Abstract

This document defines a YANG data model for BIER TE configuration and operation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on April 29, 2017.

Copyright Notice

Copyright (c) 2016 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

1. Introduction

[I-D.eckert-bier-te-arch] introduces an architecture for BIER-TE: Traffic Engineering for Bit Index Explicit Replication (BIER). This document defines a YANG data model for BIER TE. The content is in keeping with the TE architecture draft.

2. Design of the Data Model

Instead of using respective sub-domain-id, si and bsl information like in BIER yang draft [I-D.ietf-bier-bier-yang], this document tries to group these sub-domain-id, si and bsl information in a new bier-common grouping to simplify the reference. Later Yang modules may import the common grouping easily. Further, if this optimization is recognized, then BIER Yang draft [I-D.ietf-bier-bier-yang] will be updated to group these sub-domain-id, si and bsl information as well.

```yaml
module: ietf-bier-te
augment /rt:routing:
  +--rw bier-te-config
    +--rw te-subdomain* [subdomain-id]
      +--rw subdomain-id bier:sub-domain-id
      +--rw adj-id* [adjID]
        +--rw adjID adjid
        +--rw adj-if uint32
        +--rw (te-adjID-type)
          +--:(p2p)
          +--:(bfer)
          +--:(leaf-bfer)
          +--:(lan)
          +--:(spoke)
          +--:(ring-clockwise)
          +--:(ring-counterclockwise)
          +--:(ecmp)
          +--:(virtual-link)
```

augment /rt:routing-state:
  ++--ro bier-te-state
  ++--ro te-subdomain* [subdomain-id]
    ++--ro subdomain-id bier:sub-domain-id
    ++--ro adj-id* [adjID]
      ++--ro adjID adjid
      ++--ro adj-if uint32
      ++--ro (te-adjID-type)
        ++--:(p2p)
        ++--:(bfer)
        ++--:(leaf-bfer)
        ++--:(lan)
        ++--:(spoke)
        ++--:(ring-clockwise)
        ++--:(ring-counterclockwise)
        ++--:(ecmp)
        ++--:(virtual-link)
        ++--:(other)
  ++--ro te-bsl* [fwd-bsl]
    ++--ro fwd-bsl uint16
    ++--ro te-si* [si]
      ++--ro si bier:si
      ++--ro te-f-index* [te-f-index]
        ++--ro te-f-index bit-string
        ++--ro (te-adj-type)
          ++--:(connected)
          ++--:(routed)
          ++--:(local-decap)
          ++--:(ecmp)
          ++--:(other)
        ++--ro f-bm bit-string
        ++--ro f-intf uint32
        ++--ro ecmp? boolean
        ++--ro ecmp-index? uint16
        ++--ro frr? boolean
      ++--ro ecmp-path* [index]
        ++--ro index uint32
        ++--ro number* [number]
        ++--ro number uint16
        ++--ro out-if uint32
    ++--ro btaft* [adj-index]
      ++--ro adj-index uint32
      ++--ro bitposition bit-string
      ++--ro resetbitmask bit-string
      ++--ro addbitmask bit-string
3. BIER-TE configuration

The BIER-TE information is indexed by the sub-domain ID. Maybe there are some global BIER-TE information, it should be added in later version.
One interface can be used in different sub-domain, so the BIER TE adjacency information is managed by BIER TE other than by interface.

Because the BIER-TE is controlled by controller now, the information about IGP is not defined. If in the future the IGP is used to carry the information about BIER-TE, the IGP extension will be added in this document.

4. Notifications

If the adjacency id of one adjacency is set to zero, the value is invalid. The notification should be sent to controller and network manager.

5. BIER TE YANG module

<CODE BEGINS> file "ietf-bier-te.yang"

module ietf-bier-te {

    namespace "urn:ietf:params:xml:ns:yang:ietf-bier-te";

    prefix bier-te;

    import ietf-routing {
        prefix "rt";
    }

    import ietf-bier {
        prefix "bier";
    }

    organization " IETF BIER(Bit Indexed Explicit Replication ) Working Group";

contact
"WG List:  <mailto:bier@ietf.org>
WG Chair: Tony Przygienda
        <mailto:tonysietf@gmail.com>
WG Chair: Greg Shepherd
        <mailto:gjshep@gmail.com>

Editor:   Zheng Zhang
        <mailto:zhang.zheng@zte.com.cn>
Editor:   Cui Wang
        <mailto:wang.cui1@zte.com.cn>
Editor:   Ran Chen
        <mailto:chen.ran@zte.com.cn>
Editor:   Fangwei Hu
        <mailto:hu.fangwei@zte.com.cn>


This module contains a collection of YANG definitions for managing BIER TE information."

revision 2016-10-13 {
   description
   "Initial version."
}

revision 2016-03-01 {
   description
   "Initial version."
}

/*
 * Features
 */

feature bier-te-frr {
   description
   "Support ECMP feature in BIER TE."
}

grouping te-frr {
   description "The TE fast reroute information."
   list btaft {
      key "adj-index";
      description "The adjacency index of the frr paths."
      leaf adj-index {
         type uint32;
         mandatory true;
         description "The frr adjacency index."
      }
      leaf bitposition {
         type bit-string;
         mandatory true;
         description "The bitposition information."
      }
      leaf resetbitmask {
         type bit-string;
         mandatory true;
         description "The deleting bitmask of the forwarding item."
      }
      leaf addbitmask {
         type bit-string;
      }
   }
}
mandatory true;
description "The adding bitmask of the forwarding item.";
}
}
grouping te-adj-type {
  description "The collection of all possible adjacency type.";
  choice te-adj-type {
    mandatory true;
    case connected {
      description "The type of adjacency is connected. Mostly connected interfaces.";
    }
    case routed {
      description "The type of adjacency is routed. Mostly not connected interfaces.";
    }
    case local-decap {
      description "Means that the packet should be decapsulated and forward out BIER domain.";
    }
    case ecmp {
      description "There is more than one path in the adjacency with equal cost.";
    }
    case other {
      description "Means that the packet should be discarded.";
    }
  description "The collection of all possible adjacency type.";
  }
}
grouping te-adjID-type {
  description "The collection of all possible adjacency type.";
  choice te-adjID-type {
    mandatory true;
    case p2p {
      description "Describes p2p adjacency.";
    }
    case bfer {
      description "Describes bfer adjacency.";
    }
    case leaf-bfer {
      description "Describes leaf-bfer adjacency. There is no next BFR that the packet should be forwarded.";
    }
    case lan {
      description "Describes lan adjacency.";
    }
    case spoke {
      description "Describes spoke adjacency of hub-and spoke.";
    }
  }
}
case ring-clockwise {
    description "Describes clockwise adjacency in ring.";
}
case ring-counterclockwise {
    description "Describes counterclockwise adjacency in ring.";
}
case ecmp {
    description "Describes ecmp adjacency.";
}
case virtual-link {
    description "Describes virtual adjacency between two indirect connect nodes.";
}
case other {
    description "Describes other id type of adjacency.";
}
description "The collection of all possible adjacency type.";
}

typedef adjid {
    type uint32;
    description "The type for adjacency ID.";
}

typedef bit-string {
    type uint16;
    description "The bit mask of one bitstring.";
}

grouping te-adjID {
    list adj-id {
        key "adjID";
        description "This ID information of one adjacency.";
        leaf adjID {
            type adjid;
            mandatory true;
            description "The adjacency id.";
        }
        leaf adj-if {
            /* type if:if-index; */
            type uint32; /* for compilation */
            mandatory true;
            description "The corresponding interface of this adjacency.";
        }
        uses te-adjID-type;
    }
    description "This group presents adjacency ID information for BIER TE.";
}
grouping te-ecmp {
    description "The ecmp information.";
    list ecmp-path {
        key "index";
        description "The index of the ecmp paths.";
        leaf index {
            type uint32;
            mandatory true;
            description "The ecmp index.";
        }
        list number {
            key "number";
            description "The list of the ecmp paths.";
            leaf number {
                type uint16;
                mandatory true;
                description "The number of the ecmp paths.";
            }
            leaf out-if {
                type uint32; /* for compilation */
                mandatory true;
                description "The outgoing interface.";
            }
        }
    }
}

grouping te-items {
    description "The BIER TE forwarding items collection.";
    uses te-adj-type;
    leaf f-bm {
        type bit-string;
        mandatory true;
        description "The bitmask of the forwarding item.";
    }
    leaf f-intf {
        /* type if:if-index; */
        type uint32; /* for compilation */
        mandatory true;
        description "The out interface of this forwarding item.";
    }
    leaf ecmp {
        type boolean;
        description "The capability of ecmp paths.";
    }
    leaf ecmp-index {
        type uint16;
    }
}
description "The index of ecmp path.";
} leaf frr {
    type boolean;
    description "The capability of fast re-route.";
}
}

grouping te-fwd-item {
    list te-si {
        key "si";
        description "The forwarding items of one set identifier.";
        leaf si{
            type bier:si;
            mandatory true;
            description "The set identifier of this forwarding item.";
        }
    }
    list te-f-index {
        key "te-f-index";
        description "The forwarding information of one BIER TE item.";
        leaf te-f-index {
            type bit-string;
            mandatory true;
            description "The bit index of BIER TE forwarding item.";
        }
        uses te-items;
    }
    description "The forwarding items in one set identifier.";
}

grouping te-info {
    description "The BIER TE forwarding information.";
    list te-subdomain {
        key "subdomain-id";
        description "The forwarding items of one sub-domain.";
        leaf subdomain-id {
            type bier:sub-domain-id;
            description "The sub-domain-id of this sub-domain.";
        }
        uses te-adjID;
    }
    list te-bsl {
        key "fwd-bsl";
        description "The forwarding items in one bitstringlength.";
        leaf fwd-bsl {
            type uint16;
            description "The value of bitstringlength.";
        }
    }
}
uses te-fwd-item;
}
uses te-ecmp;
uses te-frr {
   if-feature bier-te-frr;
}
}

/*
 * Configuration data nodes
 */
augment "/rt:routing" {
    description "The BIER TE information.";
    container bier-te-config {
        description "The BIER TE information container.";
        uses te-info;
    }
}

/*
 * Operational state data nodes
 */
augment "/rt:routing-state" {
    description "BIER TE state information.";
    container bier-te-state {
        description "The BIER TE information in nodes.";
        uses te-info;
    }
}

/*
 * Notifications
 */
notification bier-te-notification {
    description "This notification is sent when a condition changes in BIER TE.";
    list adjID-is-zero {
        key "if-index";
        description "The adjacency id is zero.";
        leaf if-index {
            type uint32;
            description "The adjacency id of this interface is zero.";
        }
        uses te-adjID-type;
    }
}
6. Normative References

[I-D.eckert-bier-te-arch]

[I-D.eckert-bier-te-frr]

[I-D.ietf-bier-architecture]

[I-D.ietf-bier-bier-yang]


Authors' Addresses
