INTERNET-DRAFT
Intended Status: Standards Track

Mingui Zhang
Lianshu Zheng
Feng Zheng
Huawei
Fu Qiao
China Mobile

Expires: June 18, 2016

December 16, 2015

YANG Data Model for NVO3 Protocols
draft-zhang-nvo3-yang-cfg-03.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage NVO3 protocols.

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

Copyright and License Notice

Table of Contents

1. Introduction

Network Virtualization Overlays (NVO3) 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 preliminary NVO3 instances as well as their operation states. We call it the base model for NVO3 in this document.

Two default data plane encapsulation techniques for NVO3 are incorporated in the base module. It allows other possible encapsulations developed in the industry later on to be included by introducing a new encapsulation as a new 'choice'. As new solutions for NVO3 are developed, corresponding YANG data modules may be specified. 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 [RFC7364], [RFC7365], [I-D.ietf-nvo3-dataplane-requirements] and [RFC7348] is assumed in this document.

3. The YANG Data Model for NVO3

The NVO3 base YANG module 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 'nvo3Nves' 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. Encapsulation Type

The encapsulation type of NVO3 is to be determined by the operators according to their network. NVO3 encapsulation arising from the industry is not unique. The value of 'encapType' determines the encapsulation. Besides the default VxLAN encapsulation [RFC7348], the module also allows other possible encapsulation, for example the NVGRE encapsulation [I-D.sridharan-virtualization-nvgre]. It can be extended to include more encapsulation choices.

3.1.3. Source and Destination UDP Port

If the encapsulation type is VxLAN, it's recommended the source UDP port is calculated using a hash of fields (e.g., source IP address, destination IP address, protocol type, source MAC address and destination MAC address) from the inner packet. The bit value of 'srcUdpPortGenRuleFlags' indicates whether each field is used in the hashing function with the high-order bit indicating the first hashing field.

The value of destination UDP port is set to be 'destUdpPort'. For VxLAN, IANA has assigned the value 4789 for the UDP destination port, and this value SHOULD be used by default as the destination UDP port

[RFC7348].

3.1.4. 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.

3.1.5. 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.

3.1.6. BUM Mode

An NVE SHOULD support either ingress replication or point to multipoint tunnels [I-D.ietf-nvo3-dataplane-requirements] 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 'peerAddr'. 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 nvo3Nves
   |  +--rw nvo3Nve* [ifName]
   |     +--rw ifName                string
   |     +--rw srcAddr?              inet:ip-address
   |     +--rw (encapType)?
   |     |  +--:(vxlan)
   |     |     +--rw srcUdpPortGenRule?   uint8
   |     |     +--rw destUdpPort?         uint16
   |     +--rw members
   |        +--rw member* [VNI]
   |           +--rw VNI          uint32
   |           +--rw bdId         uint32
   |           +--rw (encapType)?
   |           |  +--:(vxlan)
   |           |  |  +--rw vxlanFlag?   flags-vxlan
   |           |  +--:(nvgre)
   |           |     +--rw nvgreFlag?   flags-nvgre
   |           |     +--rw flowId?      uint8
   |           +--rw (bumMode)?
   |              +--:(headEnd)
   |              |  +--rw peerAddr*    inet:ip-address
   |              +--:(multiGroup)
   |                 +--rw multiAddr?   inet:ip-address
   +--rw nvo3Infos
```



```
   |  +--rw nvo3Info* [VNI]
   |     +--rw VNI                 uint32
   |     +--rw statisticsEnable?   enumeration
   |     +--ro status?             enumeration
   +--rw nvo3Statistics
      +--ro localNVE*
      |  +--ro count?    uint64
      |  +--ro ipAddr*   inet:ip-address
      +--ro remoteNVE*
      |  +--ro count?    uint64
      |  +--ro ipAddr*   inet:ip-address
      +--rw flowStatistics
      |  +--ro instanceCount?   uint64
      |  +--ro flowStatistic*
      |     +--ro VNI?                      uint32
```

```
|     +--ro inPacketsCount?            uint64
|     +--ro inBytesCount?              uint64
|     +--ro inUnicasts?                uint64
|     +--ro inMulticasts?              uint64
|     +--ro inBroadcasts?              uint64
|     +--ro inUnknownUnicastDrops?     uint64
|     +--ro inUnknownMulticastDrops?   uint64
|     +--ro inBroadcastsDrops?         uint64
|     +--ro outPacketsCount?           uint64
|     +--ro outBytesCount?             uint64
|     +--ro outUnicasts?               uint64
|     +--ro outMulticasts?             uint64
|     +--ro outBroadcasts?             uint64
|   +--rw MacStatistics
|      +--ro MacStatistic*
|         +--ro VNI?          uint32
|         +--ro vmMacCount?   uint64
```

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

3.4. YANG Module

```
<CODE BEGINS> file "ietf-nvo3-base@2015-12-16.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";
  }
  organization "IETF NVO3 Working Group";
  contact "vero.zheng@huawei.com
```



```
    zhangmingui@huawei.com
    habby.zheng@huawei.com";
  description "nvo3 yang module";
  revision "2015-12-16" {
    description
      "Initial version";
    reference "RFC 7364";
  }
```

```
typedef flags-vxlan {
  type bits {
    bit validVni {
      position 4;
      description
        "Valid VNI.";
    }
  }
  description
    "VXLAN flags";
}

typedef  flags-nvgre {
  type bits {
    bit zero{
      position 0;
      description
        "Zero.";
    }
    bit KeyPresent{
      position 2;
      description
        "Key present.";
    }
    bit three{
      position 3;
      description
        "Three.";
    }
  }
  description
    "NVGRE flags";
}

container nvo3Nves {
  list nvo3Nve {
    key "ifName";
    leaf ifName {
      type string;
```



```
      mandatory "true";
```

```
      description
        "Interface name.";
    }
    leaf srcAddr {
      type inet:ip-address;
      description
        "Local NVE address.";
    }
    choice encapType{
      case vxlan{
        leaf srcUdpPortGenRule {
          type uint8;
          description
            "The rule of generating source udp port.";
        }
        leaf destUdpPort {
          type uint16{
            range "0 .. 65535";
          }
          default "4789";
          description
            "Destination udp port.";
        }
      }
      description
        "If the encapsulation type is VxLAN.";
    }
    container members {
      list member {
        key "VNI";
        leaf VNI {
          type uint32 {
            range "1 .. 16777215";
          }
          mandatory "true";
          description
            "Virutal Network Instance (VNI). For VxLAN, it is
            VXLAN Network Identifier; For NVGRE, it is Virtual
            Subnet Identifier.";
        }
        leaf bdId {
          type uint32 {
            range "1 .. 32768";
          }
          mandatory "true";
          description
            "Bridge Domain ID";
```

```
            }
            choice encapType{
              case vxlan{
                leaf vxlanFlag {
                  type flags-vxlan;
                  description
                    "The 8 bits flags of VXLAN. The I flag MUST
                    be set to 1 for a valid VNI.
                    Others are reserved.";
                }
              }
              case nvgre{
                leaf nvgreFlag {
                    type flags-nvgre;
                    description
                      "The 13 bits flags of NVGRE. Bit 0 and 3 are set
                      to 0 while bit 2 is set to 1.
                      Others are reserved";
                }
                leaf flowId {
                  type uint8;
                  description
                    "This is an 8-bit value that is used to provide
                    per-flow entropy for flows in the same VSID.
                    If a FlowID is not generated, it MUST be set to
                    all zero.";
                }
              }
              description
                "The encapsulation type that the overlay is using.";
            }
            choice bumMode{
              case headEnd{
                leaf-list peerAddr {
                  type inet:ip-address;
                  description
                    "Remote NVE address.";
                }
              }
              case multiGroup{
                leaf multiAddr {
                  type inet:ip-address;
                  description
                    "The multicast group address.";
                }
              }
```

```
            description
              "The replication method.";
```



```
            }
            description
              "The list of the VNIs.";
          }
          description
            "The container of the VNI list.";
        }//container members
        description
          "The list of the nvo3Nves.";
      }
      description
        "The container of the nvo3Nves list.";
    }//container nvo3Nves

    container nvo3Infos {
      list nvo3Info {
        key "VNI";
        leaf VNI {
          type uint32 {
            range "1 .. 16777215";
          }
          mandatory "true";
          description
            "Virutal Network Instance (VNI). For VxLAN, it is
            VXLAN Network Identifier; For NVGRE, it is Virtual
            Subnet Identifier.";
        }
        leaf statisticsEnable {
          type enumeration {
            enum enable {
              value "0";
              description
                "Enable";
            }
            enum disable {
              value "1";
              description
                "Disable";
            }
```

```
          }
          description
            "Collecting of statistics is enabled.";
        }
        leaf status {
          type enumeration {
            enum up {
              value "0";
              description
```



```
                "Up";
            }
            enum down {
              value "1";
              description
                "Down";
            }
          }
          config "false";
          description
            "The status can be up or down.";
        }
        description
          "The information per VNI.";
      }
      description
        "The container of the VNI info list.";
    }//container nvo3Infos

    container nvo3Statistics {
      list localNVE {
        leaf count {
          type uint64;
          config "false";
          description
            "The value of count.";
        }
        leaf-list ipAddr {
          type inet:ip-address;
          config "false";
          description
            "List of local NVE address.";
```

```
      }
      config "false";
      description
        "The number of local NVEs.";
    }
    list remoteNVE {
      config "false";
      leaf count {
        type uint64;
        config "false";
        description
          "The value of count.";
      }
      leaf-list ipAddr {
        type inet:ip-address;
        config "false";
```



```
        description
          "List of remote NVE address.";
      }
      description
        "The number of remote NVEs.";
    }
    container flowStatistics {
      grouping inputStatistics {
        leaf inPacketsCount {
          type uint64;
          config "false";
          description
            "The number of incoming packets.";
        }
        leaf inBytesCount {
          type uint64;
          config "false";
          description
            "The number of bytes.";
        }
        leaf inUnicasts {
          type uint64;
          config "false";
          description
            "The number of incoming unicast packets.";
```

```
          }
          leaf inMulticasts {
            type uint64;
            config "false";
            description
              "The number of incoming multicast packets.";
          }
          leaf inBroadcasts {
            type uint64;
            config "false";
            description
              "The number of incoming broadcast packets.";
          }
          leaf inUnknownUnicastDrops {
            type uint64;
            config "false";
            description
              "The number of incoming unknown unicast packets
              that are dropped.";
          }
          leaf inUnknownMulticastDrops {
            type uint64;
            config "false";
```



```
            description
              "The number of incoming unknown multicast packets
              that are dropped.";
          }
          leaf inBroadcastsDrops {
            type uint64;
            config "false";
            description
              "The number of incoming broadcast packets
              that are dropped";
          }
          description
            "The collection of information about incoming flows.";
        }
        grouping outputStatistics {
          leaf outPacketsCount {
            type uint64;
            config "false";
```

```
        description
          "The number of outgoing packets.";
      }
      leaf outBytesCount {
        type uint64;
        config "false";
        description
          "The number of bytes.";
      }
      leaf outUnicasts {
        type uint64;
        config "false";
        description
          "The number of outgoing unicast packets.";
      }
      leaf outMulticasts {
        type uint64;
        config "false";
        description
          "The number of outgoing multicast packets.";
      }
      leaf outBroadcasts {
        type uint64;
        config "false";
        description
          "The number of outgoing broadcast packets.";
      }
      description
        "The collection of information about outgoing flows.";
    }
```



```
    leaf instanceCount {
      type uint64;
      config "false";
      description
        "The number of instances.";
    }
    list flowStatistic{
      leaf VNI {
        type uint32;
        config "false";
        description
```

```
            "Virutal Network Instance (VNI). For VxLAN, it is
            VXLAN Network Identifier; For NVGRE, it is Virtual
            Subnet Identifier.";
          }
          config "false";
          uses inputStatistics;
          uses outputStatistics;
          description
            "The statistics per VNI.";
        }
        description
          "The statistics of the flows.";
      }//container flowStatistics
      container MacStatistics {
        list MacStatistic{
          leaf VNI {
            type uint32;
            config "false";
            description
              "Virutal Network Instance (VNI). For VxLAN, it is
              VXLAN Network Identifier; For NVGRE, it is Virtual
              Subnet Identifier.";
          }
          leaf vmMacCount {
            type uint64;
            config "false";
            description
              "Count of learning Mac addresses for the VNI.";
          }
          config "false";
          description
            "The statistics of the MAC addresses.";
        }
        description
          "The container of the MacStatistics list.";
      }//container MacStatistics
      description
```



```
        "The container of the nvo3Statistics list.";
    }//container nvo3Statistics
  }
  <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.

[I-D.ietf-nvo3-dataplane-requirements] Nabil Bitar, Marc Lasserre, et al, "NVO3 Data Plane Requirements", draft-ietf-nvo3-dataplane-requirements, 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.

[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

[I-D.zhang-nvo3-yang-active-active] M. Zhang, J. Xia, "YANG Data Model for NVO3 Active Active Access", draft-zhang-nvo3-yang-active-active, working in progress.

[I-D.sridharan-virtualization-nvgre] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre, working in progress.

Author's Addresses

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

EMail: zhangmingui@huawei.com

Lianshu Zheng
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China

EMail: vero.zheng@huawei.com

Feng Zheng
Huawei Technologies
No. 156 Beiqing Rd. Haidian District,
Beijing 100095
P.R. China

EMail: habby.zheng@huawei.com

Fu Qiao
China Mobile

EMail: fuqiao@chinamobile.com