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/1id-abstracts.html

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

Table of Contents

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.

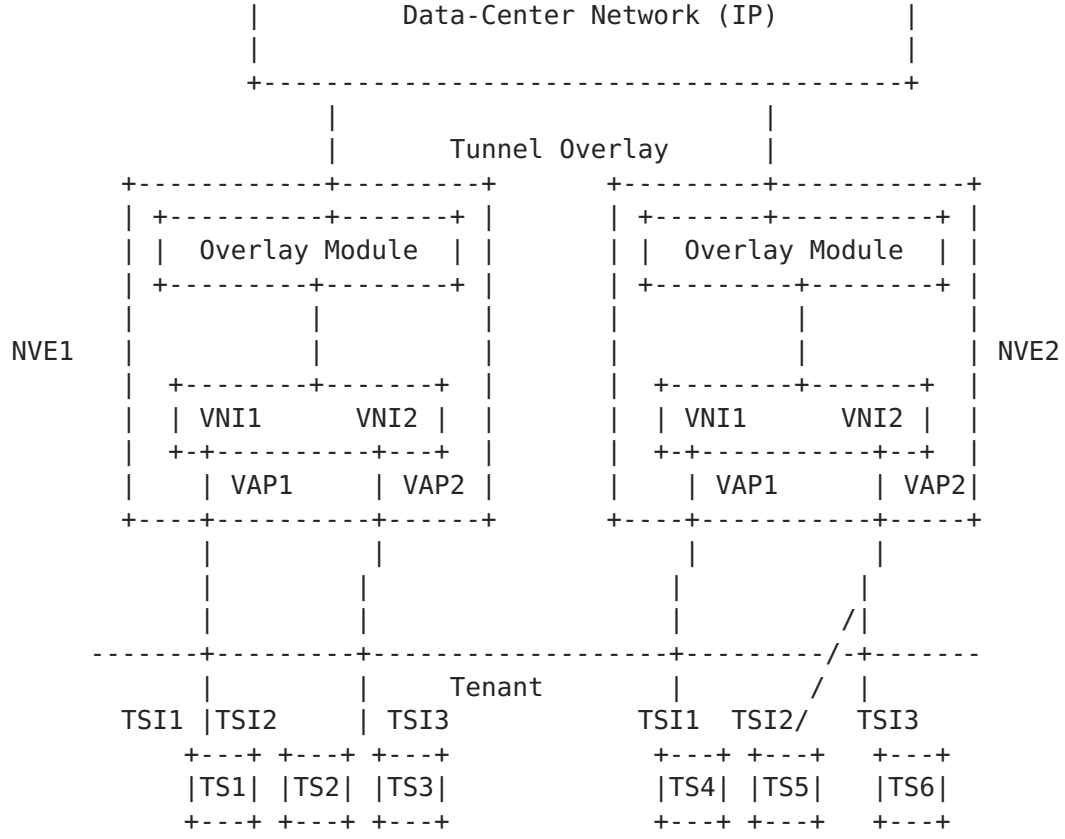

Figure 3.1. NVE Reference model in RFC 8014

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

interface YANG. If anycast gateway is enabled, the source VTEP address is the address of the anycast gateway, and a bypass address is used to uniquely identify the NVE. Otherwise, the source VTEP address is the NVE interface's own IP address.

3.2.2. 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.

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).

3.2.3. 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 [RFC8293] 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'.

3.3. Statistics

Operators can determine whether a NVE should gather statistic values on a per-VNI basis. An enabler is contained in the 'static' list as 'statistic-enable' 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

```
  +--rw nvo3
  |  +--rw vni-instances
  |     +--rw vni-instance* [vni-id]
  |        +--rw vni-id               uint32
  |        +--rw vni-mode             enumeration
```

```
|        +--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/
l3vpn: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