TRILL

Weiguo Hao
Liang Xia
Yizhou Li
Huawei

Internet Draft

Intended status: Informational
Expires: March 2015

September 1,2014

TRILL YANG Data Model
draft-hao-trill-yang-01.txt

Status of this Memo

Abstract

This document presents YANG data model for TRILL protocol [RFC6325].

Table of Contents

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 presents YANG [RFC6020] data model for the operation of TRILL base protocol [RFC6325].

2. Requirements Language

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 [RFC2119].

3. Design of Data model

There is only one module for the TRILL base protocol [RFC6325]. The module can be augmented for other TRILL extended features with their specific definitions, such as TRILL active-active connection, TRILL Fine Grained Label, and etc.
The TRILL Yang module includes one container named trillSites which contains a list of TRILL instances as single RBridge MAY support multiple TRILL instance. The configuration data is divided into four categories which are per RBridge, per nickname per RBridge, per port per RBridge and per VLAN per RBridge. The non-configuration data 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. The following notations are used within the data tree.

```
    rw  for configuration data
    ro  for non-configuration data
```

```
module: trill
   +--rw trillSites
      +--rw trillSite* [instanceId]
         +--rw instanceId            uint32
         +--rw name?                 string
         +--rw systemID*             uint8
         +--rw lspTimer
         |  +--rw lspLife            uint16
         |  +--rw lspRefresh         uint16
         |  +--rw lspGeneration      uint8
         +--rw nickNumber            uint16
         +--rw treeNumber            uint16
         +--rw referenceBandwidth    uint32
         +--rw nativeConfidence      uint8
         +--rw remoteConfidence      uint8
         +--rw minLinkMTU            uint16
         +--rw MTUProbes             uint8
         +--rw Nicknames
         |  +--rw Nickname* [nickName]
         |     +--rw nickName        uint16
         |     +--rw priority        uint8
         |     +--rw rootPriority    uint16
         +--rw trillPorts
         |  +--rw trillPort* [ifName]
         |     +--rw ifName                   string
         |     +--rw portMode                 enumeration
         |     +--rw timer
         |     |  +--rw csnp                uint16
         |     |  +--rw hello               uint8
         |     |  +--rw holdingMultiplier   uint16
         |     |  +--rw lspRetransmit       uint16
         |     |  +--rw lspThrottle
         |     |  |  +--rw throttleInterval   uint16
         |     |  |  +--rw countNumber        uint16
         |     |  +--rw inhibitionTimer       uint8
         |     +--rw drbConfig
         |     |  +--rw drbPriority    uint8
         |     |  +--rw holdingTimer   uint8
         |     +--rw macLearningFlag         boolean
         |     +--rw trillFrameReceiveFlag   boolean
         |     +--rw cost                    uint32
         |     +--rw enabledVlans            binary
         |     +--rw announcingVlans         binary
         |     +--rw forwardingVlans         binary
         |     +--ro circuitId               uint8
```

```
            |     +--ro circuitMTU              uint32
            |     +--rw designatedVlan          uint16
            |     +--ro drbStatus               enumeration
            +--rw vlanConfigs
            |  +--rw vlanConfig* [vlanID]
            |     +--rw vlanID               uint16
            |     +--rw participationFlag    boolean
            |     +--rw priority             uint8
            |     +--rw holdingTimer         uint8
            +--rw trillRouteInfos
            |  +--ro trillRouteInfo* [nickName nextHop]
            |     +--ro nickName        uint16
            |     +--ro cost            uint32
            |     +--ro outInterface    string
            |     +--ro outVlan         uint16
            |     +--ro nextHop         string
            |     +--ro hopCount        uint32
            +--rw trillMRouteInfos
            |  +--ro trillMRouteInfo* [vlan rootNickname]
            |     +--ro vlan                uint16
            |     +--ro rootNickname        uint16
            |     +--ro hopCount            uint16
            |     +--ro outInterfaceinfo
            |        +--ro trillMRouteOutInterfaceInfo* [outInterface
   outVlan
            |           +--ro outInterface    string
            |           +--ro outVlan         uint16
            +--rw trillRpfCheckInfos
            |  +--ro trillRpfCheckInfo* [ingressNickname treeNickname]
            |     +--ro ingressNickname       uint16
            |     +--ro treeNickname          uint16
            |     +--ro interfaceName         string
            |     +--ro neighborMac           string
            |     +--ro designatedVlan        uint16
            +--rw trillPeerInfos
            |  +--ro trillPeerInfo* [hostName circuitId]
            |     +--ro hostName              string
            |     +--ro interfaceName         string
            |     +--ro circuitId             uint8
            |     +--ro status                enumeration
            |     +--ro holdTime              uint16
            |     +--ro priority              uint8
            +--rw trillLsdbInfos
            |  +--ro trillLsdbInfo* [lspId]
            |     +--ro lspId                 lspID
            |     +--ro seqenceNumber         uint32
            |     +--ro checkSum              uint16
```

```
|     +--ro lspLength             uint32
|     +--ro attBit                uint8
|     +--ro partitionBit          uint8
|     +--ro overloadBit           uint8
|     +--ro holdTime              uint16
|     +--ro localLsp              boolean
+--rw trillNicknameInfos
|  +--ro trillNicknameInfo* [nickName systemId]
|     +--ro nickName              uint32
|     +--ro priority              uint8
|     +--ro rootPriority          uint32
|     +--ro systemId              string
|     +--ro conflictState         enumeration
|     +--ro staticFlag            enumeration
|     +--ro isLocal               boolean
+--rw trillStatistics
|  +--rw interfaceStat
|  |  +--ro upNum                    uint32
|  |  +--ro downNum                  uint32
|  +--rw pktstatistics
|  |  +--ro reportNum                uint32
|  |  +--ro detectNum                uint32
|  |  +--ro twoWayNum                uint32
|  +--ro unicastRoutesNum            uint32
|  +--ro multicastRoutesNum          uint32
|  +--ro rpfEntrysNum                uint32
|  +--ro remoteNicknamesNum          uint32
|  +--ro lsdbLSPsNum                 uint32
|  +--ro selfLSPsNum                 uint32
|  +--ro multicastTreesNum           uint32
|  +--ro unicastNodesNum             uint32
|  +--ro multicastNodesNum           uint32
+--rw pktStatistics
   +--ro pktStatistic* [interfaceName]
      +--ro interfaceName           string
      +--ro sentHellosNum           uint32
      +--ro recvedHellosNum         uint32
      +--ro sentLSPsNum             uint32
      +--ro recvedLSPsNum           uint32
      +--ro sentCSNPsNum            uint32
      +--ro recvedCSNPsNum          uint32
      +--ro sentPSNPsNum            uint32
      +--ro recvedPSNPsNum          uint32
      +--ro lspRetransmissionsNum   uint32
      +--ro drbElectionsNum         uint32
```

4. TRILL YANG Data model

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

  import ietf-yang-types {
    prefix yang;
  }

  revision 2014-08-10{
    description "creating trill base model";
  }

  typedef lspID {
    type string;
    description "The 8 byte LSP ID, consisting of the SystemID, Circuit
ID,            and
                Fragment Number.";
  }

  container trillSites {
    list trillSite {
      key "instanceId";
      leaf instanceId {
        description "TRILL instance ID";
        config "true";
        type uint32 {
          range "0..4294967295";
        }
      }
      leaf name {
        description "TRILL instance name";
        config "true";
        type string;
      }
      leaf-list systemID {
        config "true";
        max-elements "6";
```

```
        type uint8 {
          range "0..255";
        }
      }

      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 {
        description "Maximum nickname number";
        config "true";
        default "1";
        type uint16 {
          range "1..256";
        }
```

```
      }
      leaf treeNumber {
        description "Maximum distribution tree number";
        config "true";
        default "1";
        type uint16 {
          range "1..65535";
        }
      }
      leaf referenceBandwidth {
          description "TRILL reference bandwidth";
        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 "Only process native frame";
              }
              enum "p2p" {
                value "1";
                description "Use IS-IS P2P Hellos";
              }
              enum "trunk" {
                value "2";
                description "Only process TRILL frames";
              }
              enum "hybrid" {
                value "3";
                description "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 uint16;
          }
          leaf cost {
            config "false";
            type uint32;
          }
          leaf outInterface {
            config "false";
            type string;
          }
          leaf outVlan {
            config "false";
            type uint16;
          }
          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 uint16;
              }
```

```
            }
          }
        }
      }

      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 designatedVlan {
            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 uint8;
          }
          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 uint16;
          }
          leaf priority {
            config "false";
```

```
            type uint8;
          }
        }
      }

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

          leaf lspId {
            type lspID;
          }
          leaf seqenceNumber {
            config "false";
            type uint32;
          }
          leaf checkSum {
            config "false";
            type uint16;
          }
          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 uint16;
          }
          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 uint8;
          }
          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;
          }
        }
      }
    }
  }
}
```

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 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

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

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