ipv4-peers
|   +--rw static-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
|   |   +--rw out-vni-id?            uint32
+--rw static-ipv6-peers
|   +--rw static-ipv6-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv6-address-no-zone
+--rw flood-proxys
|   +--rw flood-proxy* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
+--rw mcast-groups
|   +--rw mcast-group* [mcast-ip]
|   |   +--rw mcast-ip               inet:ipv4-address-no-zone
+--rw statistic
|   +--rw statistic-enable?          boolean
|   +--ro statistic-info
|   |   +--ro rx-bits-per-sec?        uint64
|   |   +--ro rx-pkt-per-sec?         uint64
|   |   +--ro tx-bits-per-sec?        uint64
|   |   +--ro tx-pkt-per-sec?         uint64
|   |   +--ro rx-pkts?                uint64
|   |   +--ro rx-bytes?               uint64
|   |   +--ro tx-pkts?                uint64
|   |   +--ro tx-bytes?               uint64
|   |   +--ro rx-unicast-pkts?        uint64
|   |   +--ro rx-multicast-pkts?      uint64
|   |   +--ro rx-broadcast-pkts?      uint64
|   |   +--ro drop-unicast-pkts?      uint64
|   |   +--ro drop-multicast-pkts?    uint64
|   |   +--ro drop-broadcast-pkts?    uint64
|   |   +--ro tx-unicast-pkts?        uint64
|   |   +--ro tx-multicast-pkts?      uint64
|   |   +--ro tx-broadcast-pkts?      uint64
+--ro vni-peer-infos
|   +--ro peers
|   |   +--ro peer* [vni-id source-ip peer-ip]
|   |   |   +--ro vni-id              uint32
|   |   |   +--ro source-ip           inet:ip-address-no-zone
|   |   |   +--ro peer-ip             inet:ip-address-no-zone
|   |   |   +--ro tunnel-type?        peer-type
|   |   |   +--ro out-vni-id?         uint32
+--ro tunnel-infos
|   +--ro tunnel-info* [tunnel-id]
|   |   +--ro tunnel-id               uint32
|   |   +--ro source-ip?              inet:ip-address-no-zone

```

```

    +--ro peer-ip?          inet:ip-address-no-zone
    +--ro status?           tunnel-status
    +--ro type?             tunnel-type
    +--ro up-time?          string
    +--ro vrf-name?         -> /ni:network-instances/network-instance/name

augment /if:interfaces/if:interface:
  +--rw nvo3-nve
    +--rw nvo3-config
      +--rw source-vtep-ip?    inet:ipv4-address-no-zone
      +--rw source-vtep-ipv6?  inet:ipv6-address-no-zone
      +--rw bypass-vtep-ip?    inet:ipv4-address-no-zone
      +--rw statistics
        +--rw statistic* [vni-id mode peer-ip direction]
          +--rw vni-id        uint32
          +--rw mode          vni-type
          +--rw peer-ip       inet:ipv4-address-no-zone
          +--rw direction     direction-type
          +--ro info
            +--ro rx-pkts?      uint64
            +--ro rx-bytes?     uint64
            +--ro tx-pkts?      uint64
            +--ro tx-bytes?     uint64
            +--ro rx-unicast-pkts? uint64
            +--ro rx-multicast-pkts? uint64
            +--ro rx-broadcast-pkts? uint64
            +--ro tx-unicast-pkts? uint64
            +--ro tx-multicast-pkts? uint64
            +--ro tx-broadcast-pkts? uint64
            +--ro drop-unicast-pkts? uint64
            +--ro drop-multicast-pkts? uint64
            +--ro drop-broadcast-pkts? uint64
            +--ro rx-bits-per-sec? uint64
            +--ro rx-pkt-per-sec? uint64
            +--ro tx-bits-per-sec? uint64
            +--ro tx-pkt-per-sec? uint64
      +--rw nvo3-gateway
        +--rw nvo3-gateway
          +--rw vxlan-anycast-gateway? boolean
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l3vpn:l3vpn/l3
    vpn:l3vpn:
      +--rw vni-lists
        +--rw vni* [vni-id]
          +--rw vni-id      uint32
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l2vpn:l2vpn:
    +--rw vni-lists
      +--rw vni* [vni-id]
        +--rw vni-id      uint32
        +--rw split-horizon-mode? vni-bind-type

```

```

        +---rw split-group?          string

    rpcs:
      +---x reset-vni-instance-statistic
      |   +---w input
      |       +---w vni-id          uint32
      +---x reset-vni-peer-statistic
      |   +---w input
      |       +---w vni-id          uint32
      |       +---w mode            vni-type
      |       +---w peer-ip         inet:ipv4-address-no-zone
      |       +---w direction       direction-type

```

Figure 3.2. The tree structure of YANG module for NVO3 configuration

3.4. YANG Module

```

<CODE BEGINS> file "ietf-nvo3-base@2019-07-01.yang"
module ietf-nvo3 {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3";
  prefix "nvo3";

  import ietf-network-instance {
    prefix "ni";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-bgp-l3vpn {
    prefix "l3vpn";
  }

  organization "ietf";
  contact "ietf";
  description "Yang model for NVO3";

  revision 2019-04-01 {

```

```
    description
      "Init revision";
    reference
      "";
  }

  typedef vni-status-type {
    type enumeration {
      enum "up" {
        description
          "Vni status up.";
      }
      enum "down" {
        description
          "Vni status down.";
      }
    }
    description
      "Vni status";
  }

  typedef vni-type {
    type enumeration {
      enum "l2" {
        description
          "layer 2 mode";
      }
      enum "l3" {
        description
          "layer 3 mode";
      }
    }
    description
      "vni type";
  }

  typedef peer-type {
    type enumeration {
      enum "static" {
        description
          "Static.";
      }
      enum "dynamic" {
        description
          "Dynamic.";
      }
    }
    description
```

```
        "Peer type";
    }

typedef tunnel-status {
    type enumeration {
        enum "up" {
            description
                "The tunnel is up.";
        }
        enum "down" {
            description
                "The tunnel is down.";
        }
    }
    description
        "Tunnel status";
}

typedef tunnel-type {
    type enumeration {
        enum "dynamic" {
            description
                "The tunnel is dynamic.";
        }
        enum "static" {
            description
                "The tunnel is static.";
        }
        enum "invalid" {
            description
                "The tunnel is invalid.";
        }
    }
    description
        "Tunnel type";
}

typedef direction-type {
    type enumeration {
        enum "inbound" {
            description
                "Inbound.";
        }
        enum "outbound" {
            description
                "Outbound.";
        }
        enum "bidirection" {
            description
```

```
        "Bidirection.";
    }
}
description
    "Bound direction";
}

typedef vni-bind-type {
    type enumeration {
        enum "hub-mode" {
            description
                "Hub mode.";
        }
        enum "spoke-mode" {
            description
                "Spoke mode.";
        }
    }
    description
        "bdBindVniType";
}

container nvo3 {
    description
        "Management of NVO3.";

    container vni-instances {
        description
            "The confiuration and information table of the VNI.";
        list vni-instance {
            key "vni-id";
            must "(/if:interfaces/if:interface[if:name = current()/source_nve]/if
:type = 'Nve')";
            description
                "The confiuration and information of the VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "The id of VNI.";
            }
            leaf vni-mode {
                type enumeration {
                    enum "Local" {
                        description
                            "Local mode";
                    }
                    enum "Global" {
```

```
        description
            "Global mode";
        }
    }
    description
        "The mode of the VNI instance.";
}
leaf source-nve {
    type if:interface-ref;
    mandatory true;
    description
        "The name of the nve interface .";
}
leaf protocol-bgp {
    type boolean;
    default "false";
    description
        "Whether use bgp as vxlan's protocol.";
}
leaf status {
    type vni-status-type;
    config false;
    description
        "The status of the VNI.";
}
container static-ipv4-peers {
    description
        "The remote NVE address table in a same VNI.";
    list static-peer {
        key "peer-ip";
        description
            "The remote NVE address in a same VNI.";
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "The address of the NVE.";
        }
        leaf out-vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "The ID of the out VNI. Do not support separate deletion.";
        }
    }
}
container static-ipv6-peers {
    description
```

```
    "The remote NVE ipv6 address table in a same VNI.";
  list static-ipv6-peer {
    key "peer-ip";
    description
      "The remote NVE ipv6 address in a same VNI.";
    leaf peer-ip {
      type inet:ipv6-address-no-zone;
      description
        "The ipv6 address of the NVE.";
    }
  }
}
container flood-proxys {
  description
    "The flood proxys for this VNI";
  list flood-proxy {
    key "peer-ip";
    leaf peer-ip {
      type inet:ipv4-address-no-zone;
      description
        "peer ip address";
    }
    description
      "List of the flood proxys";
  }
}
container mcast-groups {
  description
    "The mcast address table.";
  list mcast-group {
    key "mcast-ip";
    description
      "The mcast address.";
    leaf mcast-ip {
      type inet:ipv4-address-no-zone;
      description
        "The mcast address of NVO3.";
    }
  }
}
container statistic {
  description
    "The VNI member in a same NVE.";
  leaf statistic-enable {
    type boolean;
    default "false";
    description
      "To determine whether to enable the statistics for a VNI.";
```



```
}
container statistic-info {
  config false;
  description
    "The vni instance traffic statistics information.";
  leaf rx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits received per second.";
  }
  leaf rx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets received per second.";
  }
  leaf tx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits sent per second.";
  }
  leaf tx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets sent per second.";
  }
  leaf rx-pkts {
    type uint64;
    config false;
    description
      "Total number of received packets.";
  }
  leaf rx-bytes {
    type uint64;
    config false;
    description
      "Total number of received bytes.";
  }
  leaf tx-pkts {
    type uint64;
    config false;
    description
      "Total number of sent packets.";
  }
  leaf tx-bytes {
```

```
        type uint64;
        config false;
        description
            "Total number of sent bytes.";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of received unicast packets.";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of received multicast packets.";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of received broadcast packets.";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded unicast packets.";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded multicast packets.";
    }
    leaf drop-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded broadcast packets.";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent unicast packets.";
    }
    leaf tx-multicast-pkts {
```

```
        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
}
}
}
}
container vni-peer-infos {
    config false;
    description
        "The information table of vni members.";
    container peers {
        config false;
        description
            "The remote nve address in a same VNI.";
        list peer {
            key "vni-id source-ip peer-ip";
            config false;
            description
                "The remote nve address list in a same VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                config false;
                description
                    "The ID of VNI.";
            }
            leaf source-ip {
                type inet:ip-address-no-zone;
                config false;
                description
                    "The source address of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ip-address-no-zone;
                config false;
                description
```

```
        "The remote NVE address.";
    }
    leaf tunnel-type {
        type peer-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        config false;
        description
            "The ID of the out VNI.";
    }
}
}
}

container tunnel-infos {
    config false;
    description
        "VxLAN tunnel information.";
    list tunnel-info {
        key "tunnel-id";
        config false;
        description
            "VxLAN tunnel information list.";
        leaf tunnel-id {
            type uint32 {
                range "1..4294967295";
            }
            config false;
            description
                "The ID of Vxlan tunnel.";
        }
        leaf source-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Local NVE interface address.";
        }
        leaf peer-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Remote NVE interface address.";
```

```
    }
    leaf status {
        type tunnel-status;
        config false;
        description
            "Tunnel status.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "Vxlan tunnel up time.";
    }
    leaf vrf-name {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        default "_public_";
        config false;
        description
            "The name of VPN instance.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Augment the interface, NVE as an interface.";
    container nvo3-nve {
        when "if:interfaces/if:interface/if:type = 'Nve'";
        description
            "Network virtualization edge.";
        leaf source-vtep-ip {
            type inet:ipv4-address-no-zone;
            description
                "The source address of the NVE interface.";
        }
        leaf source-vtep-ipv6 {
            type inet:ipv6-address-no-zone;
            description

```

```
        "The source ipv6 address of the NVE interface.";
    }
    leaf bypass-vtep-ip {
        type inet:ipv4-address-no-zone;
        description
            "The source address of bypass VXLAN tunnel.";
    }
    container statistics {
        description
            "VXLAN Tunnel Traffic Statistical Configuration Table.";
        list statistic {
            key "vni-id mode peer-ip direction";
            description
                "VXLAN Tunnel Traffic Statistics Configuration.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "ID of the VNI.";
            }
            leaf mode {
                type vni-type;
                description
                    "The type of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ipv4-address-no-zone;
                description
                    "IP address of the remote VTEP.";
            }
            leaf direction {
                type direction-type;
                must "(./mode='l3' and ./bound!='bidirection')";
                description
                    "Traffic statistics type about the VXLAN tunnel.";
            }
        }
        container info {
            config false;
            description
                "Traffic statistics about the peer.";
            leaf rx-pkts {
                type uint64;
                config false;
                description
                    "Total number of received packets.";
            }
            leaf rx-bytes {
```

```
        type uint64;
        config false;
        description
            "Total number of received bytes.";
    }
    leaf tx-pkts {
        type uint64;
        config false;
        description
            "Total number of sent packets.";
    }
    leaf tx-bytes {
        type uint64;
        config false;
        description
            "Total number of sent bytes.";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of received unicast packets.";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of received multicast packets.";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of received broadcast packets.";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent unicast packets.";
    }
    leaf tx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
```

```
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded unicast packets.";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded multicast packets.";
    }
    leaf drop-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded broadcast packets.";
    }
    leaf rx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits received per second.";
    }
    leaf rx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets received per second.";
    }
    leaf tx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits sent per second.";
    }
    leaf tx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets sent per second.";
    }
}
```



```
    }
  }

}

container nvo3-gateway {
  when "if:interfaces/if:interface/if:type = 'Vbdif'";
  description
    "Enable anycast gateway.";
  leaf vxlan-anycast-gateway {
    type boolean;
    default "false";
    description
      "Enable vxlan anycast gateway.";
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l3vpn:l3vpn/l3vpn:l3vpn" {
  description "Augment for l3vpn instance";
  container vni-lists {
    description "Vni list for l3vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l3vpn instance";
      leaf vni-id {
        type uint32 {
          range "1..16777215";
        }
        description
          "The id of VNI.";
      }
    }
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l2vpn:l2vpn" {
  description "Augment for l2vpn instance";
  container vni-lists {
    description "Vni list for l2vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l2vpn instance";
      leaf vni-id {
        type uint32 {
```

```
        range "1..16777215";
    }
    description
        "The id of VNI.";
}
leaf split-horizon-mode {
    type vni-bind-type;
    default "hub-mode";
    description
        "Split horizon mode.";
}
leaf split-group {
    type string {
        length "1..31";
    }
    description
        "Split group name.";
}
}
}
}

rpc reset-vni-instance-statistic {
    description
        "Clear traffic statistics about the VNI.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}

rpc reset-vni-peer-statistic {
    description
        "Clear traffic statistics about the VXLAN tunnel.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}
```

```
    leaf mode {
        type vni-type;
        mandatory true;
        description
            "The type of vni memeber statistic.";
    }
    leaf peer-ip {
        type inet:ipv4-address-no-zone;
        mandatory true;
        description
            "IP address of the remote NVE interface.";
    }
    leaf direction{
        type direction-type;
        must "(./mode='mode-l3' and ./bound!='bidirection')";
        mandatory true;
        description
            "Traffic statistics type about the VXLAN tunnel.";
    }
}
}
```

<CODE ENDS>

4. Security Considerations

This document raises no new security issues.

5. IANA Considerations

The namespace URI defined in Section 3.3 need be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in the YANG Module Names registry [RFC6020].

6. Contributors

Haibo Wang
Huawei
Email: rainsword.wang@huawei.com

Yuan Gao
Huawei
Email: sean.gao@huawei.com

Gang Yan

Huawei
Email: yangang@huawei.com

Mingui Zhang
Huawei
Email: zhangmingui@huawei.com

Yubao (Bob) Wang
ZTE Corporation
Email: yubao.wang2008@hotmail.com

Ruixue Wang
China Mobile
Email: wangruixue@chinamobile.com

Sijun Weng
China Mobile
Email: wengsijun@chinamobile.com

7. Acknowledgements

Authors would like to thank the comments and suggestions from Tao Han, Weilian Jiang.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7364] T. Narten, E. Gray, et al, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement, working in progress.
- [RFC7365] Marc Lasserre, Florin Balus, et al, "Framework for DC Network Virtualization", draft-ietf-nvo3-framework, working in progress.
- [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.
- [I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-

nvo3-geneve-10 (work in progress), March 2019.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten, An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3), RFC8014, December 2016.

8.2. Informative References

- [RFC7637] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC7637, September 2015.
- [I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [I-D.draft-ietf-bess-evpn-inter-subnet-forwarding] A. Sajassi, S. Salam, S. Thoria, J. Drake, J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-08, March 4, 2019.
- [RFC8293] A. Ghanwani, L. Dunbar, V. Bannai, M. McBride, R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC8293, January 2018.

Author's Addresses

Bing Liu
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China

Email: remy.liubing@huawei.com

Ran Chen
ZTE Corporation

Email: chen.ran@zte.com.cn

Fengwei Qin
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing, Beijing 100053
China

Email: qinfengwei@chinamobile.com

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com