Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2021

V. Vassilev
Lightside Instruments AS

February 17, 2021

A YANG Data Model for Network Bridge Management
draft-vassilev-netmod-network-bridge-05

Abstract

This document introduces new YANG model of a flow capable 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 August 21, 2021.

Table of Contents

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

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

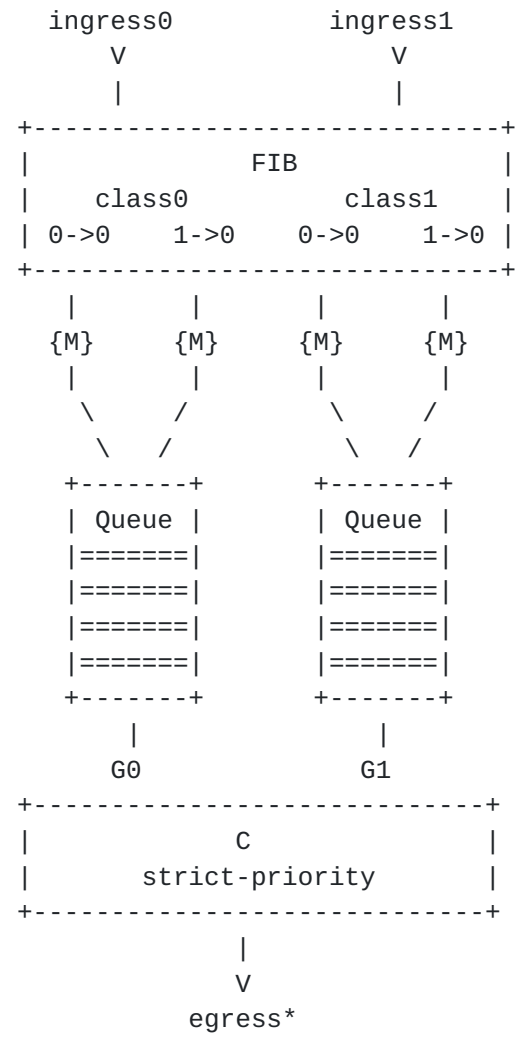
A Network bridge has more then 1 ingress and 1 or more egress ports. It has 1 or more traffic classes. The proposed model splits the design into 2 components - 1) Forwarding component and 2) Scheduling component. The forwarding component is connected to all ingress ports and forwards traffic from them 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.

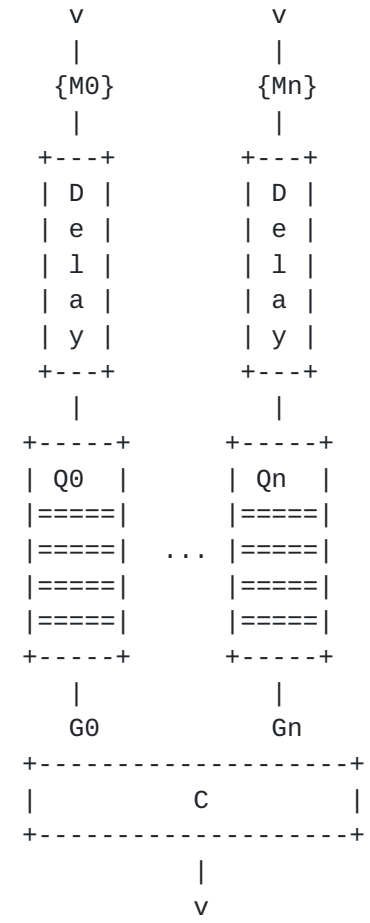
1.3.2. Scheduling

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:

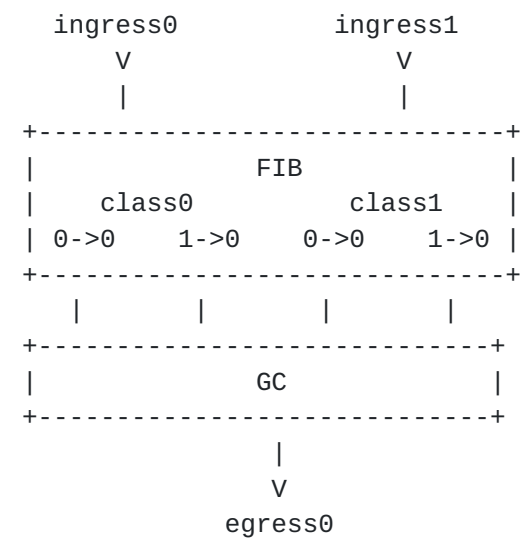


The common structure of a scheduler module (S) consisting of topology of consecutive flow-meters (M), gates (G) connected to a common gate control - (C) with a single egress port. A new module type representing delay line (D) is added to the structure of the

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
      |     +--rw (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?   ether-type
      |        |     +--rw pcp?             vlan-pcp
      |        |     +--rw cfi?             vlan-cfi
      |        |     +--rw vlan-id?         vlan-id
```

```
          |        +--:(set-vlan-cfi-action-case)
          |        |  +--rw set-vlan-cfi-action
          |        |     +--rw vlan-cfi?   vlan-cfi
          |        +--:(set-vlan-id-action-case)
          |        |  +--rw set-vlan-id-action
          |        |     +--rw vlan-id?   vlan-id
          |        +--:(set-vlan-pcp-action-case)
          |        |  +--rw set-vlan-pcp-action
          |        |     +--rw vlan-pcp?   vlan-pcp
          |        +--:(strip-vlan-action-case)
          |           +--rw strip-vlan-action!
          +--ro flow-statistics
             +--ro packet-count?   yang:counter64
             +--ro byte-count?     yang:counter64

    rpcs:
      +---x transmit-packet
         +---w input
            +---w egress?    netbr:port-ref
            +---w ingress?   netbr:port-ref
            +---w payload?   binary
            +---w action* [order]
               +---w order                        int32
               +---w (action)?
                  +--:(output-action-case)
                  |  +---w output-action
                  |     +---w out-port?     netbr:port-ref
                  |     +---w max-length?   uint16
                  +--:(controller-action-case)
                  |  +---w controller-action
                  |     +---w max-length?   uint16
                  +--:(drop-action-case)
                  |  +---w drop-action!
                  +--:(pop-vlan-action-case)
                  |  +---w pop-vlan-action!
                  +--:(push-vlan-action-case)
                  |  +---w push-vlan-action
                  |     +---w ethernet-type?   ether-type
                  |     +---w pcp?             vlan-pcp
                  |     +---w cfi?             vlan-cfi
                  |     +---w vlan-id?         vlan-id
                  +--:(set-vlan-cfi-action-case)
                  |  +---w set-vlan-cfi-action
                  |     +---w vlan-cfi?   vlan-cfi
                  +--:(set-vlan-id-action-case)
                  |  +---w set-vlan-id-action
                  |     +---w vlan-id?   vlan-id
                  +--:(set-vlan-pcp-action-case)
```

```
                |  +---w set-vlan-pcp-action
                |     +---w vlan-pcp?   vlan-pcp
                +--:(strip-vlan-action-case)
                   +---w strip-vlan-action!

  notifications:
    +---n packet-received
       +--ro packet-in-reason?   identityref
       +--ro ingress?            netbr:port-ref
       +--ro payload?            binary
       +--ro match
          +--ro in-port?          netbr:port-ref
          +--ro ethernet-match
          |  +--ro ethernet-source!
          |  |  +--ro address    yang:mac-address
          |  |  +--ro mask?      yang:mac-address
          |  +--ro ethernet-destination!
          |  |  +--ro address    yang:mac-address
          |  |  +--ro mask?      yang:mac-address
          |  +--ro ethernet-type!
          |     +--ro type    ether-type
          +--ro vlan-match
             +--ro vlan-id!
             |  +--ro vlan-id-present?   boolean
             |  +--ro vlan-id?           vlan-id
             +--ro vlan-pcp?   vlan-pcp
```

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?                  port-class-ref
    +--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]
             +--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?   leafref
                     +--rw input-class?       leafref
                     +--rw index?             uint32
```

5. Network Bridge Module YANG

```
<CODE BEGINS> file "ietf-network-bridge@2021-02-17.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@lightside-instruments.com>";
  description
    "This module contains a collection of YANG definitions for
     description and management of network bridges.

     Copyright (c) 2020 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 2021-02-17 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Network Bridge";
  }

  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.";
    }

    augment "/if:interfaces/if:interface" {
      description
        "Bridge port specific data.";
      leaf port-name {
        type leafref {
          path "/netbr:bridge/netbr:ports/netbr:port/netbr:name";
        }
        description
          "Reference to the bridge port";
      }
    }

    container bridge {
      description
        "Bridge parameters.";
      container ports {
        description
          "Member ports.";
        list port {
          key "name";
          unique "index";
          description
            "The list of bridge ports on the device.";
          leaf name {
            type string;
            description
              "Name of the port.";
          }
          leaf index {
            type uint64;
            description
              "Index of the port.";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

6. Network Bridge Flows Module YANG

<CODE BEGINS> file "ietf-network-bridge-flows@2021-02-17.yang"

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

  import ietf-network-bridge {
    prefix netbr;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }

  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@lightside-instruments.com>";
  description
    "This module contains a collection of YANG definitions for
     description and management of network bridge based on
     flows.

     Copyright (c) 2020 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 2021-02-17 {
    description
      "Unreleased revision.";
    reference
```

```
      "RFC XXXX: Network Bridge";
  }

  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";
  }

  typedef vlan-pcp {
    type uint8 {
      range "0..7";
    }
    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.";
  }

  typedef vlan-id {
    type uint16 {
      range "0..4095";
    }
    description
      "IETF 802.1q VLAN tag id.";
  }

  typedef ether-type {
    type uint16;
    description
      "Length/Type field of the generated Ethernet packet.";
```

```
    reference
      "IEEE 802-2014 Clause 9.2";
  }

  typedef vlan-cfi {
    type int32;
    description
      "Canonical Format Identifier (CFI) field
       (shall be 0 for Ethernet switches) of the
       transmitted 802.1q VLAN tag.";
  }

  typedef flow-id {
    type inet:uri;
    description
      "Flow identifier.";
  }

  grouping address {
    description
      "IP address.";
    choice address {
      description
        "Address choice.";
      case ipv4 {
        leaf ipv4-address {
          type inet:ipv4-prefix;
          description
            "IPv4 address.";
        }
      }
      case ipv6 {
        leaf ipv6-address {
          type inet:ipv6-prefix;
          description
            "IPv6 address.";
        }
      }
    }
  }

  grouping action-list {
    description
      "Action list grouping.";
    list action {
      key "order";
      description
        "Contains action with corresponding order index.";
```

```
      leaf order {
        type int32;
        description
          "Order index.";
      }
      uses action;
    }
  }

  grouping action {
    description
      "Grouping of all action data definitions.";
    choice action {
      description
        "Choice with alternative action cases.";
      case output-action-case {
        container output-action {
          description
            "Contains output action specific data.";
          leaf out-port {
            type netbr:port-ref;
            description
              "Port on which the packet is sent.";
          }
          leaf max-length {
            type uint16;
            description
              "Packets above this length are discarded.";
          }
        }
      }
      case controller-action-case {
        container controller-action {
          description
            "Contains controller action specific data.";
          leaf max-length {
            type uint16;
            description
              "Packets above this length are discarted.";
          }
        }
      }
      case drop-action-case {
        container drop-action {
          presence "Drop action case";
          description
            "Drop action presence container.";
        }
```

```
      }
      case pop-vlan-action-case {
        container pop-vlan-action {
          presence "Pop-vlan action case.";
          description
            "Pop-vlan presence container";
        }
      }
      case push-vlan-action-case {
        container push-vlan-action {
          description
            "Contains push-vlan action specific data.";
          leaf ethernet-type {
            type ether-type;
            description
              "Tag protocol identifier (TPID) as defined in
               IEEE 802.1q";
          }
          leaf pcp {
            type vlan-pcp;
            description
              "Specifies the IEEE 802.1p Priority Code Point (PCP) value
               of the pushed 802.1q VLAN tag.";
          }
          leaf cfi {
            type vlan-cfi;
            description
              "Configures the Canonical Format Identifier (CFI) field
               (shall be 0 for Ethernet switches) of the transmitted
               802.1q VLAN tag.";
          }
          leaf vlan-id {
            type vlan-id;
            description
              "Specifies the VLAN ID as defined in IEEE 802.1q of the
               pushed VLAN tag.";
          }
        }
      }
      case set-vlan-cfi-action-case {
        container set-vlan-cfi-action {
          description
            "Contains set-vlan-cfi action specific data. The
             set-vlan-cfi action is used to replace CFI field
             on already tagged packets.";
          leaf vlan-cfi {
            type vlan-cfi;
            description
```

```
            "Configures the Canonical Format Identifier (CFI) field
             to set on the transmitted 802.1q VLAN tagged packet.";
          }
        }
      }
      case set-vlan-id-action-case {
        container set-vlan-id-action {
          description
            "Contains set-vlan-id action specific data. The set-vlan-id
             action is used to replace VLAN ID field on already tagged
             packets.";
          leaf vlan-id {
            type vlan-id;
            description
              "Specifies the VLAN ID to set on the 802.1q VLAN tagged
               packet.";
          }
        }
      }
      case set-vlan-pcp-action-case {
        container set-vlan-pcp-action {
          description
            "Contains set-vlan-pcp action specific data. The
             set-vlan-pcp action is used to replace VLAN PCP
             field on already tagged packets.";
          leaf vlan-pcp {
            type vlan-pcp;
            description
              "Specifies the IEEE 802.1p Priority Code Point
               (PCP) value to set on the 802.1q VLAN tagged
               packet.";
          }
        }
      }
      case strip-vlan-action-case {
        container strip-vlan-action {
          presence "Strip-vlan action case";
          description
            "Strip-vlan presence container.";
        }
      }
    }
  }

  grouping mac-address-filter {
    description
      "Defines address and mask pair for
       definition of basic MAC address filter rules.";
```

```
    leaf address {
      type yang:mac-address;
      mandatory true;
      description
        "MAC address to compare with.";
    }
    leaf mask {
      type yang:mac-address;
      description
        "The mask specifies the bits to compare.
         All bits that are 1s are significant.";
    }
  }

  grouping ethernet-match-fields {
    description
      "Defines data for specification of filter rules for Ethernet
       frames based on source and destination MAC addresses and
       the ethernet type field.";
    container ethernet-source {
      presence "Match field is active and set";
      description
        "Ethernet source address.";
      uses mac-address-filter;
    }
    container ethernet-destination {
      presence "Match field is active and set";
      description
        "Ethernet destination address.";
      uses mac-address-filter;
    }
    container ethernet-type {
      presence "Match field is active and set";
      description
        "Ethernet frame type.";
      leaf type {
        type ether-type;
        mandatory true;
        description
          "Type field of the Ethernet frame.";
      }
    }
  }

  grouping vlan-match-fields {
    description
      "Defines data for specification of filter rules for
       VLAN tagged or not Ethernet frames based on the
```

```
       presence of VLAN tag, the value of the VLAN ID and
       VLAN PCP fields.";
    container vlan-id {
      presence "Match field is active and set";
      description
        "Match 802.1q VLAN ID.";
      leaf vlan-id-present {
        type boolean;
        description
          "If set to false match packets with different
           VLAN ID then the specified in the vlan-id leaf.";
      }
      leaf vlan-id {
        type vlan-id;
        description
          "802.1q VLAN ID to match. The match rule can
           be inverted by setting vlan-id-present to
           false.";
      }
    }
    leaf vlan-pcp {
      type vlan-pcp;
      description
        "Match 802.1p VLAN Priority code point (PCP) field.";
    }
  }

  grouping match {
    description
      "Defines data for specification of filter rules.";
    leaf in-port {
      type netbr:port-ref;
      description
        "Input port to match.";
    }
    container ethernet-match {
      description
        "Ethernet match rules.";
      uses ethernet-match-fields;
    }
    container vlan-match {
      description
        "VLAN match rules.";
      uses vlan-match-fields;
    }
  }

  grouping raw-packet {
```

```
    description
      "Basic packet structure.";
    leaf ingress {
      type netbr:port-ref;
      description
        "Port the packet was received on.";
    }
    leaf payload {
      type binary;
      description
        "Payload of the packet.";
    }
  }

  grouping packet-in {
    description
      "Input packet event data:
       ingress port, payload, event reason.";
    leaf packet-in-reason {
      type identityref {
        base packet-in-reason;
      }
      description
        "Reason identity:
         no-match, send-to-controller, invalid-ttl or another
         reason for the corresponding packet-in event.";
    }
    uses raw-packet;
  }

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

  grouping flow {
    description
      "The definition of a flow has a name,
```

```
       match container specifying filter rules and
       ordered list of actions to be performed on
       each packet.";
    leaf id {
      type flow-id;
      description
        "Flow identifier.";
    }
    container match {
      description
        "Filter rules for the flow.";
      uses match;
    }
    container actions {
      description
        "Ordered list of actions.";
      uses action-list;
    }
  }

  rpc transmit-packet {
    description
      "Sending packet out.";
    input {
      leaf egress {
        type netbr:port-ref;
        description
          "Egress port the packet will be transmitted from.";
      }
      uses raw-packet;
      uses action-list;
    }
  }

  notification packet-received {
    description
      "Delivery of incoming packet.";
    uses packet-in;
    container match {
      description
        "Match data that triggered the source of the
         packet-received event.";
      uses match;
    }
  }

  container packet-in-message {
    description
```

```
      "Container with the last packet-in reported.
       Useful for basic monitoring.";
    uses packet-in;
    container match {
      description
        "Match data that triggered the source of the
         last packet-received event.";
      uses match;
    }
  }
  container flows {
    description
      "Contains the list of all configured flows.";
    list flow {
      key "id";
      description
        "";
      uses flow;
      container flow-statistics {
        config false;
        description
          "Contains flow counters.";
        leaf packet-count {
          type yang:counter64;
          description
            "Packets matched.";
        }
        leaf byte-count {
          type yang:counter64;
          description
            "Sum of the bytes of all matched packets.";
        }
      }
    }
  }
}
```

<CODE ENDS>

7. Network Bridge Scheduler Module YANG

<CODE BEGINS> file "ietf-network-bridge-scheduler@2021-02-17.yang"

```
module ietf-network-bridge-scheduler {
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler";
  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@lightside-instruments.com>";
  description
    "This module contains a collection of YANG definitions for
     description and management of network bridge schedulers.

     Copyright (c) 2020 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 2021-02-17 {
    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-802dot1qbv 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.";
  }

  grouping gate-controller-input-config {
    description
      "Common gate controller input configuration data definitions.";
    leaf constant-propagation-delay {
      type uint64;
      units "picoseconds";
      description
        "Constant delay attributed to delays in the gate-controller.";
    }
    leaf configurable-delay-line {
      type uint64;
      units "picoseconds";
      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
         leafs. Inputs with queues have variable higher delay with
         dynamic component based on the controllers logic.";
```

```
    }
    leaf queue-len {
      type uint32;
      units "bytes";
      description
        "Length of the queue.";
    }
  }

  grouping gate-controller-queue-state {
    description
      "Common gate controller queue state data definitions.";
    leaf queued-pkts {
      type uint64;
      config false;
      description
        "Number of packets queued.";
    }
    leaf queued-bytes {
      type uint64;
      config false;
      description
        "Number of bytes of the packets queued.";
    }
    leaf discards {
      type uint64;
      config false;
      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 {
      type uint64;
      config false;
      description
        "Unintended discard caused by overflow of
         the input queue of the gate controller.";
    }
    leaf error-discards {
```

```
      type uint64;
      config false;
      description
        "Unintended discards caused by error
         in the scheduler.";
    }
  }

  augment "/flow:flows/flow:flow" {
    description
      "Adds traffic-class to the flow model.";
    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" {
    description
      "Adds port class and class-instance-index leafs.";
    leaf class {
      type port-class-ref;
      description
        "A port class allows discrimination of ports based on features
         and supported scheduler options.";
    }
    leaf class-instance-index {
      type uint32;
      description
        "Index enumerating the instances of the same port class.";
    }
  }

  augment "/netbr:bridge" {
    description
      "Adds scheduler specific data to the bridge model.";
    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 {
      description
        "Contains the leaf-list of available traffic classes.";
      leaf-list traffic-class {
        type identityref {
          base traffic-class;
        }
        description
          "Leaf-list of available traffic classes.";
      }
    }
    container port-classes {
      description
        "Contains the leaf-list of available port classes.";
      leaf-list port-class {
        type identityref {
          base port-class;
        }
        description
          "Leaf-list of available port classes.";
      }
    }
  }

  augment "/if:interfaces/if:interface" {
    description
      "Augments the interface model with scheduler specific data.";
    container scheduler {
      description
        "Each egress capable interface has scheduler. The scheduler
         is a tree of interconnected gate controllers.";
      container gate-controllers {
        description
          "Contains the list of gate controllers.";
        list gate-controller {
          key "id";
          description
            "The gate controller model can be augmented by
             external modules defining custom gate controller
             types.";
          leaf id {
            type string;
            description
              "Gate controller identifier.";
```

```
          }
          leaf type {
            type identityref {
              base gate-controller;
            }
            mandatory true;
            description
              "Gate controller type.";
          }
          container inputs {
            description
              "Contains the list of inputs.";
            list input {
              key "class index";
              description
                "Double key list. There can be multiple
                 instances of each input class.";
              leaf class {
                type identityref {
                  base gate-controller-input;
                }
                description
                  "Input class.";
              }
              leaf index {
                type uint32;
                description
                  "Index of the input instance of the corresponding
                   input class.";
              }
              uses gate-controller-queue-state;
            }
          }
          container input-classes {
            description
              "Contains the list of input-classes.";
            list input-class {
              key "class";
              description
                "Contains configuration and state data that is common
                 for all instances of certain input class.";
              leaf class {
                type identityref {
                  base gate-controller-input;
                }
                description
                  "Input class.";
              }
```

```
            uses gate-controller-queue-state;
          }
        }
      }
    }
  }

  augment "/netbr:bridge" {
    description
      "Augments the bridge module with
       scheduler specific configuration data.";
    container scheduler-classes {
      description
        "Contains list of scheduler-classes.";
      list scheduler-class {
        key "egress-port-class";
        description
          "All ports of same class inherit the scheduler configuration.
           The instance specific scheduler configuration defined under
           /interfaces/interface/scheduler can override this
           configuration.";
        leaf egress-port-class {
          type sched:port-class-ref;
          description
            "The port class the scheduler coonfiguration
             applies for.";
        }
        container inputs {
          description
            "Contains list of inputs.";
          list input {
            key "traffic-class ingress-port-class";
            description
              "Double key list. There can be multiple
               instances of each input class.";
            leaf traffic-class {
              type traffic-class-ref;
              description
                "Reference to traffic class.";
            }
            leaf ingress-port-class {
              type port-class-ref;
              description
                "Reference to port class.";
            }
            leaf gate-controller {
              type leafref {
```

```
            path "../../../gate-controllers/gate-controller/id";
          }
          description
            "Reference to gate controller id.";
        }
        leaf input-class {
          type leafref {
            path "../../../gate-controllers/gate-controller"
               + "[id=current()/../gate-controller]"
               + "/inputs/input/class";
          }
          description
            "Reference to a input class defined for the specified
             gate controller.";
        }
        leaf base-index {
          type uint32;
          default "0";
          description
            "Base index for the inputs of this class.";
        }
      }
    }
    container gate-controllers {
      description
        "Contains the list of gate controllers.";
      list gate-controller {
        key "id";
        description
          "The gate controller model can be augmented by
           external modules defining custom gate controller
           types.";
        leaf id {
          type string;
          description
            "Gate controller identifier.";
        }
        leaf type {
          type identityref {
            base gate-controller;
          }
          mandatory true;
          description
            "Gate controller type.";
        }
        container inputs {
          description
            "Contains the list of inputs.";
```

```
        list input {
          key "class";
          description
            "Double key list. There can be multiple
             instances of each input class.";
          leaf class {
            type identityref {
              base gate-controller-input;
            }
            mandatory true;
            description
              "Input class.";
          }
          leaf instance-count {
            type uint32;
            description
              "Number of input instances of the specified
               class.";
          }
          uses gate-controller-input-config;
        }
      }
      container output {
        description
          "Configuration of the gate-controller input
            the output is connected to.";
        leaf gate-controller {
          type leafref {
            path "../../../gate-controller/id";
          }
          description
            "Specifies the gate-controller
             this output is connected to.";
        }
        leaf input-class {
          type leafref {
            path "../../../gate-controller"
               + "[id=current()/../gate-controller]/"
               + "inputs/input/class";
          }
          description
            "Specifies the input-class of the gate-controller
             the input this output is connected to.";
        }
        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].

```
name:    ietf-network-bridge
namespace:
         urn:ietf:params:xml:ns:yang:ietf-network-bridge
prefix:  netbr
// RFC Ed. remove this line and replace XXXX in next line
reference:    RFC XXXX

name:    ietf-network-bridge-flows
namespace:
         urn:ietf:params:xml:ns:yang:ietf-network-bridge-flows
prefix:  flow
// RFC Ed. remove this line and replace XXXX in next line
reference:    RFC XXXX

name:    ietf-network-bridge-scheduler
namespace:
         urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler
prefix:  sched
// RFC Ed. remove this line and replace XXXX in next line
reference:    RFC XXXX
```

9. Security Considerations

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

10. References

10.1. Normative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

10.2. Informational References

[OpenFlow] "Open Networking Foundation", ""OpenFlow Switch Specification"", December 2009, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Example

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

A.1. Model

```
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;
  }
```

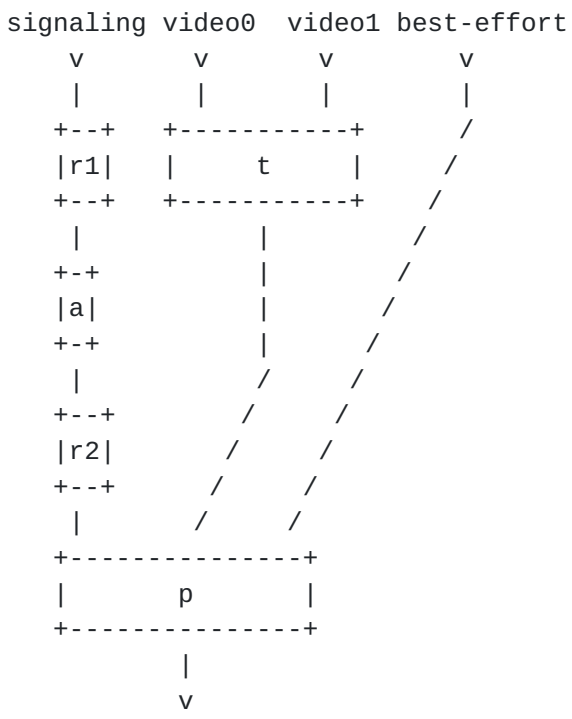
```
  identity pri0 {
    base sched:shared-queue-aggregator-input;
    base strict-priority-aggregator;
  }

  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

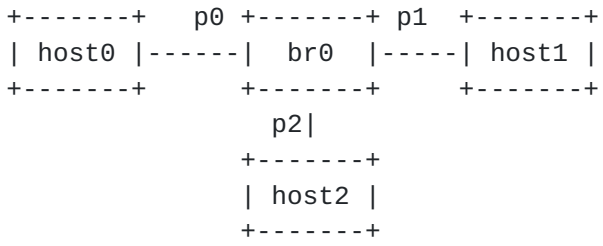
The scheduler toplogy and the gate controller instances are specified in the operational configuration data that can be modified or not depending on the underlying implementation. The single letter identifiers for the gate-controllers have the following identities:

- 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 \
    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 \
    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\
/address=00:01:02:03:00:00 \
    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\
/address=00:01:02:03:00:01 \
    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\
/address=00:01:02:03:00:02 \
    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\
/address=00:01:02:03:00:00 \
    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\
/address=00:01:02:03:00:01 \
    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\
/address=00:01:02:03:00:02 \
    actions/action[order='0']/output-action/out-port=p2
> merge /flows/flow[id='ptp-to-host2'] -- \
    traffic-class=signaling
> commit
```

CLI commands configuring and monitorig 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>
      </port>
      </ports>
      <default-traffic-class
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler"
      xmlns: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"
      xmlns:example="http://example.com/ns/example-bridge">
      example:best-effort</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
      xmlns="urn:ietf:params:xml:ns:yang:ietf-network-bridge-scheduler">

        <scheduler-class>
          <egress-port-class
          xmlns:example="http://example.com/ns/example-bridge">
          example:default-port</egress-port-class>
          <inputs>
            <input>
```

```
            <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">
```

```
            example:timeslot1</input-class>
            <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>
            <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-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>
              <input>
                <class
```

```
        xmlns:example="http://example.com/ns/example-bridge">
        example:pri2</class>
        <instance-count>3</instance-count>
        <queue-len>1048576</queue-len>
      </input>
    </inputs>
  </gate-controller>
  <gate-controller>
    <id>r1</id>
    <type
    xmlns:example="http://example.com/ns/example-bridge">
    example:rate-limiter</type>
    <inputs>
      <input>
        <class
        xmlns:example="http://example.com/ns/example-bridge">
        example:in</class>
        <instance-count>3</instance-count>
      </input>
    </inputs>
    <output>
      <gate-controller>a</gate-controller>
      <input-class
      xmlns:example="http://example.com/ns/example-bridge">
      example:pri0</input-class>
      <index>0</index>
    </output>
    <interval xmlns="http://example.com/ns/example-bridge">
    10000000</interval>
    <limit xmlns="http://example.com/ns/example-bridge">
    12500</limit>
  </gate-controller>
  <gate-controller>
    <id>r2</id>
    <type xmlns:example="http://example.com/ns/example-bridge">
    example:rate-limiter</type>
    <inputs>
      <input>
        <class
        xmlns:example="http://example.com/ns/example-bridge">
        example:in</class>
        <instance-count>1</instance-count>
      </input>
    </inputs>
    <output>
      <gate-controller>p</gate-controller>
      <input-class
      xmlns:example="http://example.com/ns/example-bridge">
```

```
              example:pri0</input-class>
              <index>0</index>
            </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>
              <index>2</index>
            </output>
            <period xmlns="http://example.com/ns/example-bridge">
            10000000</period>
            <time-slot0-interval
            xmlns="http://example.com/ns/example-bridge">
            5000000</time-slot0-interval>
            <time-slot1-interval
            xmlns="http://example.com/ns/example-bridge">
            5000000</time-slot1-interval>
          </gate-controller>
        </gate-controllers>
      </scheduler-class>
    </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>
    <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>
      <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:signaling</traffic-class>
    </flow>
    <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: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>
```

```
      <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@2021-02-17.yang"

```
module ietf-network-bridge-scheduler-state {
  namespace
    "urn:ietf:params:xml:ns:yang: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>

     Editor:   Vladimir Vassilev
```

```
                <mailto:vladimir@lightside-instruments.com>";
    description
      "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) 2020 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 2021-02-17 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Network Bridge";
    }

    augment "/if:interfaces-state/if:interface" {
      description
        "Augments the interface model with scheduler specific data.";
      container scheduler {
        description
          "Each egress capable interface has scheduler. The scheduler
           is a tree of interconnected gate controllers.";
        container gate-controllers {
          description
            "Contains the list of gate controllers.";
          list gate-controller {
            key "id type";
            description
              "The gate controller model can be augmented by
               external modules defining custom gate controller
               types.";
            leaf id {
              type string;
              description
```

```
        "Gate controller identifier.";
      }
      leaf type {
        type identityref {
          base sched:gate-controller;
        }
        mandatory true;
        description
          "Gate controller type.";
      }
      container inputs {
        description
          "Contains the list of inputs.";
        list input {
          key "class index";
          description
            "Double key list. There can be multiple
             instances of each input class.";
          leaf class {
            type identityref {
              base sched:gate-controller-input;
            }
            description
              "Input class.";
          }
          leaf index {
            type uint32;
            description
              "Index of the input instance of the corresponding
               input class.";
          }
          uses sched:gate-controller-queue-state;
        }
      }
      container input-classes {
        description
          "Contains the list of input-classes.";
        list input-class {
          key "class";
          description
            "Contains the list of input-classes.";
          leaf class {
            type identityref {
              base sched:gate-controller-input;
            }
            description
              "Input class.";
          }
```

```
                uses sched:gate-controller-queue-state;
              }
            }
          }
        }
      }
    }
  }
```

<CODE ENDS>

Author's Address

Vladimir Vassilev
Lightside Instruments AS

Email: vladimir@lightside-instruments.com