TRILL Working Group
INTERNET-DRAFT
Intended status: Standards Track

W. Hao
Y. Li
Huawei Technologies
D. Kumar
Cisco
M. Durrani
Cisco
H. Zhai
JIT
L. Xia
Huawei Technologies

Expires: November 2015
May 26, 2015

TRILL YANG Data Model
draft-ietf-trill-yang-02.txt

Abstract

This document defines a YANG data model for TRILL protocol.

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 Notice

Copyright (c) 2015 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 2
2. Conventions Used in This Document..... 2
3. Design of Data model..... 3
4. TRILL YANG Data model..... 6
5. Security Considerations..... 22
6. IANA Considerations 22
7. References 22
   7.1. Normative References..... 22
   7.2. Informative References..... 23
8. Acknowledgments 23

1. Introduction

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 defines a YANG [RFC6020] data model for the operation of TRILL base protocol.

2. Conventions Used in This Document

This document uses the acronyms defined in [RFC6325], in addition to the following:

CSNP: Complete Sequence Number Protocol Data Unit
DRB: Designated RBridge

IS-IS: Intermediate System to Intermediate System
LSDB: Link State Database
MAC: Media Access Control address
MTU: Maximum Transmission Unit
NETCONF: Network Configuration Protocol
PSNP: Partial Sequence Number Packet
RBridge: An alternative name for a TRILL Switch
RPF: Reverse Path Forward
SNP: Scalable Network Pack
SSH: Secure Shell
VLAN: Virtual Local-Area Network

3. Design of Data model

There is only one module for the TRILL base protocol. The module can be augmented for other TRILL extended features with their specific definitions, such as TRILL active-active connection, TRILL Fine Grained Label, etc.

The TRILL Yang module includes one container of trillSites which contains a list of instances as many implementations are currently supporting multiple ISIS instances within a single RBridge. The configuration data is divided into four categories which include per RBridge, per nickname per RBridge, per port per RBridge, and per VLAN per RBridge. The operating status includes the information of LSDB, unicast and multicast routing table, RPF check, nickname, peer and some statistics.

The figure below describes the overall structure of the TRILL Yang model:

```
module: trill
+--rw instances* [instanceID]
|  +--rw instanceID
|  +--rw name
|  +--rw network-entity* [netEntity]
|  +--rw lspTimer
|  |  +--rw lspLife
|  |  +--rw lspRefresh
|  |  +--rw lspGeneration
|  +--rw nickNumber
|  +--rw treeNumber
|  +--rw referenceBandwidth
|  +--rw nativeConfidence
|  +--rw remoteConfidence
|  +--rw minLinkMTU
|  +--rw MTUProbes
|  +--rw nicknames* [nickname]
|  |  +--rw nickname
|  |  +--rw priority
|  |  +--rw rootPriority
|  +--rw trillPorts* [ifName]
|  |  +--rw ifName
|  |  +--rw portMode
|  |  +--rw timer
|  |  |  +--rw csnp
|  |  |  +--rw hello
|  |  |  +--rw holdingMultiplier
|  |  |  +--rw lspRetransmit
|  |  |  +--rw lspThrottle
|  |  |  |  +--rw throttleInterval
|  |  |  |  +--rw countNumber
|  |  |  +--rw inhibitionTimer
|  |  +--rw drbConfig
|  |  |  +--rw drbPriority
|  |  |  +--rw holdingTimer
|  |  +--rw macLearningFlag
|  |  +--rw trillFrameReceiveFlag
|  |  +--rw cost
|  |  +--rw enabledVlans
|  |  +--rw announcingVlans
|  |  +--rw forwardingVlans
|  |  +--rw designatedVlan
|  |  +--ro circuitId
|  |  +--ro circuitMTU
|  |  +--ro drbStatus
|  |  +--ro trillStatus
```

```
   |  |  +--ro designatedVlan
   |  +--rw vlanConfigs* [vlanID]
   |  |  +--rw vlanID
   |  |  +--rw participationFlag
   |  |  +--rw priority
   |  |     +--rw priority
   |  |     +--rw holdingTimer
   |  +--ro trillLsdbInfos* [lspId]
   |  |  +--ro lspId
   |  |  +--ro seqenceNumber
   |  |  +--ro checkSum
   |  |  +--ro lspLength
   |  |  +--ro attBit
   |  |  +--ro partitionBit
   |  |  +--ro overloadBit
   |  |  +--ro holdTime
   |  |  +--ro localLsp
   |  +--ro trillRouteInfos* [nickname nextHop]
   |  |  +--ro nickName
   |  |  +--ro cost
   |  |  +--ro outInterface
   |  |  +--ro outVlan
   |  |  +--ro nextHop
   |  |  +--ro hopCount
   |  +--ro trillRpfCheckInfos* [ingressNickname treeNickname]
   |  |  +--ro ingressNickname
   |  |  +--ro treeNickname
   |  |  +--ro interfaceName
   |  |  +--ro neighborMac
   |  |  +--ro designatedVlan
   |  +--ro trillMRouteInfos* [vlan rootNickname]
   |  |  +--ro vlan
   |  |  +--ro rootNickname
   |  |  +--ro hopCount
   |  |  +--ro outInterfaceinfo* [outInterface outVlan]
   |  |  |  +--ro outInterface
   |  |  |  +--ro outVlan
   |  +--ro trillNicknameInfos* [nickname systemId]
   |  |  +--ro nickName
   |  |  +--ro priority
   |  |  +--ro rootPriority
   |  |  +--ro systemId
   |  |  +--ro conflictState
   |  |  +--ro staticFlag
   |  |  +--ro isLocal
   |  +--ro trillPeerInfos* [hostName circuitId]
   |  |  +--ro hostName
```

```
|  |  +--ro interfaceName
|  |  +--ro circuitId
|  |  +--ro status
|  |  +--ro holdTime
|  |  +--ro priority
|  +--ro trillStatistics
|  |  +--ro interfaceStat
|  |  |  +--ro upNum
|  |  |  +--ro downNum
|  |  +--ro broadcastPeerStat
|  |  |  +--ro reportNum
|  |  |  +--ro detectNum
|  |  |  +--ro twoWayNum
|  |  +--ro p2pPeerStat
|  |  |  +--ro upNum
|  |  |  +--ro initNum
|  |  +--ro unicastRoutesNum
|  |  +--ro multicastRoutesNum
|  |  +--ro rpfEntrysNum
|  |  +--ro remoteNicknamesNum
|  |  +--ro lsdbLSPsNum
|  |  +--ro selfLSPsNum
|  |  +--ro multicastTreesNum
|  |  +--ro unicastNodesNum
|  |  +--ro multicastNodesNum
|  +--ro pktStatistics* [interfaceName]
|  |  +--ro interfaceName
|  |  +--ro sentHellosNum
|  |  +--ro recvedHellosNum
|  |  +--ro sentLSPsNum
|  |  +--ro recvedLSPsNum
|  |  +--ro sentCSNPsNum
|  |  +--ro recvedCSNPsNum
|  |  +--ro sentPSNPsNum
|  |  +--ro recvedPSNPsNum
|  |  +--ro lspRetransmissionsNum
|  |  +--ro drbElectionsNum
+  +
```

4. TRILL YANG Data model

```
<CODE BEGINS> file "ietf-trill@2015-05-26.yang"

module ietf-trill {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-trill";
  prefix ietf-trill;
```

```
     revision 2015-04-30{
       description "creating trill base model";
     }

     container trillSites {
       list trillSite {
        key "instanceId";

        leaf instanceId {
          config "true";
          type uint32 {
           range "0..4294967295";
          }
        }
        leaf name {
          config "true";
          type string;
        }
        container trillNetEntitys {
          list trillNetEntity {
            key "netEntity";

            leaf netEntity {
             config "true";
             type string;
            }
          }
        }
        container lspTimer {
          leaf lspLife {
            description "LSP aging timer.Unit:Second";
            config "true";
            default "1200";
            type uint16 {
              range "2..65535";
            }
          }
             leaf lspRefresh {
            description "LSP refresh timer.Unit:Second";
              config "true";
              default "900";
              type uint16 {
                range "1..65534";
              }
          }
          leaf lspGeneration {
```

```
            description "LSP generate timer.Unit:Second";
            config "true";
            default "2";
            type uint8 {
              range "1..120";
            }
            }
          }
          leaf nickNumber {
            config "true";
            default "1";
            type uint16 {
              range "1..256";
            }
          }
          leaf treeNumber {
            config "true";
            default "1";
            type uint16 {
              range "1..65535";
            }
          }
          leaf referenceBandwidth {
            config "true";
            default "20000000";
            type uint32 {
              range "1..2147483648";
            }
          }
          leaf nativeConfidence {
            description "The confidence in { MAC, VLAN, local port }
   triples learned from locally received native frames";
            config "true";
            default "32";
            type uint8 {
              range "0..255";
            }
          }
          leaf remoteConfidence {
            description "The confidence in { MAC, VLAN, remote
   RBridge } triples learned from decapsulating frames";
            config "true";
            default "32";
            type uint8 {
              range "0..255";
            }
          }
```

```
          leaf minLinkMTU {
            config "true";
            default "1470";
            type uint16 {
              range "1..65535";
            }
          }
          leaf MTUProbes {
            description "The number of failed MTU-probes";
            config "true";
            default "3";
            type uint8 {
              range "1..255";
            }
          }
          container Nicknames {
            list Nickname {
              key "nickName";
              leaf nickName {
                config "true";
                mandatory "true";
                type uint16 {
                    range "1..65471";
                }
              }
              leaf priority {
                config "true";
                default "192";
                type uint8 {
                    range "128..255";
                }
              }
              leaf rootPriority {
                config "true";
                default "32768";
                type uint16 {
                    range "1..65535";
                }
              }
            }
          }
        container trillPorts {
          list trillPort {
            key "ifName";
            leaf ifName {
              description "trill interface";
              type string;
```

```
            }
            leaf portMode {
              config "true";
              default "p2p";
              type enumeration {
                enum "access" {
                  value "0";
                  description "access:only process native frame";
                }
                enum "p2p" {
                  value "1";
                  description "p2p:use IS-IS P2P Hellos";
                }
                enum "trunk" {
                  value "2";
                  description "trunk:only process TRILL frames";
                }
                enum "hybrid" {
                  value "3";
                  description "hybrid: both trunk and access port";
                }
              }
            }
            container timer {
              leaf csnp {
              config "true";
              default "10";
              type uint16 {
                range "1..65535";
              }
            }
            leaf hello {
              config "true";
              default "10";
              type uint8 {
                range "3..255";
              }
            }
            leaf holdingMultiplier {
              config "true";
              default "3";
              type uint16 {
                range "3..1000";
              }
            }
            leaf lspRetransmit {
              config "true";
```

```
              default "5";
              type uint16 {
                range "1..300";
              }
            }
            container lspThrottle {
              leaf throttleInterval {
                description "The interval timer between two LSP
   messages.Unit:ms";
                config "true";
                default "50";
                type uint16 {
                  range "1..10000";
                }
              }
              leaf countNumber {
                description "The max messages number being sent each
   time.Unit:ms";
                  config "true";
                  default "10";
                  type uint16 {
                    range "1..1000";
                  }
                }
            }
               leaf inhibitionTimer {
                 description "The inhibition time for the port when
   root bridge changes.Unit:Second";
              config "true";
              default "30";
              type uint8 {
                range "0..30";
              }
            }
               }
               container drbConfig {
                 leaf drbPriority {
              config "true";
              default "64";
              type uint8 {
                range "0..127";
              }
              }
              leaf holdingTimer {
                config "true";
                default "10";
                type uint8 {
```

```
                  range "3..255";
                }
              }
            }
            leaf macLearningFlag {
              description "if learning MAC address from locally
   received native frames";
              config "true";
              default "true";
              type boolean;
            }
            leaf trillFrameReceiveFlag {
              description "if receiving of TRILL frames from non IS-IS
   adjacency";
              config "true";
              default "false";
              type boolean;
            }
            leaf cost {
              config "true";
              default "0";
              type uint32 {
                range "0..16777215";
              }
            }
            leaf enabledVlans {
              config "true";
              type binary{
                length "1..4096";
              }
            }
            leaf announcingVlans {
              config "true";
              type binary{
                length "1..4096";
              }
            }
            leaf forwardingVlans {
              config "true";
              type binary{
                length "1..4096";
              }
            }
            leaf circuitId {
              config "false";
              type uint8;
            }
```

```
            leaf circuitMTU {
              config "false";
              type uint32;
            }
            leaf designatedVlan {
              config "true";
              type uint16{
                range "1..4096";
              }
            }
            leaf drbStatus {
              config "false";
              type enumeration {
                enum "Non-DRB" {
                  value "0";
                  description "Non-DRB:";
                }
                enum "DRB" {
                  value "1";
                  description "DRB:";
                }
                enum "Down" {
                  value "2";
                  description "Down:";
                }
                enum "Suspended" {
                  value "3";
                  description "Suspended:";
                }
              }
            }
          }
        }
           container vlanConfigs {
             list  vlanConfig{
             key "vlanID";
          config "true";

             leaf vlanID {
          config "true";
          type uint16 {
            range "1..4096";
          }
          }
             leaf participationFlag {
            config "true";
```

```
            default "false";
            type boolean;
          }
             leaf priority {
            config "true";
            default "64";
            type uint8 {
              range "0..127";
            }
          }
          leaf holdingTimer {
            config "true";
            default "10";
            type uint8 {
              range "3..255";
            }
          }
             }
           }
        container trillRouteInfos {
          list  trillRouteInfo {
            key "nickName nextHop";
            config "false";

            leaf nickName {
              config "false";
              type uint32;
            }
            leaf cost {
              config "false";
              type uint32;
            }
            leaf outInterface {
              config "false";
              type string;
            }
            leaf outVlan {
              config "false";
              type uint32;
            }
            leaf nextHop {
              config "false";
              type string;
            }
            leaf hopCount {
              config "false";
              type uint32;
```

```
            }
          }
        }

        container trillMRouteInfos {
          list trillMRouteInfo {
            description "Distribution pruning tree route table; For
   non-pruning tree, VLAN is set to be 0XFFFF";
            key "vlan rootNickname";
            config "false";

            leaf vlan {
              config "false";
              type uint16;
            }
            leaf rootNickname {
              config "false";
              type uint16;
            }
            leaf hopCount {
              config "false";
              type uint16;
            }
            container outInterfaceinfo {

              list trillMRouteOutInterfaceInfo {

                key "outInterface outVlan";
                config "false";

                leaf outInterface {
                  config "false";
                  type "string";
                }
                leaf outVlan {
                  config "false";
                  type "uint32";
                }
              }
            }
          }
        }

        container trillRpfCheckInfos {
          list trillRpfCheckInfo {
            key "ingressNickname treeNickname";
            config "false";
```

```
            leaf ingressNickname {
              config "false";
              type "uint16";
            }
            leaf treeNickname {
              config "false";
              type "uint16";
            }
            leaf interfaceName {
              config "false";
              type "string";
            }
            leaf neighborMac {
              config "false";
              type "string";
            }
            leaf outVlan {
              config "false";
              type "uint16";
            }
          }
        }

        container trillPeerInfos {
          list trillPeerInfo {
            key "hostName circuitId";
            config "false";

            leaf hostName {
              description "Peer RBridge name";
              config "false";
              type "string";
            }
            leaf interfaceName {
              config "false";
              type "string";
            }
            leaf circuitId {
              config "false";
              type "string";
            }
            leaf status {
              config "false";
              type enumeration {
              enum "report" {
                value "0";
```

```
                description "report:";
              }
              enum "detect" {
                value "1";
                description "detect:";
              }
              enum "down" {
                value "2";
                description "down:";
              }
              enum "2way" {
                value "3";
                description "2way:";
              }
              }
            }
            leaf holdTime {
              config "false";
              type "uint32";
            }
            leaf priority {
              config "false";
              type "string";
            }
          }
        }

        container trillLsdbInfos {
          list trillLsdbInfo {
            key "lspId";
            config "false";

            leaf lspId {
              config "false";
              type "string";
            }
            leaf seqenceNumber {
              config "false";
              type "string";
            }
            leaf checkSum {
              config "false";
              type "string";
            }
            leaf lspLength {
              config "false";
              type uint32 {
```

```
                range "0..2000";
              }
            }
            leaf attBit {
              config "false";
              type uint8 {
                range "0..1";
              }
            }
            leaf partitionBit {
              config "false";
              type uint8 {
                range "0..1";
              }
            }
            leaf overloadBit {
              config "false";
              type uint8 {
                range "0..1";
              }
            }
            leaf holdTime {
              config "false";
              type "string";
            }
            leaf localLsp {
              config "false";
              type "boolean";
            }
          }
        }
        container trillNicknameInfos {
          list trillNicknameInfo {

            key "nickName systemId";
            config "false";

            leaf nickName {
              config "false";
              type uint32;
            }
            leaf priority {
              config "false";
              type uint32;
            }
            leaf rootPriority {
              config "false";
```

```
              type uint32;
            }
            leaf systemId {
              config "false";
              type string;
            }
            leaf conflictState {
              config "false";
              type enumeration {
              enum "S" {
                value "0";
                description "S:";
              }
              enum "A" {
                value "1";
                description "A:";
              }
              }
            }
            leaf staticFlag {
              config "false";
              type enumeration {
              enum "S" {
                value "0";
                description "S:";
              }
              enum "D" {
                value "1";
                description "D:";
              }
              }
            }
            leaf isLocal {
              config "false";
              type boolean;
            }
          }
        }
        container trillStatistics {

             container interfaceStat {
               leaf upNum {
                  config "false";
                  type uint32;
               }
               leaf downNum {
                  config "false";
```

```
                 type uint32;
               }
             }
             container pktstatistics {
               leaf reportNum {
                 config "false";
                 type uint32;
               }
               leaf detectNum {
                  config "false";
                  type uint32;
               }
               leaf twoWayNum {
                 config "false";
                 type uint32;
               }
             }
             leaf unicastRoutesNum {
               config "false";
               type uint32;
             }
             leaf multicastRoutesNum {
               config "false";
               type uint32;
             }
             leaf rpfEntrysNum {
               config "false";
               type uint32;
             }
             leaf remoteNicknamesNum {
               config "false";
               type uint32;
             }
             leaf lsdbLSPsNum {
               config "false";
               type uint32;
             }
             leaf selfLSPsNum {
               config "false";
               type uint32;
             }
             leaf multicastTreesNum {
               config "false";
               type uint32;
             }
             leaf unicastNodesNum {
               config "false";
```

```
              type uint32;
            }
            leaf multicastNodesNum {
              config "false";
              type uint32;
            }
          }

          container pktStatistics {
            list pktStatistic {
              key "interfaceName";
              config "false";

              leaf interfaceName {
                config "false";
                type string;
              }
              leaf sentHellosNum {
                config "false";
                type uint32;
              }
              leaf recvedHellosNum {
                config "false";
                type uint32;
              }
              leaf sentLSPsNum {
                config "false";
                type uint32;
              }
              leaf recvedLSPsNum {
                config "false";
                type uint32;
              }
              leaf sentCSNPsNum {
                config "false";
                type uint32;
              }
              leaf recvedCSNPsNum {
                config "false";
                type uint32;
              }
              leaf sentPSNPsNum {
                config "false";
                type uint32;
              }
              leaf recvedPSNPsNum {
                config "false";
```

```
                type uint32;
              }
              leaf lspRetransmissionsNum {
                config "false";
                type uint32;
              }
              leaf drbElectionsNum {
                config "false";
                type uint32;
              }
            }
          }
        }
      }
    }

    <CODE ENDS>
```

5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241] [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242] [RFC6242]. The NETCONF access control model [RFC6536] [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

6. IANA Considerations

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

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

7.2. Informative References

[RFC6325] Perlman, R., et.al., "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, July 2011.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC6241, June 2011.

8. Acknowledgments

The authors wish to acknowledge the important contributions of Donald Eastlake, Susan Hares, Guangying Zheng, Xianping Zhang, Wenxia Hou, Zhibo Hu.

Authors' Addresses

Weiguo Hao
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China
Phone: +86-25-56623144
Email: haoweiguo@huawei.com

Yizhou Li
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China
Phone: +86-25-56625375
Email: liyizhou@huawei.com

Deepak Kumar
CISCO Systems
510 McCarthy Blvd
Milpitas, CA 95035.
Email: dekumar@cisco.com

Muhammad Durrani
Cisco
Email: mdurrani@cisco.com

Hongjun Zhai
Jinling Institute of Technology
99 Hongjing Avenue, Jiangning District
Nanjing, Jiangsu 211169
China
Email: honjun.zhai@tom.com

Liang Xia
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China
Email: frank.xialiang@huawei.com