INTERNET-DRAFT
Intended Status: Standards Track

Mingui Zhang
Yuan Gao
Haibo Wang
Huawei
Fu Qiao
China Mobile

Expires: June 18, 2016
December 16, 2018

YANG Data Model for NVO3 Protocols
draft-zhang-nvo3-yang-cfg-04.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. 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 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-vxlan
  +--rw vxlan
     +--rw common
     |  +--rw vxlan-enable?   boolean
     +--rw nves
     |  +--rw nve* [if-name source-interface]
     |     +--rw if-name                          pub-type:ifName
     |     +--rw vtep-ip?                         inet:ipv4-address-no-zone
     |     +--rw ipv6-vtep-ip?                    inet:ipv6-address-no-zone
     |     +--rw source-interface                 pub-type:ifName
     |     +--rw mac-address?                     pub-type:macAddress
     |     +--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-zon
     |     |     |     +--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    pub-type: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
|     |     +--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
|           +--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 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?    pub-type:vrfName
```

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

3.4. YANG Module

```
<CODE BEGINS> file "ietf-vxlan-base@2018-12-18.yang"

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

```
organization "IETF NVO3 Working Group";
contact "zhangmingui@huawei.com
  sean.gao@huawei.com";
description "vxlan yang module";
revision "2015-12-16" {
  description
    "Initial version";
  reference "RFC 7364";
}

 container vxlan {
  container common {
    leaf vxlan-enable {
```



```
      type boolean;
      default "false";
      description
        "Enable/Disable VXLAN featrue";
    }
  }
  container nves {
    list nve {
      key "if-name source-interface";
      ext:entry-from "system";
      leaf if-name {
        type pub-type:ifName;
        description
          "nve interface name";
      }
      leaf vtep-ip {
        type inet:ipv4-address-no-zone;
        description
          "vxlan tunnel source  address";
        ext:allowDelete "true";
      }
      leaf ipv6-vtep-ip {
        type inet:ipv6-address-no-zone;
        description
          "ipv6 vxlan tunnel source address";
        ext:allowDelete "true";
      }
      leaf source-interface {
```

```
        type pub-type:ifName;
        description
          "source interface";
      }
      leaf mac-address {
        type pub-type:macAddress;
        description
          "mac address";
        ext:allowDelete "true";
      }
      leaf bypass-vtep-ip {
        type inet:ipv4-address-no-zone;
        description
          "bypass vxlan tunnel source address";
        ext:allowDelete "true";
      }
      container vni-members {
        list vni-member {
          key "vni-id";
          description
```



```
            "nve vni member";
          leaf vni-id {
            type uint32 {
              range "1..16777215";
            }
            description
              "Associate VXLAN VNIs (Virtual Network Identifiers) with the
          }
          leaf protocol-bgp {
            type protocolType;
            default "null";
            description
              "Enables BGP EVPN with ingress replication for the VNI.";
            ext:allowDelete "true";
          }
          container peers {
            list peer {
              key "peer-ip";
              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";
              ext:allowDelete "true";
            }
            leaf split-horizon-group {
              type string {
                length "1..31";
              }
              description
                "split group name";
              ext:allowDelete "true";
            }
          }
        }
        container ipv6-peers {
          list ipv6-peer {
            key "ipv6-peer-ip";
            leaf ipv6-peer-ip {
              type inet:ipv6-address-no-zone;
              description
                "peer ipv6 address";
```



```
            }
          }
        }
        container flood-proxys {
          list flood-proxy {
            key "peer-ip";
            leaf peer-ip {
              type inet:ipv4-address-no-zone;
              description
                "peer ip address";
            }
          }
        }
        list mcast-group {
```

```
            key "mcast-group";
            leaf mcast-group {
              type inet:ipv4-address-no-zone;
              description
                "mcast ip address";
            }
          }
        }
      }
      container vni-map-l2vpns {
      list vni-map-l2vpn {
        key "vni-id";
        leaf l2vpn-id {
          type uint32 {
            range "1..16777215";
          }
          mandatory true;
          description
            "l2vpn id";
        }
        leaf l2vpn-name {
          type string {
            length "1..31";
          }
          description
            "l2vpn name";
          ext:allowDelete "true";
        }
        leaf vni-id {
          type uint32 {
            range "1..16777215";
          }
          description
            "vni id";
```



```
        }
        leaf split-horizon-group {
          type string {
            length "1..31";
          }
          description
            "split group name";
```

```
            ext:allowDelete "true";
          }
        }
      }
      container vni-map-l3vpns {
        list vni-map-l3vpn {
          key "l3vpn-name";
          leaf l3vpn-name {
            type pub-type:vrfName;
            description
              "l3vpn name";
          }
          leaf vni-id {
            type uint32 {
              range "1..16777215";
            }
            description
              "vni id";
            ext:allowDelete "true";
          }
        }
      }
    container vni-statistics-infos {
       config false;
       list vni-statistic-info {
         key "vni-id";
         config false;
         leaf vni-id {
           type uint32 {
             range "1..16777215";
           }
           config false;
           description
             "vni id";
         }
         container vni-statistics {
           config false;
           leaf rx-bits-persec {
             type uint64;
             config false;
           }
```

```
leaf rx-pkts-persec {
  type uint64;
  config false;
}
leaf tx-bits-persec {
  type uint64;
  config false;
}
leaf tx-pkts-persec {
  type uint64;
  config false;
}
leaf rx-pkts {
  type uint64;
  config false;
}
leaf rx-bytes {
  type uint64;
  config false;
}
leaf tx-pkts {
  type uint64;
  config false;
}
leaf tx-bytes {
  type uint64;
  config false;
}
leaf rx-unicast-pkts {
  type uint64;
  config false;
}
leaf rx-multicast-pkts {
  type uint64;
  config false;
}
leaf rx-broadcast-pkts {
  type uint64;
  config false;
}
leaf drop-unicast-pkts {
  type uint64;
  config false;
}
leaf drop-multicast-pkts {
  type uint64;
  config false;
}
```

```
            leaf drop-broadcast-pkts {
              type uint64;
              config false;
            }
            leaf tx-unicast-pkts {
              type uint64;
              config false;
            }
            leaf tx-multicast-pkts {
              type uint64;
              config false;
            }
            leaf tx-broadcast-pkts {
              type uint64;
              config false;
            }
          }
        }
      }
      container vnipeer-statistics-cfgs {
        list vnipeer-satistics-cfg {
          key "vni-id peer-ip";
          leaf vni-id {
            type uint32 {
              range "1..16777215";
            }
            description
              "vni id";
          }
          leaf peer-ip {
            type inet:ipv4-address-no-zone;
          }
        }
      }
      container vnipeer-statistics-infos {
        config false;
        list vnipeer-satistics-info {
          key "vni-id source-ip peer-ip";
          config false;
          leaf vni-id {
            type uint32 {
              range "0..16777215";
            }
```

```
                config false;
                description
                  "vni id";
              }
              leaf source-ip {
```



```
                type inet:ipv4-address-no-zone;
                config false;
              }
              leaf peer-ip {
                type inet:ipv4-address-no-zone;
                config false;
              }
              container vnipeer_statistics {
                config false;
                leaf rx-bits-persec {
                  type uint64;
                  config false;
                }
                leaf rx-pkts-persec {
                  type uint64;
                  config false;
                }
                leaf tx-bits-persec {
                  type uint64;
                  config false;
                }
                leaf tx-pkts-persec {
                  type uint64;
                  config false;
                }
                leaf rx-pkts {
                  type uint64;
                  config false;
                }
                leaf rx-bytes {
                  type uint64;
                  config false;
                }
                leaf tx-pkts {
                  type uint64;
                  config false;
```

```
                }
                leaf tx-bytes {
                  type uint64;
                  config false;
                }
                leaf rx-unicast-pkts {
                  type uint64;
                  config false;
                }
                leaf rx-multicast-pkts {
                  type uint64;
                  config false;
```



```
                }
                leaf rx-broadcast-pkts {
                  type uint64;
                  config false;
                }
                leaf drop-unicast-pkts {
                  type uint64;
                  config false;
                }
                leaf drop-multicast-pkts {
                  type uint64;
                  config false;
                }
                leaf drop-broadcast-pkts {
                  type uint64;
                  config false;
                }
                leaf tx-unicast-pkts {
                  type uint64;
                  config false;
                }
                leaf tx-multicast-pkts {
                  type uint64;
                  config false;
                }
                leaf tx-broadcast-pkts {
                  type uint64;
                  config false;
                }
```

```
      }
    }
  }
  container ipv6-vnipeer-statistics-cfgs {
    list ipv6-vnipeer-statistics-cfg {
      key "vni-id ipv6-source-ip ipv6-peer-ip";
      leaf vni-id {
        type uint32 {
          range "1..16777215";
        }
        description
          "vni id";
      }
      leaf ipv6-source-ip {
        type inet:ipv6-address-no-zone;
      }
      leaf ipv6-peer-ip {
        type inet:ipv6-address-no-zone;
      }
```



```
    }
  }
  container ipv6-vnipeer-statistics-infos {
    config false;
    list ipv6-vnipeer-statistics-info {
      key "vniId ipv6-source-ip ipv6-peer-ip";
      config false;
      leaf vniId {
        type uint32 {
          range "1..16777215";
        }
        config false;
        description
          "vni id";
      }
      leaf ipv6-source-ip {
        type inet:ipv6-address-no-zone;
        config false;
      }
      leaf ipv6-peer-ip {
        type inet:ipv6-address-no-zone;
        config false;
```

```
            }
            container ipv6_vnipeer_statistics {
              config false;
              leaf rx-bits-persec {
                type uint64;
                config false;
              }
              leaf rx-pkts-persec {
                type uint64;
                config false;
              }
              leaf tx-bits-persec {
                type uint64;
                config false;
              }
              leaf tx-pkts-persec {
                type uint64;
                config false;
              }
              leaf rx-pkts {
                type uint64;
                config false;
              }
              leaf rx-bytes {
                type uint64;
                config false;
```



```
              }
              leaf tx-pkts {
                type uint64;
                config false;
              }
              leaf tx-bytes {
                type uint64;
                config false;
              }
              leaf rx-unicast-pkts {
                type uint64;
                config false;
              }
              leaf rx-multicast-pkts {
                type uint64;
```

```
              config false;
            }
            leaf rx-broadcast-pkts {
              type uint64;
              config false;
            }
            leaf drop-unicast-pkts {
              type uint64;
              config false;
            }
            leaf drop-multicast-pkts {
              type uint64;
              config false;
            }
            leaf drop-broadcast-pkts {
              type uint64;
              config false;
            }
            leaf tx-unicast-pkts {
              type uint64;
              config false;
            }
            leaf tx-multicast-pkts {
              type uint64;
              config false;
            }
            leaf tx-broadcast-pkts {
              type uint64;
              config false;
            }
          }
        }
      }
```



```
    }
  }
  container vnipeer-infos {
    config false;
    list vnipeer-info {
      key "vni-id source-ip peer-ip";
      config false;
      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
              "tunnel type";
          }
          leaf out-vni-id {
            type uint32 {
              range "1..16777215";
            }
            config false;
            description
              "out vni id";
          }
        }
      }
      container vni-infos {
        list vni_info {
          key "vni-id";
          leaf vni-id {
            type uint32 {
```



```
              range "1..16777215";
```

```
        }
        description
          "vni id";
      }
      leaf statistics-enable {
        type vniStatisticsEnable;
        default "disable";
        ext:allowDelete "true";
      }
    }
  }
  container tunnel-infos {
    config false;
    list tunnel-infos {
      key "tunnel-id";
      config false;
      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";
        ext:support-filter "true";
      }
      leaf peer-ip {
        type inet:ip-address-no-zone;
        config false;
        description
          "peer ip";
        ext:support-filter "true";
      }
      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 pub-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, Shouchuan Yang, Feng Zheng, Lianshu Zheng, 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

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

Fu Qiao
China Mobile

EMail: fuqiao@chinamobile.com