A YANG Data Model for Network Bridge Management
draft-vassilev-netmod-network-bridge-01

Abstract

This document introduces new YANG model of a network bridge.

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 https://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 July 11, 2019.

Copyright Notice

Copyright (c) 2019 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 (https://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.
1. Introduction

There is a need for a YANG model for management of network bridges. The model should allow the variety of existing forwarding and scheduling technologies to be defined as interoperable modules that can be interconnected and extended.

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].
1.1.1. YANG

The following terms are defined in [RFC7950]:

- must statement
- augment statement
- context node
- container
- data node
- key leaf
- leaf
- leaf-list
- list

1.1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.2. Problem Statement

This document attempts to address the problem of defining YANG model of a network bridge that can be used as common framework by different forwarding and scheduling implementations.

1.3. Solution

A Network bridge has more than 1 ingress and 1 or more egress ports. It has 1 or more traffic classes. The proposed model splits the design into 2 components - 1) Forwarding component and 2) Scheduling component. The forwarding component is connected to all ingress ports and forwards traffic from them to the scheduler instances connected to the egress ports. The scheduling component is a set of scheduler instances - topologies of interconnected aggregators and filters connected to a single egress port and as many as ingress_ports_count*traffic_class_count datapaths from the forwarding component.
1.3.1. Forwarding

The simple idea of creating a YANG model for a subset of the original [OpenFlow] specification is used as base for the model for management of the Forwarding Information Base (FIB) of the bridge.

1.3.2. Scheduling

The scheduler(s) have 1 or more input datapaths and 1 output. To each datapath the forwarding component can forward flows. Many different scheduler implementations have structure based on common modular abstractions flow meters, delay lines, queues, gates and gate control logic that determines the gate states based on variables defined in the flow meter, the delay line or the queue or signals and timers available to the gate control logic algorithm. The concept is illustrated with the following model of a 2 ingress ports, 2 traffic classes implementation of a 2 class strict priority scheduling bridge:
The common structure of a scheduler module (S) consisting of topology of consecutive flow-meters (M), gates (G) connected to a common gate control - (C) with a single egress port. A new module type representing delay line (D) is added to the structure of the scheduler before Q. The delay line (D) is important for time-sensitive scheduler models where propagation delays, store-and-forward delays and even programmable delays in some cases need to be represented. For certain time sensitive applications it is important to differentiate between different ports due to rate conversion, store and forward and other factors influencing the behavior of the bridge. This is why the concept of a port class is introduced in the model.
Depending on the scheduler design the ingress flows can specify different D and Q parameters e.g. D.time=0 means no delay, Q.len=0 means no buffering and immediate drop of packets in case the gate is closed. With the submodules collapsed to an integral generic gate controller module (GC) the diagram becomes much simpler.
Complex scheduler designs exist that can combine several different gate controllers into complex topology. This concept is demonstrated in the example bridge.

2. Network Bridge Module Tree Diagram

module: ietf-network-bridge
  +--rw bridge
    +--rw ports
      +--rw port* [name]
        +--rw name string
        +--rw index? uint64

    augment /if:interfaces/if:interface:
      +--rw port-name? -> /bridge/ports/port/name

3. Network Bridge Flows Module Tree Diagram

module: ietf-network-bridge-flows
  +--rw packet-in-message
    | +--rw packet-in-reason? identityref
    | +--rw ingress? netbr:port-ref
    | +--rw payload? binary
    | +--rw match
    |   | +--rw in-port? netbr:port-ref
    |   | +--rw ethernet-match
    |   |   | +--rw ethernet-source!
    |   |   |   | +--rw address yang:mac-address
    |   |   |   | +--rw mask? yang:mac-address
    |   |   | +--rw ethernet-destination!
    |   |   |   | +--rw address yang:mac-address
++-rw mask?     yang:mac-address
++-rw ethernet-type!
++-rw type      ether-type
++-rw vlan-match
    ++-rw vlan-id!
    | ++-rw vlan-id-present?   boolean
    | ++-rw vlan-id?           vlan-id
    ++-rw vlan-pcp?     vlan-pcp
++-rw flows
    ++-rw flow* [id]
    | ++-rw id               flow-id
++-rw match
    ++-rw in-port?          netbr:port-ref
    ++-rw ethernet-match
        ++-rw ethernet-source!
        | ++-rw address     yang:mac-address
        | ++-rw mask?      yang:mac-address
        ++-rw ethernet-destination!
        | ++-rw address     yang:mac-address
        | ++-rw mask?      yang:mac-address
        ++-rw ethernet-type!
        | ++-rw type      ether-type
    ++-rw vlan-match
    ++-rw vlan-id!
    | ++-rw vlan-id-present?   boolean
    | ++-rw vlan-id?           vlan-id
    ++-rw vlan-pcp?     vlan-pcp
++-rw actions
    ++-rw action* [order]
    | ++-rw order            int32
    | (action)?
        +++:(output-action-case)
        | ++-rw output-action
        |    ++-rw out-port?     netbr:port-ref
        |    ++-rw max-length?   uint16
        +++:(controller-action-case)
        | ++-rw controller-action
        |    ++-rw max-length?   uint16
        +++:(drop-action-case)
        | ++-rw drop-action!
        +++:(pop-vlan-action-case)
        | ++-rw pop-vlan-action!
        +++:(push-vlan-action-case)
        | ++-rw push-vlan-action
            | ++-rw ethernet-type?   uint16
            | ++-rw tag?             int32
            | ++-rw pcp?             int32
            | ++-rw cfi?             vlan-cfi
### RFC Draft: NETWORK-BRIDGE-01

#### Section: Action Cases

- **set-vlan-cfi-action-case**:
  - **set-vlan-cfi-action**
    - **vlan-cfi**

- **set-vlan-id-action-case**:
  - **set-vlan-id-action**
    - **vlan-id**

- **set-vlan-pcp-action-case**:
  - **set-vlan-pcp-action**
    - **vlan-pcp**

- **strip-vlan-action-case**:
  - **strip-vlan-action**

- **transmit-packet**:
  - **input**
    - **egress**
      - netbr:port-ref
    - **ingress**
      - netbr:port-ref
    - **payload**
      - binary
  - **action**
    - **order**
      - int32

- **output-action-case**:
  - **output-action**
    - **out-port**
      - netbr:port-ref
    - **max-length**
      - uint16

- **controller-action-case**:
  - **controller-action**
    - **max-length**
      - uint16

- **drop-action-case**:
  - **drop-action**

- **pop-vlan-action-case**:
  - **pop-vlan-action**

- **push-vlan-action-case**:
  - **push-vlan-action**
    - **ethernet-type**
      - uint16
    - **tag**
      - int32
    - **pcp**
      - int32
    - **cfi**
      - vlan-cfi
    - **vlan-id**
      - vlan-id

- **set-vlan-cfi-action-case**:
  - **set-vlan-cfi-action**
    - **vlan-cfi**

- **set-vlan-id-action-case**:
  - **set-vlan-id-action**
    - **vlan-cfi**

- **priority**
  - uint16

- **container-name**
  - string

- **flow-name**
  - string
4. Network Bridge Scheduler Module Tree Diagram

module: ietf-network-bridge-scheduler
augment /flow:flows/flow:flow:
  +--rw traffic-class?
    -> /netbr:bridge/sched:traffic-classes/traffic-class
augment /netbr:bridge/netbr:ports/netbr:port:
  +--rw class?
  +--rw class-instance-index?  uint32
augment /netbr:bridge:
  +--rw default-traffic-class?  traffic-class-ref
  +--rw default-port-class?  traffic-class-ref
  +--rw traffic-classes
    |  +--rw traffic-class*  identityref
  +--rw port-classes
    |  +--rw port-class*  identityref
augment /if:interfaces/if:interface:
  +--rw scheduler
    +--rw gate-controllers
+-rw gate-controller* [id type]
  +-rw id               string
  +-rw type             identityref
  +-rw inputs
     |  +-rw input* [class index]
     |     |  +-rw class    identityref
     |     |  +-rw index   uint32
     |     |  +-ro queued-pkts?  uint64
     |     |  +-ro queued-bytes? uint64
     |     |  +-ro discards?   uint64
     |     |  +-ro overflow-discards?  uint64
     |     |  +-ro error-discards? uint64
     |  +-rw input-classes
     |     |  +-rw input-class* [class]
     |     |     |  +-rw class    identityref
     |     |     |  +-ro queued-pkts?  uint64
     |     |     |  +-ro queued-bytes? uint64
     |     |     |  +-ro discards?   uint64
     |     |     |  +-ro overflow-discards?  uint64
     |     |     |  +-ro error-discards? uint64
     |  augment /netbr:bridge:

+-rw scheduler-classes
  +-rw scheduler-class* [egress-port-class]
     |  +-rw egress-port-class    sched:port-class-ref
  +-rw inputs
     |  +-rw input* [traffic-class ingress-port-class]
     |     |  +-rw traffic-class    traffic-class-ref
     |     |  +-rw ingress-port-class    port-class-ref
     |     |  +-rw gate-controller?    leafref
     |     |  +-rw input-class?         leafref
     |     |  +-rw base-index?          uint32
  +-rw gate-controllers
     |  +-rw gate-controller* [id]
     |     |  +-rw id               string
     |     |  +-rw type             identityref
     |  +-rw inputs
     |     |  +-rw input* [class]
     |     |     |  +-rw class    identityref
     |     |     |  +-rw instance-count?  uint32
     |     |     |  +-rw constant-propagation-delay?  uint64
     |     |     |  +-rw configurable-delay-line?  uint64
     |     |     |  +-rw queue-len?  uint32
  +-rw output
     |  +-rw gate-controller?
     |     |  -> ../../../gate-controller/id
     |  +-rw input-class?  leafref
     |  +-rw index?       uint32
5. Network Bridge Module YANG

<CODE BEGINS> file "ietf-network-bridge@2018-07-15.yang"

module ietf-network-bridge {
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-bridge";
  prefix netbr;

  import ietf-interfaces {
    prefix if;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/netmod/>
    WG List:  <mailto:netmod@ietf.org>
    Editor:  Vladimir Vassilev
      <mailto:vladimir@transpacket.com>";
  description
    "This module contains a collection of YANG definitions for
    description and management of network bridges.

    Copyright (c) 2015 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust’s Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision 2018-07-15 {
    description
      "Initial revision.";
    reference "RFC XXXX: Network Bridge";
  }

  container bridge {
    container ports {
      list port {
        key "name";
        unique "index";
      }
leaf name {
  type string;
}
leaf index {
  type uint64;
}

augment "/if:interfaces/if:interface" {
  leaf port-name {
    type leafref {
      path "/netbr:bridge/netbr:ports/netbr:port/netbr:name";
    }
  }
}

typedef port-ref {
  type leafref {
    path "/if:interfaces/if:interface/netbr:port-name";
  }
  description
    "This type is used by data models that need to reference configured bridge ports.";
}

6. Network Bridge Flows Module YANG

<CODE BEGINS> file "ietf-network-bridge-flows@2018-07-15.yang"

module ietf-network-bridge-flows {
  prefix flow;

import ietf-network-bridge {
  prefix netbr;
}
import ietf-inet-types {
  prefix inet;
  revision-date 2013-07-15;
}
import ietf-yang-types {
  prefix yang;
  revision-date 2013-07-15;
}
This module contains a collection of YANG definitions for description and management of network bridge based on flows.

revision 2018-07-15 {
  description
    "Unreleased revision.";
  reference "RFC XXXX: Network Bridge";
}

typedef vlan-cfi {
  type int32;
}

grouping address {
  choice address {
    case ipv4 {
      leaf ipv4-address {
        type inet:ipv4-prefix;
      }
    }
    case ipv6 {
      leaf ipv6-address {
        type inet:ipv6-prefix;
      }
    }
  }
}

grouping ordered {
  leaf order {
    type int32;
  }
}

grouping action-list {
  list action {
    key "order";
  }
}
uses ordered;
uses action;

grouping action {
    choice action {
        case output-action-case {
            container output-action {
                leaf out-port {
                    type netbr:port-ref;
                }
                leaf max-length {
                    type uint16;
                }
            }
        }
        case controller-action-case {
            container controller-action {
                leaf max-length {
                    type uint16;
                }
            }
        }
        case drop-action-case {
            container drop-action {
                presence "no content action";
            }
        }
        case pop-vlan-action-case {
            container pop-vlan-action {
                presence "no content action";
            }
        }
        case push-vlan-action-case {
            container push-vlan-action {
                leaf ethernet-type {
                    type uint16;
                }
                leaf tag {
                    type int32;
                }
                leaf pcp {
                    type int32;
                }
                leaf cfi {
                    type vlan-cfi;
                }
            }
        }
    }
}

Vassilev                  Expires July 11, 2019                [Page 15]
leaf vlan-id {
    type vlan-id;
}
}
case set-vlan-cfi-action-case {
    container set-vlan-cfi-action {
        leaf vlan-cfi {
            type vlan-cfi;
        }
    }
}
case set-vlan-id-action-case {
    container set-vlan-id-action {
        leaf vlan-id {
            type vlan-id;
        }
    }
}
case set-vlan-pcp-action-case {
    container set-vlan-pcp-action {
        leaf vlan-pcp {
            type vlan-pcp;
        }
    }
}
case strip-vlan-action-case {
    container strip-vlan-action {
        presence "no content action";
    }
}
}
typedef vlan-pcp {
    description
        "IEEE 802.1p priority. It indicates the frame priority level.
        Values are from 0 (best effort) to 7 (highest);
        1 represents the lowest priority.";
    type uint8 {
        range "0..7";
    }
}
typedef vlan-id {
    type uint16 {
        range "0..4095";
    }
}
typedef ether-type {
    type uint32;
}

grouping mac-address-filter {
    leaf address {
        mandatory true;
        type yang:mac-address;
    }
    leaf mask {
        type yang:mac-address;
    }
}

grouping ethernet-match-fields {
    container ethernet-source {
        description
        "Ethernet source address.";
        presence "Match field is active and set";
        uses mac-address-filter;
    }
    container ethernet-destination {
        description
        "Ethernet destination address.";
        presence "Match field is active and set";
        uses mac-address-filter;
    }
    container ethernet-type {
        description
        "Ethernet frame type.";
        presence "Match field is active and set";
        leaf type {
            mandatory true;
            type ether-type;
        }
    }
}

grouping vlan-match-fields {
    container vlan-id {
        description
        "VLAN id.";
        presence "Match field is active and set";
        leaf vlan-id-present {
            type boolean;
        }
    }
}
leaf vlan-id {
    type vlan-id;
}

leaf vlan-pcp {
    description
        "VLAN priority.";
    type vlan-pcp;
}

grouping match {
    leaf in-port {
        type netbr:port-ref;
    }
    container ethernet-match {
        uses ethernet-match-fields;
    }
    container vlan-match {
        uses vlan-match-fields;
    }
}

grouping raw-packet {
    description
        "Basic packet structure.";
    leaf ingress {
        type netbr:port-ref;
    }
    leaf payload {
        type binary;
    }
}

grouping packet-in {
    leaf packet-in-reason {
        type identityref {
            base packet-in-reason;
        }
        uses raw-packet;
    }
}

grouping ethernet-packet {
    description
        "Ethernet packet headers structure.";
    leaf source {
        type yang:mac-address;
    }
}
leaf destination {
    type yang:mac-address;
}

identity packet-in-reason {
    description
        "Base identity for all the available packet in reasons."
}

identity no-match {
    base packet-in-reason;
    description
        "No matching flow in the classifier"
}

identity send-to-controller {
    base packet-in-reason;
    description
        "Explicit instruction to send packet to controller"
}

identity invalid-ttl {
    base packet-in-reason;
    description
        "Packet with invalid TTL"
}

notification packet-received {
    description
        "Delivery of incoming packet."
    uses packet-in;
    container match {
        uses match;
    }
}

rpc transmit-packet {
    description
        "Sending packet out."
    input {
        leaf egress {
            type netbr:port-ref;
        }
        uses raw-packet;
        uses action-list;
    }
}
container packet-in-message {
    uses packet-in;
    container match {
        uses match;
    }
}

grouping flow {
    container match {
        uses match;
    }
    container actions {
        uses action-list;
    }
    leaf priority {
        type uint16;
    }
    leaf container-name {
        type string;
    }
    leaf flow-name {
        type string;
    }
}

typedef flow-id {
    type inet:uri;
}

container flows {
    list flow {
        key "id";
        leaf id {
            type flow-id;
        }
        uses flow;
    }
}

7. Network Bridge Scheduler Module YANG

<CODE BEGINS> file "ietf-network-bridge-scheduler@2019-01-07.yang"

module ietf-network-bridge-scheduler {

Vassilev                        Expires July 11, 2019     [Page 20]
prefix sched;

import ietf-network-bridge {
    prefix netbr;
}
import ietf-network-bridge-flows {
    prefix flow;
}
import ietf-interfaces {
    prefix if;
}

organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
    WG List:  <mailto:netmod@ietf.org>
    Editor:   Vladimir Vassilev
                <mailto:vladimir@transpacket.com>";
description
    "This module contains a collection of YANG definitions for
description and management of network bridge schedulers.

Copyright (c) 2019 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents
(http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.";

revision 2019-01-07 {
    description
        "Initial revision.";
    reference "RFC XXXX: Network Bridge";
}

identity gate-controller {
    description
        "Represents the gate control block type e.g. round-robin,
priority-based, time-aware-802dot1q-bv etc.";
}
identity aggregator {
    base gate-controller;
    description
    "Abstract identity that all gate control blocks with multiple
    inputs and single output use as basetype e.g. round-robin,
    priority-based, time-aware-802dot1qbv etc.";
}

identity filter {
    base gate-controller;
    description
    "Abstract identity that all gate control blocks with corresponding
    input and output instances use as basetype e.g. rate-limiters,
    simple propagation delays, shapers etc.";
}

identity gate-controller-input {
    description
    "Identifies gate controller input type.";
}

identity private-queue-aggregator-input {
    base gate-controller-input;
    description
    "Abstract input identifier for gate controller
    inputs of the aggregator type where all
    instances of the input types derived from
    this identifier have their own private queue.";
}

identity shared-queue-aggregator-input {
    base gate-controller-input;
    description
    "Abstract input identifier for gate controller
    inputs of the aggregator type where all
    instances of the input types derived from
    this identifier have shared queue.";
}

identity filter-input {
    base gate-controller-input;
    description
    "Abstract input identifier for gate controller
    inputs of the filter type.";
}

identity traffic-class {
    description
"Identifies traffic class."
}

identity port-class {
  description
  "Identifies port class. Ports that belong to a class
  will have the same scheduler-class on their egress
  and have identical flow path through the rest of
  the scheduler classes."
}

typedef port-class-ref {
  type leafref {
    path "/netbr:bridge/sched:port-classes/sched:port-class";
  }
  description
  "This type is used by data models that need to reference
  configured port-class.";
}

typedef traffic-class-ref {
  type leafref {
    path "/netbr:bridge/sched:traffic-classes/sched:traffic-class";
  }
  description
  "This type is used by data models that need to reference
  configured traffic-class.";
}

augment "/flow:flows/flow:flow" {
  leaf traffic-class {
    type leafref {
      path "/netbr:bridge/sched:traffic-classes/sched:traffic-class";
    }
    description
    "Specifies the traffic class of a flow.
    When not present the default traffic class is used.";
  }
}

augment "/netbr:bridge/netbr:ports/netbr:port" {
  leaf class {
    type port-class-ref;
  }
  leaf class-instance-index {
    type uint32;
  }
}

augment "/netbr:bridge" {
leaf default-traffic-class {
  type traffic-class-ref;
  description
    "Specifies the traffic-class for flows without
    /flow:flows/flow:flow/sched:traffic-class leaf.";
}
leaf default-port-class {
  type traffic-class-ref;
  description
    "Specifies the traffic-class for flows without
    /flow:flows/flow:flow/sched:traffic-class leaf.";
}
container traffic-classes {
  leaf-list traffic-class {
    type identityref {
      base traffic-class;
    }
  }
}
container port-classes {
  leaf-list port-class {
    type identityref {
      base port-class;
    }
  }
}

grouping gate-controller-input-config {
  leaf constant-propagation-delay {
    type uint64;
    description
      "Constant delay attributed to delays in the gate-controller.";
    units "picoseconds";
  }
  leaf configurable-delay-line {
    type uint64;
    description
      "Some gate controllers can delay the flow of packets with
      configurable delay which is added to the constant
      propagation-delay. Only inputs with zero queue lengths
      have deterministic delays equal to the sum of the
      constant-propagation-delay and the configurable-delay-line
      leaves. Inputs with queues have variable higher delay with
      dynamic component based on the controllers logic.";
    units "picoseconds";
  }
  leaf queue-len {
type uint32;
units "bytes";
}
}

grouping gate-controller-queue-state {
  leaf queued-pkts {
    config false;
    type uint64;
  }
  leaf queued-bytes {
    config false;
    type uint64;
  }
  leaf discards {
    config false;
    type uint64;
    description
    "The total number of discarded packets that were received on this input. This includes but is not limited to the overflow-discards. For example gate-controllers can start discarding certain packets before the input queue is filled. These discards are not registered as overflow-discards.

    The lower 32 bits of the sum of all discards counters part of a scheduler are equal to the /if:interfaces/if:interface/if:statistics/if:out-discards counter for the corresponding interface.";
  }
  leaf overflow-discards {
    config false;
    type uint64;
    description
    "Unintended discard caused by overflow of the input queue of the gate controller.";
  }
  leaf error-discards {
    config false;
    type uint64;
  }
}

augment "/if:interfaces/if:interface" {
  container scheduler {
    container gate-controllers {
      list gate-controller {
        key "id";
      }
    }
  }
}
leaf id {
  type string;
}
leaf type {
  type identityref {
    base gate-controller;
  }
  mandatory true;
}
container inputs {
  list input {
    key "class index";
    leaf class {
      type identityref {
        base gate-controller-input;
      }
    }
    leaf index {
      type uint32;
    }
    uses gate-controller-queue-state;
  }
}
container input-classes {
  list input-class {
    key "class";
    leaf class {
      type identityref {
        base gate-controller-input;
      }
    }
    uses gate-controller-queue-state;
  }
}
}
}
augment "/netbr:bridge" {
  container scheduler-classes {
    list scheduler-class {
      key "egress-port-class";
      leaf egress-port-class {
        type sched:port-class-ref;
      }
    }
    container inputs {
      list input {
        key "traffic-class ingress-port-class";
      }
    }
  }
}
leaf traffic-class {
   type traffic-class-ref;
}
leaf ingress-port-class {
   type port-class-ref;
}
leaf gate-controller {
   type leafref {
      path "../../../gate-controllers/gate-controller/id";
   }
}
leaf input-class {
   type leafref {
      path "../../../gate-controllers/gate-controller" + "/[id=current()]/../gate-controller" + "/inputs/input/class";
   }
}
leaf base-index {
   type uint32;
   default "0";
}
}
}
container gate-controllers {
   list gate-controller {
      key "id";
      leaf id {
         type string;
      }
      leaf type {
         type identityref {
            base gate-controller;
         }
         mandatory true;
      }
   }
   container inputs {
      list input {
         key "class";
         leaf class {
            type identityref {
               base gate-controller-input;
            }
            mandatory true;
         }
         leaf instance-count {
            type uint32;
         }
      }
   }
}

Vassilev                  Expires July 11, 2019                [Page 27]
uses gate-controller-input-config;
}
}
} container output {
leaf gate-controller {
  type leafref {
    path "../../../gate-controller/id";
  }
}
leaf input-class {
  type leafref {
    path "../../../gate-controller"
    + "[id=current()]/../gate-controller]/"
    + "inputs/input/class";
  }
}
leaf index {
  type uint32;
  description "In case the gate-controller is aggregator this is the index of the only input it is connected to from the specified class. If the gate-controller is filter with more then one input-output pairs this is the base index and the remaining indexes are connected to consecutive input indexes of the specified input class.";
}
}
}
}
}
}
}
}

<CODE ENDS>

8. IANA Considerations

8.1. NETWORK BRIDGE YANG Modules

This document registers 3 YANG modules in the YANG Module Names registry [RFC7950].
9. Security Considerations

This document does not introduce any new security concerns in addition to those specified in [RFC7950], section 15.

10. Normative References

[OpenFlow]
"Open Networking Foundation", "OpenFlow Switch Specification", December 2009,


Appendix A. Example

Example bridge with signaling, video0, video1 and best-effort traffic classes.

A.1. Model

<CODE BEGINS> file "example-bridge@2018-07-15.yang"

module example-bridge {
  yang-version 1.1;
  namespace "http://example.com/ns/example-bridge";
  prefix example;

  import ietf-network-bridge {
    prefix netbr;
  }
  import ietf-network-bridge-scheduler {
    prefix sched;
  }

  organization "example.com";
  description "Example of bridge."

  revision 2018-07-15 {
    description "Initial."
  }

}
identity video0 {
    base sched:traffic-class;
}

identity video1 {
    base sched:traffic-class;
}

identity signaling {
    base sched:traffic-class;
}

identity best-effort {
    base sched:traffic-class;
}

identity default-port {
    base sched:port-class;
}

//Strict priority aggregator with 3 classes:
identity strict-priority-aggregator {
    base sched:aggregator;
}

identity pri0 {
    base sched:shared-queue-aggregator-input;
    base strict-priority-aggregator;
}

identity pri1 {
    base sched:shared-queue-aggregator-input;
    base strict-priority-aggregator;
}

identity pri2 {
    base sched:shared-queue-aggregator-input;
    base strict-priority-aggregator;
}

//Cyclic timeslot schedule aggregator with 2 timeslots:
identity cyclic-timeslot-schedule-aggregator {
    base sched:aggregator;
}

identity timeslot0 {
    base sched:shared-queue-aggregator-input;
base cyclic-timeslot-schedule-aggregator;
}

identity timeslot1 {
    base sched:shared-queue-aggregator-input;
    base cyclic-timeslot-schedule-aggregator;
}

augment "/netbr:bridge/sched:scheduler-classes/sched:scheduler-class"
   + "/sched:gate-controllers/sched:gate-controller" {
    when "./sched:type = ‘example:cyclic-timeslot-schedule-aggregator’";
    leaf period {
        type uint32;
        units "nanoseconds";
    }
    leaf time-slot0-interval {
        type uint32;
        units "nanoseconds";
    }
    leaf time-slot1-interval {
        type uint32;
        units "nanoseconds";
    }
}

//Rate limiter - filter:
identity rate-limiter {
    base sched:filter;
}

identity in {
    base sched:filter-input;
    base rate-limiter;
}

augment "/netbr:bridge/sched:scheduler-classes/sched:scheduler-class"
   + "/sched:gate-controllers/sched:gate-controller" {
    when "./sched:type = ‘example:rate-limiter’";
    leaf interval {
        type uint32;
        units "nanoseconds";
    }
    leaf limit {
        type uint32;
        units "octets";
    }
}
A.2. Scheduler diagram

The scheduler topology and the gate controller instances are specified in the operational configuration data that can be modified or not depending on the underlying implementation. The single letter identifiers for the gate-controllers have the following identities:

- r1, r2 - rate limiter instances
- a - trivial aggregator instance (implemented using strict-priority-aggregator)
- t - cyclic-timeslot-schedule-aggregator instance
- p - strict-priority-aggregator instance

A.3. Topology

The example flow configuration is for the topology in the diagram below.
A.4. CLI listing

CLI commands configuring flows and assigning flows to traffic-classes:
> create /flows/flow[id='video0'] -- \
  match/vlan-match/vlan-id/vlan-id=10 \n  actions/action[order='0']/output-action/out-port=p2
> merge /flows/flow[id='video0'] -- traffic-class=video0
> create /flows/flow[id='video1'] -- \
  match/vlan-match/vlan-id/vlan-id=11 \n  actions/action[order='0']/output-action/out-port=p2
> merge /flows/flow[id='video1'] -- traffic-class=video1
> create /flows/flow[id='best-effort-to-host0'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:00 \n  actions/action[order='0']/output-action/out-port=p0
> merge /flows/flow[id='best-effort-to-host0'] -- traffic-class=best-effort
> create /flows/flow[id='best-effort-to-host1'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:01 \n  actions/action[order='0']/output-action/out-port=p1
> merge /flows/flow[id='best-effort-to-host1'] -- traffic-class=best-effort
> create /flows/flow[id='best-effort-to-host2'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:02 \n  actions/action[order='0']/output-action/out-port=p2
> merge /flows/flow[id='best-effort-to-host2'] -- traffic-class=best-effort
> create /flows/flow[id='ptp-to-host0'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:00 \n  actions/action[order='0']/output-action/out-port=p0
> merge /flows/flow[id='ptp-to-host0'] -- traffic-class=signaling
> create /flows/flow[id='ptp-to-host1'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:01 \n  actions/action[order='0']/output-action/out-port=p1
> merge /flows/flow[id='ptp-to-host1'] -- traffic-class=signaling
> create /flows/flow[id='ptp-to-host2'] -- \
  match/ethernet-match/ethernet-destination\n/address=00:01:02:03:00:02 \n  actions/action[order='0']/output-action/out-port=p2
> merge /flows/flow[id='ptp-to-host2'] -- traffic-class=signaling
> commit

CLI commands configuring and monitoring the scheduler:
> replace /bridge/scheduler-classes/scheduler-class/gate-controllers/
>       /gate-controller[id='p']/inputs/input/queue-len value=1048576
> replace /bridge/scheduler-classes/scheduler-class/gate-controllers/
>       /gate-controller[id='t']/time-slot0-interval value=5000000
> commit
> xget /interfaces/interface[name='if2']/scheduler/gate-controllers/
>       /gate-controller[id='r1']/inputs/input[index='1']/overflow-discards
... overflow-discards 33 ...
> xget /interfaces/interface[name='if2']/scheduler/gate-controllers/
>       /gate-controller[id='p']/input-classes/
>           input-class[class='pri2']/overflow-discards
... overflow-discards 1000000 ...

A.5. Configuration Data Instance

<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <bridge xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge">
    <ports>
      <port>
        <name>p0</name>
        <index>0</index>
        <class
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
          xmlns:example="http://example.com/ns/example-bridge">
          example:default-port</class>
        <class-instance-index
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
          0</class-instance-index>
      </port>
      <port>
        <name>p1</name>
        <index>1</index>
        <class
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
          xmlns:example="http://example.com/ns/example-bridge">
          example:default-port</class>
        <class-instance-index
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
          1</class-instance-index>
      </port>
      <port>
        <name>p2</name>
        <index>2</index>
        <class
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
          xmlns:example="http://example.com/ns/example-bridge">
          example:default-port</class>
        <class-instance-index
          xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
          2</class-instance-index>
    </ports>
  </bridge>
</config>
<class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
xmns:example="http://example.com/ns/example-bridge">
  example:default-port</class>
</class-instance-index>
</port>
</ports>
<default-traffic-class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
xmns:example="http://example.com/ns/example-bridge">
  example:best-effort</default-traffic-class>
<default-port-class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
xmns:example="http://example.com/ns/example-bridge">
  example:default-port</default-port-class>
<traffic-classes
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
  <traffic-class
    xmlns:example="http://example.com/ns/example-bridge">
    example:best-effort</traffic-class>
  <traffic-class
    xmlns:example="http://example.com/ns/example-bridge">
    example:signaling</traffic-class>
  <traffic-class
    xmlns:example="http://example.com/ns/example-bridge">
    example:video0</traffic-class>
  <traffic-class
    xmlns:example="http://example.com/ns/example-bridge">
    example:video1</traffic-class>
</traffic-classes>
<port-classes
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
  <port-class
    xmlns:example="http://example.com/ns/example-bridge">
    example:default-port</port-class>
</port-classes>
</scheduler-classes>
</scheduler-class>
<egress-port-class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">
  example:default-port</egress-port-class>
</scheduler-class>
</scheduler-class>
</scheduler-class>
</scheduler-class>
<traffic-class xmlns:example="http://example.com/ns/example-bridge">
  example:best-effort
</traffic-class>

<ingress-port-class xmlns:example="http://example.com/ns/example-bridge">
  example:default-port
</ingress-port-class>

<gate-controller>p</gate-controller>

<input-class xmlns:example="http://example.com/ns/example-bridge">
  example:pri2
</input-class>

<base-index>0</base-index>

</input>

<input>
  <traffic-class xmlns:example="http://example.com/ns/example-bridge">
    example:signaling
  </traffic-class>
  <ingress-port-class xmlns:example="http://example.com/ns/example-bridge">
    example:default-port
  </ingress-port-class>
  <gate-controller>r1</gate-controller>
  <input-class xmlns:example="http://example.com/ns/example-bridge">
    example:in
  </input-class>
  <base-index>0</base-index>
</input>

<input>
  <traffic-class xmlns:example="http://example.com/ns/example-bridge">
    example:video0
  </traffic-class>
  <ingress-port-class xmlns:example="http://example.com/ns/example-bridge">
    example:default-port
  </ingress-port-class>
  <gate-controller>t</gate-controller>
  <input-class xmlns:example="http://example.com/ns/example-bridge">
    example:timeslot0
  </input-class>
  <base-index>0</base-index>
</input>

<input>
  <traffic-class xmlns:example="http://example.com/ns/example-bridge">
    example:video1
  </traffic-class>
  <ingress-port-class xmlns:example="http://example.com/ns/example-bridge">
    example:default-port
  </ingress-port-class>
  <gate-controller>t</gate-controller>
  <input-class xmlns:example="http://example.com/ns/example-bridge">

Vassilev

Expires July 11, 2019

[Page 38]
```xml
<example:timeslot1
    <base-index>0</base-index>
</input>
</inputs>
<gate-controllers>
    <gate-controller>
        <id>a</id>
        <type
            xmlns:example="http://example.com/ns/example-bridge">
            example:strict-priority-aggregator</type>
        <inputs>
            <input>
                <class
                    xmlns:example="http://example.com/ns/example-bridge">
                    example:pri0</class>
                <instance-count>3</instance-count>
                <queue-len>2048</queue-len>
            </input>
        </inputs>
    </gate-controller>
    <gate-controller>
        <id>p</id>
        <type
            xmlns:example="http://example.com/ns/example-bridge">
            example:strict-priority-aggregator</type>
        <inputs>
            <input>
                <class
                    xmlns:example="http://example.com/ns/example-bridge">
                    example:pri0</class>
                <instance-count>1</instance-count>
                <queue-len>2048</queue-len>
            </input>
            <input>
                <class
                    xmlns:example="http://example.com/ns/example-bridge">
                    example:pri1</class>
                <instance-count>1</instance-count>
                <queue-len>32768</queue-len>
            </input>
        </inputs>
    </gate-controller>
    <output>
        <gate-controller>r2</gate-controller>
        <input-class
            xmlns:example="http://example.com/ns/example-bridge">
            example:in</input-class>
        <index>0</index>
    </output>
</gate-controllers>
```
<example:in class="example:in" instance-count="3" queue-len="1048576"/>
</input>
</gate-controller>
<gate-controller id="r1" type="example:rate-limiter">
<input class="example:in" instance-count="3"/>
</input>
</inputs>
<output>
<gate-controller>a</gate-controller>
<input-class class="example:in" index="0"/>
</input>
</output>
<gate-controller id="r2" type="example:rate-limiter">
<input class="example:in" instance-count="1"/>
</input>
</inputs>
<output>
<gate-controller>p</gate-controller>
<input-class class="example:in"/>
</output>

<example:in class="example:in" instance-count="3" queue-len="1048576"/>
</input>
</gate-controller>
<gate-controller id="r1" type="example:rate-limiter">
<input class="example:in" instance-count="3"/>
</input>
</inputs>
<output>
<gate-controller>a</gate-controller>
<input-class class="example:in" index="0"/>
</input>
</output>
<gate-controller id="r2" type="example:rate-limiter">
<input class="example:in" instance-count="1"/>
</input>
</inputs>
<output>
<gate-controller>p</gate-controller>
<input-class class="example:in"/>
</output>
example:pri0</input-class>
</output>
<interval xmlns="http://example.com/ns/example-bridge">
10000000</interval>
<limit xmlns="http://example.com/ns/example-bridge">
125000</limit>
</gate-controller>
<gate-controller>
<id>t</id>
<type xmlns:example="http://example.com/ns/example-bridge">
example:cyclic-timeslot-schedule-aggregator</type>
<inputs>
<input>
<class xmlns:example="http://example.com/ns/example-bridge">
example:timeslot0</class>
<instance-count>3</instance-count>
<queue-len>1048576</queue-len>
</input>
<input>
<class xmlns:example="http://example.com/ns/example-bridge">
example:timeslot1</class>
<instance-count>3</instance-count>
<queue-len>1048576</queue-len>
</input>
</inputs>
<output>
<gate-controller>p</gate-controller>
<input-class xmlns:example="http://example.com/ns/example-bridge">
example:pri0</input-class>
</output>
</gate-controllers>
</scheduler-classes>
</bridge>
<flows xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-flows">
  <flow>
    <id>best-effort-to-host0</id>
    <match>
      <ethernet-match>
        <ethernet-destination>
          <address>00:01:02:03:00:00</address>
        </ethernet-destination>
      </ethernet-match>
    </match>
    <actions>
      <action>
        <order>0</order>
        <output-action>
          <out-port>p0</out-port>
        </output-action>
      </action>
    </actions>
    <traffic-class
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
      xmlns:example="http://example.com/ns/example-bridge">
      example:best-effort
    </traffic-class>
  </flow>
  <flow>
    <id>best-effort-to-host1</id>
    <match>
      <ethernet-match>
        <ethernet-destination>
          <address>00:01:02:03:00:01</address>
        </ethernet-destination>
      </ethernet-match>
    </match>
    <actions>
      <action>
        <order>0</order>
        <output-action>
          <out-port>p1</out-port>
        </output-action>
      </action>
    </actions>
    <traffic-class
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
      xmlns:example="http://example.com/ns/example-bridge">
      example:best-effort
    </traffic-class>
  </flow>
  <flow>
    <id>best-effort-to-host2</id>
    <match>
      <ethernet-match>
        <ethernet-destination>
          <address>00:01:02:03:00:02</address>
        </ethernet-destination>
      </ethernet-match>
    </match>
    <actions>
      <action>
        <order>0</order>
        <output-action>
          <out-port>p2</out-port>
        </output-action>
      </action>
    </actions>
    <traffic-class
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
      xmlns:example="http://example.com/ns/example-bridge">
      example:best-effort
    </traffic-class>
  </flow>
</flows>
<ethernet-match>
  <ethernet-destination>
    <address>00:01:02:03:00:02</address>
  </ethernet-destination>
</ethernet-match>
</match>
<actions>
  <action>
    <order>0</order>
    <output-action>
      <out-port>p2</out-port>
    </output-action>
  </action>
</actions>
<traffic-class
  xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
  xmlns:example="http://example.com/ns/example-bridge">
  example:best-effort
</traffic-class>
</flow>

<flow>
  <id>ptp-to-host0</id>
  <match>
    <ethernet-match>
      <ethernet-destination>
        <address>00:01:02:03:00:00</address>
      </ethernet-destination>
    </ethernet-match>
  </match>
  <actions>
    <action>
      <order>0</order>
      <output-action>
        <out-port>p0</out-port>
      </output-action>
    </action>
  </actions>
  <traffic-class
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
    xmlns:example="http://example.com/ns/example-bridge">
    example:signaling
  </traffic-class>
</flow>

<flow>
  <id>ptp-to-host1</id>
  <match>
    <ethernet-match>
      <ethernet-destination>
        <address>00:01:02:03:00:01</address>
      </ethernet-destination>
    </ethernet-match>
  </match>
</flow>
</ethernet-match>
</match>
</actions>

<flow>
  <id>ptp-to-host2</id>
  <match>
    <ethernet-match>
      <ethernet-destination>
        <address>00:01:02:03:00:02</address>
      </ethernet-destination>
    </ethernet-match>
  </match>
  <actions>
    <action>
      <order>0</order>
      <output-action>
        <out-port>p2</out-port>
      </output-action>
    </action>
  </actions>
  <traffic-class
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
    xmlns:example="http://example.com/ns/example-bridge">
    example:signaling
  </traffic-class>
</flow>

<flow>
  <id>video0</id>
  <match>
    <vlan-match>
      <vlan-id>
        <vlan-id>10</vlan-id>
      </vlan-id>
    </vlan-match>
  </match>
  <actions>
    <action>
      <order>0</order>
      <output-action>
        <out-port>p2</out-port>
      </output-action>
    </action>
  </actions>
  <traffic-class
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
    xmlns:example="http://example.com/ns/example-bridge">
    example:signaling
  </traffic-class>
</flow>
<order>0</order>
<output-action>
  <out-port>p2</out-port>
</output-action>
</action>
</actions>
<traffic-class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
xmlns:example="http://example.com/ns/example-bridge">
  example:video0
</traffic-class>
</flow>

<flow>
  <id>video1</id>
  <match>
    <vlan-match>
      <vlan-id>
        <vlan-id>11</vlan-id>
      </vlan-id>
    </vlan-match>
  </match>
  <actions>
    <action>
      <order>0</order>
      <output-action>
        <out-port>p2</out-port>
      </output-action>
    </action>
  </actions>
  <traffic-class
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
xmlns:example="http://example.com/ns/example-bridge">
    example:video1
  </traffic-class>
</flow>
</flows>

<interfaces
xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
  <interface>
    <name>if0</name>
    <type
xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
      ianaift:ethernetCsmacd</type>
    <port-name
xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge">
      p0</port-name>
  </interface>
  <interface>
    <name>if1</name>
    <type
xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
      ianaift:ethernetCsmacd</type>
  </interface>
</interfaces>
<port-name xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge">
  p1</port-name>
</interface>

<interface>
  <name>if2</name>
  <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    ianaift:ethernetCsmacd</type>
  <port-name xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge">
    p2</port-name>
</interface>
</interfaces>

<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
</nacm>
</config>

A.6. Companion YANG Data Model for Implementations Not Compliant with NMDA

The YANG modules defined in this document are designed to be used in conjunction with implementations that support the Network Management Datastore Architecture (NMDA) as defined in [RFC8342]. In order to allow implementations to use the data model even in cases when NMDA is not supported, the following companion module is defined.

<CODE BEGINS> file "ietf-network-bridge-scheduler-state@2019-01-07.yang"

module ietf-network-bridge-scheduler-state {

  prefix sched-state;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-network-bridge-scheduler {
    prefix sched;
  }

  organization "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact "WG Web:  <http://tools.ietf.org/wg/netmod/>
          WG List:  <mailto:netmod@ietf.org>"
This module contains /if:interfaces-state/if:interface augmentation which mirrors the ‘scheduler’ container as the one part of the 'ietf-network-bridge-scheduler' but contains only read-only state data. The data model is not needed when the underlying implementation infrastructure supports the Network Management Datastore Architecture (NMDA).

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

revision 2019-01-07 {
  description
    "Initial revision.";
  reference "RFC XXXX: Network Bridge";
}

augment "/if:interfaces-state/if:interface" {
  container scheduler {
    container gate-controllers {
      list gate-controller {
        key "id type";
        leaf id {
          type string;
        }
        leaf type {
          type identityref {
            base sched:gate-controller;
          }
          mandatory true;
        }
      }
    }
    container inputs {
      list input {
        key "class index";
        leaf class {
          type identityref {
            base sched:input;
          }
          mandatory true;
        }
      }
    }
  }
}
base sched:gate-controller-input;
}
leaf index {
    type uint32;
}
uses sched:gate-controller-queue-state;
}
}
container input-classes {
    list input-class {
        key "class";
        leaf class {
            type identityref {
                base sched:gate-controller-input;
            }
        }
        uses sched:gate-controller-queue-state;
    }
}

<CODE ENDS>

Author’s Address

Vladimir Vassilev
Transpacket

Email: vladimir@transpacket.com