Deterministic Networking (DetNet) Configuration YANG Model
draft-ietf-detnet-yang-02

Abstract

This document contains the specification for Deterministic Networking flow configuration YANG Model. The model allows for provisioning of end-to-end DetNet service along the path without dependency on any signaling protocol.

The YANG module defined in this document conforms to the Network Management Datastore Architecture (NMDA).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 September 27, 2019.
1. Introduction

Deterministic Networking (DetNet) [I-D.ietf-detnet-architecture] is defined to provide high-quality network service with extremely low packet loss rate, bounded low latency and jitter.

DetNet flow information is defined in [I-D.ietf-detnet-flow-information-model], and the DetNet models are categorized as:

- **Flow models**: describe characteristics of data flows. These models describe in detail all relevant aspects of a flow that are needed to support the flow properly by the network between the source and the destination(s).
Service models: describe characteristics of services being provided for data flows over a network. These models can be treated as a network operator independent information model.

Configuration models: describe in detail the settings required on network nodes to serve a data flow properly. Service and flow information models are used between the user and the network operator. Configuration information models are used between the management/control plane entity of the network and the network nodes.

They are shown in the Figure 1.

```
User          Network Operator
| flow/service | +-----+ model +-----+ |
| model        |   ^<--------------->^ X | management/control |
| ^            |    ^+-^ plane entity   |
| v configuration model |      ^-----|^ configuration model |
| +-----| v network |
| v      | +-----| v nodes |
+-----+ +-----+ +-----+
```

Figure 1. Three Information Models

DetNet YANG [RFC7950] [RFC6991] models include:

DetNet YANG [RFC7950] [RFC6991] models are used for DetNet service configurations, QoS configuration and topology discovery. DetNet topology model is defined in ietf-detnet-topology-yang. This document defines two YANG models, which are referred to as DetNet flow configuration model and DetNet transport QoS model. DetNet flow model is designed for DetNet flow path configuration and flow status reporting. DetNet transport QoS model is designed for QoS attributes configuration of transport tunnels to achieve end-to-end bounded latency and zero congestion loss.

2. Terminologies

This document uses the terminologies defined in [I-D.ietf-detnet-architecture].
3. DetNet Configuration Model

DetNet flow configuration includes DetNet Service Proxy configuration, DetNet Service Layer configuration and DetNet Transport Layer configuration. The corresponding attributes used in different layers are defined in Section 3.1, 3.2, 3.3, respectively.

3.1. DetNet Service Proxy Configuration Attributes

DetNet service proxy is responsible for mapping between application flows and DetNet flows at the edge node (egress/ingress node). Where the application flows can be either layer 2 or layer 3 flows. To identify a flow at the User Network Interface (UNI), as defined in [I-D.ietf-detnet-flow-information-model], the following flow attributes are introduced:

- DetNet L3 Flow Identification, refers to Section 7.1.1 of [I-D.ietf-detnet-flow-information-model]
- DetNet L2 Flow Identification, refers to Section 7.1.2 of [I-D.ietf-detnet-flow-information-model]

DetNet service proxy can also do flow filtering and policing at the ingress to prevent the misbehaved flows from going into the network, which needs:

- Traffic Specification, refers to Section 7.2 of [I-D.ietf-detnet-flow-information-model]

The YANG module structure is shown below:
3.2. DetNet Service Layer Configuration Attributes

DetNet service functions, e.g., DetNet tunnel initialization/termination and service protection, are provided in DetNet service layer. To support these functions, the following service attributes need to be configured:

- DetNet flow identification, refers to Section 7.1.3 of [I-D.ietf-detnet-flow-information-model].
- Service function indication, indicates which service function will be invoked at a DetNet edge, relay node or end station. (DetNet tunnel initialization or termination are default functions in...
DetNet service layer, so there is no need for explicit indication.)

- **Flow Rank**, refers to Section 7.3 of [I-D.ietf-detnet-flow-information-model].

- **Service Rank**, refers to Section 7.4 of [I-D.ietf-detnet-flow-information-model].

- **Service decapsulation**, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls]

- **Transport decapsulation**, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls] and Section 3 of [I-D.ietf-detnet-dp-sol-ip]

- **Service encapsulation**, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls]

- **Transport encapsulation**, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls] and Section 3 of [I-D.ietf-detnet-dp-sol-ip]

The YANG module structure is shown below:

```Yang
++--:(relay-node) {detnet-mpls-dp-sol}?
   ++--rw relay-node
      +--rw name?          string
      +--rw service-rank
      +--rw in-segment* [in-segment-id]
         +--rw in-segment-id        uint32
         +--rw (flow-type)?
            +--:(IP)
               +--rw (ip-flow-type)?
                  +--:(ipv4)
                     +--rw src-ipv4-address?   inet:ipv4-address
                     +--rw dest-ipv4-address?   inet:ipv4-address
                     +--rw dscp?                uint8
                  +--:(ipv6)
                     +--rw src-ipv6-address?   inet:ipv6-address
                     +--rw dest-ipv6-address?   inet:ipv6-address
                     +--rw traffic-class?       uint8
                     +--rw flow-label?          inet:ipv6-flow-label
                         +--rw source-port?        inet:port-number
                         +--rw destination-port?    inet:port-number
                         +--rw protocol?            uint8
      +--:(MPLS)
         +--rw service-label         uint32
```
| +--rw service-function?    service-function-type
| +--rw out-segment* [out-segment-id]
| +--rw out-segment-id        uint32
| +--rw detnet-service-encapsulation
| | +--rw service-label        uint32
| | +--rw control-word?       uint32
| +--rw detnet-transport-encapsulation
| +--rw (tunnel-type)?
| | +--:(IPv4) {ipv4-tunnel}?
| | | +--rw ipv4-encapsulation
| | | | +--rw src-ipv4-address    inet:ipv4-address
| | | | +--rw dest-ipv4-address   inet:ipv4-address
| | | | +--rw protocol            uint8
| | | | +--rw ttl?                uint8
| | | | +--rw dscp?               uint8
| | +--:(IPv6) {ipv6-tunnel}?
| | | +--rw ipv6-encapsulation
| | | | +--rw src-ipv6-address    inet:ipv6-address
| | | | +--rw dest-ipv6-address   inet:ipv6-address
| | | | +--rw next-header          uint8
| | | | +--rw traffic-class?      uint8
| | | | +--rw flow-label?         inet:ipv6-flow-label
| | | | +--rw hop-limit?          uint8
| +--:(MPLS) {mpls-tunnel}?
| | +--rw mpls-encapsulation
| | | +--rw label-operations* [label-oper-id]
| | | +--rw label-oper-id       uint32
| | | +--rw (label-actions)?
| | | | +--:(label-push)
| | | | | +--rw label-push
| | | | | | +--rw label        uint32
| | | | | | +--rw s-bit?           boolean
| | | | | | +--rw tc-value?         uint8
| | | | | | +--rw ttl-value?        uint8
| | | | +--:(label-swap)
| | | | +--rw label-swap
| | | | | +--rw out-label         uint32
| | | | | +--rw ttl-action?       ttl-action-definition
| +--:(MPLS-over-UDP) {mpls-over-udp-tunnel}?
| | +--rw mpls-over-udp-encapsulation
| | | +--rw label-operations* [label-oper-id]
| | | +--rw label-oper-id       uint32
| | | +--rw (label-actions)?
| | | | +--:(label-push)
| | | | | +--rw label-push
| | | | | | +--rw label        uint32
| | | | | | +--rw s-bit?           boolean
| | | | | | +--rw tc-value?         uint8
| | | | | +--:(label-swap)
3.3. DetNet Transport Layer Configuration Attributes

As defined in [I-D.ietf-detnet-architecture], DetNet transport layer optionally provides congestion protection for DetNet flows over paths provided by the underlying network. Explicit route is another mechanism that is used by DetNet to avoid temporary interruptions caused by the convergence of routing or bridging protocols, and it is also implemented at the DetNet transport layer.

To support congestion protection and explicit route, the following transport layer related attributes are necessary:

- **Traffic Specification**, refers to Section 7.2 of [I-D.ietf-detnet-flow-information-model]. It may used for bandwidth reservation, flow shaping, filtering and policing.

- **Explicit path**, existing explicit route mechanisms can be reused. For example, if Segment Routing (SR) tunnel is used as the transport tunnel, the configuration is mainly at the ingress node of the transport layer; if the static MPLS tunnel is used as the transport tunnel, the configurations need to be at every transit node along the path; for pure IP based transport tunnel, it’s similar to the static MPLS case.

The YANG module structure is shown below:
4. DetNet Configuration YANG Structure

module: ietf-detnet-flow-config
  +++: (transit-node)
  +++:rw transit-node
  +++:rw interval? uint32
  +++:rw max-packets-per-interval? uint32
  +++:rw max-payload-size? uint32
  +++:rw average-packets-per-interval? uint32
  +++:rw average-payload-size? uint32
  +++: (relay-node) {detnet-mpls-dp-sol}?
  +++:rw relay-node
  +++:rw name? string
  +++:rw service-rank
  +++:in-segment* [in-segment-id]
   +++:in-segment-id uint32
  +++: (flow-type)?
   +++: (IP)
    +++: (ip-flow-type)?
     +++: (ipv4)
      +++:rw src-ipv4-address inet:ipv4-address
      +++:rw dest-ipv4-address inet:ipv4-address
      +++:rw dscp? uint8
     +++: (ipv6)
      +++:rw src-ipv6-address inet:ipv6-address
      +++:rw dest-ipv6-address inet:ipv6-address
      +++:rw traffic-class? uint8
      +++:rw flow-label? inet:ipv6-flow-label
      +++:rw source-port? inet:port-number
      +++:rw destination-port? inet:port-number
      +++:rw protocol? uint8
     +++: (MPLS)
      +++:rw service-label uint32
      +++:rw service-function? service-function-type
      +++:out-segment* [out-segment-id]
      +++:out-segment-id uint32

The parameters for DetNet transport QoS are defined in Section 5.
+-rw out-label       uint32
++-rw ttl-action?    ttl-action-definition
+-rw source-port?    inet:port-number
++-rw destination-port?  inet:port-number
+-rw (address-family)?
    +--:{IPv4}
        +-rw src-ipv4-address     inet:ipv4-address
        +-rw dest-ipv4-address    inet:ipv4-address
        +-rw protocol             uint8
        +-rw ttl?                 uint8
        +-rw dscp?                uint8
    +--:{IPv6}
        +-rw src-ipv6-address     inet:ipv6-address
        +-rw dest-ipv6-address    inet:ipv6-address
        +-rw next-header          uint8
        +-rw traffic-class?       uint8
        +-rw flow-label?          inet:ipv6-flow-label
        +-rw hop-limit?           uint8
    +-rw interval?       uint32
    +-rw max-packets-per-interval?       uint32
    +-rw max-payload-size?      uint32
    +-rw average-packets-per-interval?   uint32
    +-rw average-payload-size?    uint32
++-:(edge-node) {detnet-mpls-dp-sol}?
    +--rw edge-node
        +--rw (edge-node-type)?
            +--:(ingress-node)
                +-rw client-flow* [flow-id]
                    +--rw flow-id             uint32
                    +--rw flow-rank?         boolean
                    +--rw (flow-type)?
                        +--:{l2-flow}
                            +-rw source-mac-address?  yang:mac-address
                            +-rw destination-mac-address?  yang:mac-address
                            +-rw ethertype?             eth:ethertype
                            +-rw vlan-id?              uint16
                            +--rw pcp
                        +--:{l3-flow}
                            +-rw (ip-flow-type)?
                                +--:{ipv4}
                                    +-rw src-ipv4-address?     inet:ipv4-address
                                    +-rw dest-ipv4-address?    inet:ipv4-address
                                    +-rw dscp?                uint8
                                +--:{ipv6}
                                    +-rw src-ipv6-address?     inet:ipv6-address
                                    +-rw dest-ipv6-address?    inet:ipv6-address
                                    +-rw flow-label?           inet:ipv6-flow-label
Internet-Draft                DetNet Model                    March 2019

|           |  |  |  |     +--rw src-ipv6-address? inet:ipv6-address
|           |  |  |  |     +--rw dest-ipv6-address? inet:ipv6-address
|           |  |  |  |     +--rw traffic-class? uint8
|           |  |  |  |     +--rw flow-label? inet:ipv6-flow-label
|           |  |  |  |     +--rw source-port? inet:port-number
|           |  |  |  |     +--rw destination-port? inet:port-number
|           |  |  |  |     +--rw protocol? uint8
|           |  |  |  +--:(MPLS)
|           |  |     +--rw service-label uint32
|           |  |  +--rw service-function? service-function-type
|           |  +--rw (tunnel-type)?
|           |     +--:(ipv4) {ipv4-tunnel}?
|           |     |  +--rw ipv4-decap tunnel-decap-action-def
|           |     |  +--rw ttl-action? ttl-action-def
|           |     +--:(ipv6) {ipv6-tunnel}?
|           |     |  +--rw ipv6-decap tunnel-decap-action-def
|           |     |  +--rw hop-limit-action? hop-limit-action-def
|           |     +--:(mpls) {mpls-tunnel}?
|           |     |  +--rw label-pop mpls-label-action-def
|           |     +--rw ttl-action? ttl-action-def
|           +--rw label-pop
|           +--rw ttl-action? ttl-action-def
|       +--:(end-station)
|       +--rw end-station
|     +--rw (edge-node-type)?
|     +--:(ingress-node)
|     |  +--rw client-flow* [flow-id]
|     |     +--rw flow-id uint32
|     |     +--rw flow-rank? boolean
|     |     +--rw (flow-type)?
|     |     +--:(l2-flow)
|     |     |  +--rw source-mac-address? yang:mac-address
|     |     |  +--rw destination-mac-address? yang:mac-address
|     |     |  +--rw ethertype? eth:ethertype
|     |     |  +--rw vlan-id? uint16
|     |     |  +--rw pcp
|     |     +--:(l3-flow)
|     |     |  +--rw (ip-flow-type)?
|     |     |     +--:(ipv4)
|     |     |     |  +--rw src-ipv4-address? inet:ipv4-address
|     |     |     |  +--rw dest-ipv4-address? inet:ipv4-address
|     |     |     |  +--rw dscp? uint8
|     |     |     +--:(ipv6)
|     |     |     |  +--rw src-ipv6-address? inet:ipv6-address
|     |     |     |  +--rw dest-ipv6-address? inet:ipv6-address
|     |     |     +--:(MPLS)
|     |     |     +--rw service-label uint32
|     |     |     +--rw service-function? service-function-type
++rw traffic-class?  uint8
  ++rw flow-label?    inet:ipv6-flow-label
++rw source-port?    inet:port-number
  ++rw destination-port?  inet:port-number
  ++rw protocol?      uint8
++rw traffic-specification
  ++rw interval?      uint32
  ++rw max-packets-per-interval? uint32
  ++rw max-payload-size?  uint32
  ++rw average-packets-per-interval? uint32
  ++rw average-payload-size?  uint32
++rw service-function?  service-function-type
++rw sequence-number-generation-type?  sequence-number-generation
  ++rw sequence-number-length?      uint8
++rw out-segment* [out-segment-type]
  ++rw out-segment-id            uint32
++rw detnet-service-encapsulation
  ++rw service-label  uint32
  ++rw control-word?  uint32
++rw detnet-transport-encapsulation
  ++rw (tunnel-type)?
    +++:(IPv4) {ipv4-tunnel}?
      ++rw ipv4-encapsulation
        ++rw src-ipv4-address     inet:ipv4-address
        ++rw dest-ipv4-address    inet:ipv4-address
        ++rw protocol             uint8
        ++rw ttl?                 uint8
        ++rw dscp?                uint8
    +++:(IPv6) {ipv6-tunnel}?
      ++rw ipv6-encapsulation
        ++rw src-ipv6-address     inet:ipv6-address
        ++rw dest-ipv6-address    inet:ipv6-address
        ++rw traffic-class?       uint8
        ++rw flow-label?          inet:ipv6-flow-label
        ++rw hop-limit?           uint8
    +++:(MPLS) {mpls-tunnel}?
      ++rw mpls-encapsulation
        ++rw label-operations* [label-oper-id]
          ++rw label-oper-id    uint32
          ++rw (label-actions)?
            +++:(label-push)
              ++rw label-push
                ++rw label
                  ++rw s-bit?    boolean
                  ++rw tc-value?  uint8
                  ++rw ttl-value? uint8
                +++:(label-swap)
5. DetNet Configuration YANG Model

<CODE BEGINS> file "ietf-detnet@20190321.yang"
module ietf-detnet{
   namespace "urn:ietf:params:xml:ns:yang:ietf-detnet";
   //yang-version 1.1;
   prefix "detnet-flow";

   import ietf-yang-types { 
      prefix "yang";
   }

   import ietf-interfaces { 
      prefix "if";
   }

   import ietf-inet-types{ 
      prefix "inet";
   }

   ...
/* import ietf-ethertypes {
    prefix "eth";
} */

import ietf-routing-types {
    prefix "rt-types";
}

organization "IETF DetNet Working Group";

contact
"WG Web: <http://tools.ietf.org/wg/detnet/>
WG List: <mailto: detnet@ietf.org>
WG Chair: Lou Berger
    <mailto:lberger@labn.net>
    Janos Farkas
    <janos.farkas@ericsson.com>
Editor: Xuesong Geng
    <mailto:gengxuesong@huawei.com>
Editor: Mach Chen
    <mailto:mach.chen@huawei.com>
Editor: Zhenqiang Li
    <lizhenqiang@chinamobile.com>
Editor: Reshad Rahman
    <rrahman@cisco.com>";

description
"This YANG module describes the parameters needed
for DetNet flow configuration and flow status
reporting.";

revision "2018-09-10" {
    description "initial revision";
    reference "RFC XXXX: draft-geng-detnet-config-yang-05";
}

feature ipv4-tunnel {
    description
        "This feature means that a node support
           IPv4 tunnel encapsulation capability.";
}

feature ipv6-tunnel {
    description
"This feature means that a node support IPv6 tunnel encapsulation capability."

}  

feature mpls-tunnel {  
description  
"This feature means that a node support MPLS tunnel encapsulation capability."  
}

feature mpls-over-udp-tunnel {  
description  
"This feature means that a node supports MPLS over UDP tunnel encapsulation capability."  
}

feature detnet-mpls-dp-sol {  
description  
"This feature means that MPLS data plane solution is supported."  
}

identity detnet-node-role {  
description  
"base detnet-node-role"  
}

identity end-station {  
base detnet-node-role;  
description  
"Commonly called a ‘host’ in IETF documents, and an ‘end station’ is IEEE 802 documents. End systems of interest to this document are either sources or destinations of DetNet flows. And end system may or may not be DetNet transport layer aware or DetNet service layer aware."  
}

identity edge-node {  
base detnet-node-role;  
description  
"An instance of a DetNet relay node that includes either a DetNet service layer proxy function for DetNet service protection (e.g. the addition or removal of packet sequencing information) for one or more end systems, or
starts or terminate congestion protection at
the DetNet transport layer, analogous to a
Label Edge Router (LER).";
}

identity relay-node {
  base detnet-node-role;
  description "A DetNet node including a service layer
  function that interconnects different DetNet
  transport layer paths to provide service
  protection. A DetNet relay node can be a bridge,
a router, a firewall, or any other system that
participates in the DetNet service layer. It
typically incorporates DetNet transport layer
functions as well, in which case it is
collocated with a transit node.";
}

identity transit-node {
  base detnet-node-role;
  description "A node operating at the DetNet transport layer,
  that utilizes link layer and/or network layer
  switching across multiple links and/or
  sub-networks to provide paths for DetNet
  service layer functions. Optionally provides
  congestion protection over those paths. An MPLS
  LSR is an example of a DetNet transit node.";
}

identity tunnel-decap-action {
  description "Base identify from which all tunnel decap
  actions are derived.
  Tunnel decap actions include:
  ipv4-decap - to decap an IPv4 tunnel,
  ipv6-decap - to decap an IPv6 tunnel.";
}

identity ipv4-decap {
  base "tunnel-decap-action";
  description "IPv4 tunnel decap.";
}

identity ipv6-decap {
  base "tunnel-decap-action";
typedef tunnel-decap-action-def {
  type identityref {
    base "tunnel-decap-action";
  }
  description
    "Tunnel decap def.";
}

identity ttl-action {
  description
    "Base identity from which all TTL actions are derived.";
}

identity no-action {
  base "ttl-action";
  description
    "Do nothing regarding the TTL.";
}

identity copy-to-inner {
  base "ttl-action";
  description
    "Copy the TTL of the outer header to the inner header.";
}

identity decrease-and-copy-to-inner {
  base "ttl-action";
  description
    "Decrease TTL by one and copy the TTL to the inner header.";
}

typedef ttl-action-def {
  type identityref {
    base "ttl-action";
  }
  description
    "TTL action definition.";
}

identity hop-limit-action {
  description
"Base identity from which all hop limit actions are derived."
}
typedef hop-limit-action-def {
type identityref {
    base "hop-limit-action";
}
description
"hop limit action definition.";
}

identity mpls-label-action {
description
"Base identity from which all MPLS label operations are derived. The MPLS label stack operations include:
push - to add a new label to a label stack,
pop - to pop the top label from a label stack,
swap - to exchange the top label of a label stack with new label.";
}

identity label-push {
    base "mpls-label-action";
    description
    "MPLS label stack operation: push.";
}

identity label-pop {
    base "mpls-label-action";
    description
    "MPLS label stack operation: pop.";
}

identity label-swap {
    base "mpls-label-action";
    description
    "MPLS label stack operation: swap.";
}

typedef mpls-label-action-def {
type identityref {
    base "mpls-label-action";
}
description
"MPLS label action definition.";
}
identity detnet-transport-layer {
  description
    "The layer that optionally provides congestion
    protection for DetNet flows over paths provided
    by the underlying network."
}

identity detnet-service-layer {
  description
    "The layer at which service protection is
    provided, either packet sequencing, replication,
    and elimination or packet encoding"
}

typedef service-function-type {
  type enumeration {
    enum null {
      description
        "No service function is enabled.";
    }
    enum replication {
      description
        "A Packet Replication Function (PRF) replicates
        DetNet flow packets and forwards them to one or
        more next hops in the DetNet domain. The number
        of packet copies sent to each next hop is a
        DetNet flow specific parameter at the node doing
        the replication. PRF can be implemented by an
        edge node, a relay node, or an end system";
    }
    enum elimination {
      description
        "A Packet Elimination Function (PEF) eliminates
        duplicate copies of packets to prevent excess
        packets flooding the network or duplicate
        packets being sent out of the DetNet domain.
        PEF can be implemented by an edge node, a relay
        node, or an end system.";
    }
    enum ordering {
      description
        "A Packet Ordering Function (POF) re-orders
        packets within a DetNet flow that are received
        out of order. This function can be implemented
        by an edge node, a relay node, or an end system.";
    }
    enum elimination-ordering {
      description
        "A Packet Elimination-Ordering Function (PEOF) en-
        sures that packets are eliminated in a fixed
        order. This function can be implemented by an
        edge node, a relay node, or an end system.";
    }
  }
}
"A combination of PEF and POF that can be implemented by an edge node, a relay node, or an end system."

} enum elimination-replication {
    description
    "A combination of PEF and PRF that can be implemented by an edge node, a relay node, or an end system"
}

} enum elimination-ordering-replication {
    description
    "A combination of PEF, POF and PRF that can be implemented by an edge node, a relay node, or an end system"
}
}

} description
"DetNet service function and function combination types.";

}

typedef sequence-number-generation {
    type enumeration {
        enum "copy-from-app-flow" {
            description
            "DetNet flow sequence number is copied from application flow."
        }
        enum "generated-by-edge-node" {
            description
            "DetNet flow sequence number is generated by DetNet edge node."
        }
    }
    description
    "DetNet sequence number generation types.";
}

grouping detnet-sequence-number {
    description
    "DetNet sequence number.");
    leaf sequence-number-generation-type {
        type sequence-number-generation;
        description
        "The way on how sequence number is generated.";
    }
    leaf sequence-number-length {

type uint8;
description
"DetNet sequence number length."
}
}

grouping detnet-transport-identifier {
description
"DetNet transport identifier"
}

grouping detnet-transport-qos {
//Editor notes: this will be defined in a separate
//              YANG model (detnet-transport-qos).
//              More inputs and discussions are needed here.
description
"DetNet transport tunnel QoS attributes."
uses traffic-specification;
}

grouping ipv4-header {
description
"The IPv4 header encapsulation information."
leaf src-ipv4-address {
type inet:ipv4-address;
mandatory true;
description
"The source IP address of the header."
}
leaf dest-ipv4-address {
type inet:ipv4-address;
mandatory true;
description
"The destination IP address of the header."
}
leaf protocol {
type uint8;
mandatory true;
description
"The protocol id of the header."
}
leaf ttl {
type uint8;
description
"The TTL of the header."
}
leaf dscp {
type uint8;
description
"The DSCP field of the header."
}
}
grouping ipv6-header {

description
"The IPv6 header encapsulation information."
leaf src-ipv6-address {
    type inet:ipv6-address;
    mandatory true;
    description
    "The source IP address of the header."
}
leaf dest-ipv6-address {
    type inet:ipv6-address;
    mandatory true;
    description
    "The destination IP address of the header."
}
leaf next-header {
    type uint8;
    mandatory true;
    description
    "The next header of the IPv6 header."
}
leaf traffic-class {
    type uint8;
    description
    "The traffic class value of the header."
}
leaf flow-label {
    type inet:ipv6-flow-label;
    description
    "The flow label of the header."
}
leaf hop-limit {
    type uint8 {
        range "1..255";
    }
    description
    "The hop limit of the header."
}
}
grouping mpls-header {

description
"The MPLS encapsulation header information."
}
list label-operations {
  key "label-oper-id";
  description
   "Label operations.";
  leaf label-oper-id {
    type uint32;
    description
     "An optional identifier that points
     to a label operation.";
  }
}

choice label-actions {
  description
   "Label action options.";
  case label-push {
    container label-push {
      description
       "Label push operation.";
      leaf label {
        type uint32;
        mandatory true;
        description
         "The label to be pushed.";
      }
      leaf s-bit {
        type boolean;
        description
         "The s-bit of the label to be pushed.";
      }
      leaf tc-value {
        type uint8;
        description
         "The traffic class value of the label
         to be pushed.";
      }
      leaf ttl-value {
        type uint8;
        description
         "The TTL value of the label to be
         pushed.";
      }
    }
  }
  case label-swap {
    container label-swap {
      description
       "Label swap operation.";
      leaf out-label {
        type uint32;
      }
    }
  }
}
mandatory true;
    description
        "The out MPLS label."
    }
leaf ttl-action {
    type ttl-action-def;
    description
        "The label ttl actions:
        - No-action, or
        - Copy to inner label, or
        - Decrease (the in label) by 1 and
          copy to the out label."
    }
}

grouping mpls-detnet-header {
    description
        "The MPLS DetNet encapsulation header information.";
    leaf service-label {
        type uint32;
        mandatory true;
        description
            "The service label of the DetNet header."
    }
    leaf control-word {
        type uint32;
        description
            "The control word of the DetNet header."
    }
}

grouping udp-header {
    description
        "UDP header.";
    leaf source-port {
        type inet:port-number;
        description
            "The source port number."
    }
    leaf destination-port {
        type inet:port-number;
        description
            "The destination port number."
    }
}
grouping transport-tunnel-encap {
  description "Defines the transport tunnel encapsulation header.";
  choice tunnel-type {
    description "Tunnel type includes: IPv4, IPv6, MPLS, MPLS over UDP tunnels.";
    case IPv4 {
      if-feature ipv4-tunnel;
      description "IPv4 tunnel.";
      container ipv4-encapsulation {
        description "IPv4 encapsulation.";
        uses ipv4-header;
      }
    }
    case IPv6 {
      if-feature ipv6-tunnel;
      description "IPv6 tunnel.";
      container ipv6-encapsulation {
        description "IPv6 encapsulation.";
        uses ipv6-header;
      }
    }
    case MPLS {
      if-feature mpls-tunnel;
      description "MPLS tunnel.";
      container mpls-encapsulation {
        description "MPLS encapsulation.";
        uses mpls-header;
      }
    }
    case MPLS-over-UDP {
      if-feature mpls-over-udp-tunnel;
      description "MPLS over UDP tunnel.";
      container mpls-over-udp-encapsulation {
        description "MPLS over udp encapsulation.";
      }
    }
  }
}
uses mpls-header;
uses udp-header;
choice address-family {
  description
  "According to IP address family (IPv4 and IPv6)
  to apply corresponding IP header.";
  case IPv4 {
    description
    "IPv4 address family.";
    uses ipv4-header;
  }
  case IPv6 {
    description
    "IPv6 address family.";
    uses ipv6-header;
  }
}
}
}


grouping transport-tunnel-decap {
  description
  "Tunnel decapsulation information.";
  choice tunnel-type {
    description
    "Next hop tunnel type options.";
    case ipv4 {
      if-feature ipv4-tunnel;
      container ipv4-decap {
        description
        "IPv4 decap.";
        leaf ipv4-decap {
          type tunnel-decap-action-def;
          mandatory true;
          description
          "IPv4 decap operations.";
        }
        leaf ttl-action {
          type ttl-action-def;
          description
          "The ttl actions:
          no-action or copy to inner header.";
        }
      }
    }
  }
}
case ipv6 {
  if-feature ipv6-tunnel;
  container ipv6-decap {
    description "IPv6 decap.";
    leaf ipv6-decap {
      type tunnel-decap-action-def;
      mandatory true;
      description "IPv6 decap operations.";
    }
    leaf hop-limit-action {
      type hop-limit-action-def;
      description "The hop limit actions:
          no-action or copy to inner header.";
    }
  }
}
case mpls {
  if-feature mpls-tunnel;
  container label-pop {
    description "MPLS decap.";
    leaf label-pop {
      type mpls-label-action-def;
      mandatory true;
      description "Pop a label from the label stack.";
    }
    leaf ttl-action {
      type ttl-action-def;
      description "The label ttl actions:
          no-action or copy to inner label/header.";
    }
  }
}
}

grouping detnet-transport-instance {
  description "An instance of the DetNet transport layer, which
  depends on the specific data plane that is used
  as the underlay tunnel.";
  uses transport-tunnel-encap;
  uses detnet-transport-qos;
grouping ipv6-flow-identification {
  description
    "IPv6 flow identification.";
  leaf src-ipv6-address {
    type inet:ipv6-address;
    description
      "The source IP address of the header.";
  }
  leaf dest-ipv6-address {
    type inet:ipv6-address;
    description
      "The destination IP address of the header.";
  }
  leaf traffic-class {
    type uint8;
    description
      "The traffic class value of the header.";
  }
  leaf flow-label {
    type inet:ipv6-flow-label;
    description
      "The flow label of the header.";
  }
  leaf source-port {
    type inet:port-number;
    description
      "The source port number.";
  }
  leaf destination-port {
    type inet:port-number;
    description
      "The destination port number.";
  }
  leaf protocol {
    type uint8;
    description
      "The protocol id of the header.";
  }
}


grouping ipv4-flow-identification {
  description
    "IPv4 flow identification.";
  leaf src-ipv4-address {
    type inet:ipv4-address;
    description
      "The source IP address of the header.";
  }
  leaf dest-ipv4-address {
    type inet:ipv4-address;
    description
      "The destination IP address of the header.";
  }
  leaf traffic-class {
    type uint8;
    description
      "The traffic class value of the header.";
  }
  leaf flow-label {
    type inet:ipv6-flow-label;
    description
      "The flow label of the header.";
  }
  leaf source-port {
    type inet:port-number;
    description
      "The source port number.";
  }
  leaf destination-port {
    type inet:port-number;
    description
      "The destination port number.";
  }
  leaf protocol {
    type uint8;
    description
      "The protocol id of the header.";
  }
}
"The source IP address of the header of a DetNet flow."
}
leaf dest-ipv4-address {
    type inet:ipv4-address;
    description
        "The destination IP address of the header of a DetNet flow.";
}
leaf dscp {
    type uint8;
    description
        "The DSCP field of the header of a DetNet flow.";
}
leaf source-port {
    type inet:port-number;
    description
        "The source port number.";
}
leaf destination-port {
    type inet:port-number;
    description
        "The destination port number.";
}
leaf protocol {
    type uint8;
    description
        "The protocol id of the header of a DetNet flow.";
}
}
grouping ip-flow-identification {
    description
        "IP flow identification.";
    choice ip-flow-type {
        description
            "IP flow types: IPv4, IPv6.";
        case ipv4 {
            description
                "IPv4 flow identification.";
            leaf src-ipv4-address {
                type inet:ipv4-address;
                description
                    "The source IP address of the header.";
            }
            leaf dest-ipv4-address {
                type inet:ipv4-address;
                description
                    "The destination IP address of the header.";
            }
            leaf dscp {
                type uint8;
                description
                    "The DSCP field of the header of a DetNet flow.";
            }
            leaf source-port {
                type inet:port-number;
                description
                    "The source port number.";
            }
            leaf destination-port {
                type inet:port-number;
                description
                    "The destination port number.";
            }
            leaf protocol {
                type uint8;
                description
                    "The protocol id of the header of a DetNet flow.";
            }
        }
    }
"The destination IP address of the header."
}
leaf dscp {
  type uint8;
  description
    "The DSCP field of the header.";
}
}

case ipv6 {
  description
    "IPv6 flow identification."
leaf src-ipv6-address {
  type inet:ipv6-address;
  description
    "The source IP address of the header.";
}
leaf dest-ipv6-address {
  type inet:ipv6-address;
  description
    "The destination IP address of the header.";
}
leaf traffic-class {
  type uint8;
  description
    "The traffic class value of the header.";
}
leaf flow-label {
  type inet:ipv6-flow-label;
  description
    "The flow label of the header.";
}
}
leaf source-port {
  type inet:port-number;
  description
    "The source port number.";
}
leaf destination-port {
  type inet:port-number;
  description
    "The destination port number.";
}
leaf protocol {
  type uint8;
  description
    "The protocol id of the header.";
}
grouping l3-flow-identification {
    description
    "Layer 3 flow identification in a DetNet domain.";
    choice flow-type {
        description
        "L3 DetNet flow types: IP and MPLS.";
        case IP {
            description
            "IP (IPv4 or IPv6) flow identification.";
            uses ip-flow-identification;
        }
        case MPLS {
            description
            "MPLS flow identification.";
            leaf service-label {
                type uint32;
                mandatory true;
                description
                "The service label of a DetNet flow.";
            }
        }
    } //l3-flow-identification
}

grouping in-segments {
    description
    "From a receiving node point of view, In-segments are a set of instances of a DetNet flow at the receiving node. This occurs when Packet Replication Function (PRF) is enabled at an upstream node or multiple flows map/aggregate to a single DetNet flow.";
    list in-segment {
        key "in-segment-id";
        description
        "A list of in segments, there will be multiple in-segments for a DetNet flow when PRF and PEF enabled.";
        leaf in-segment-id {
            type uint32;
            description
            "in-segment identifier.";
        }
    }
} //in-segments
uses l3-flow-identification;

leaf service-function {
  type service-function-type;
  description
    "DetNet service function indication.";
}
}

grouping out-segments {
  description
    "Out-segments are a set of instances of a DetNet flow, this occurs when implement packet replication function, where an in-segment of a DetNet flow is replicated to multiple out-segments.";
}

list out-segment {
  key "out-segment-id";
  description
    "A list of segments, there will be multiple out-segments when perform PRF.";
  leaf out-segment-id {
    type uint32;
    description
      "The out-segment identifier";
  }
}

container detnet-service-encapsulation {
  description
    "Only MPLS based DetNet defines DetNet service layer. The service encapsulation includes service label and control word.";
  uses mpls-detnet-header;
}

container detnet-transport-encapsulation {
  description
    "Each out-segment corresponds to a transport instance.";
  uses detnet-transport-instance;
}
}

grouping detnet-service-instance {
  description
"An end-2-end DetNet service is consisted of multiple segments. The concept of segment is similar to PW segment. For DetNet, since the existing of PREOF, there could be three cases:
1 - One in-segment maps to multiple out-segments, when implement PRF;
2 - Multiple in-segments map to one out-segment, when implement PEF;
3 - Multiple in-segments map to multiple out-segments, when implement a combination of PEF and PRF."

leaf name {
  type string;
  description "The name of the service instance. This MUST be unique across all service instances in a given network device.";
}

leaf service-rank {
  type boolean;
  description "Service rank is used by the network to decide which services can and cannot exist when network resources reach their limit. Rank is used to help determine which services can be dropped (i.e., removed from node configuration) if a port of a node becomes oversubscribed (e.g., due to network reconfiguration). The true value is more important than the false value (i.e., services with false are dropped first).";
  reference "draft-ietf-detnet-flow-information-model";
}

uses in-segments;
uses out-segments;

grouping l2-flow-identification-at-uni {
  description "Layer 2 flow identification at UNI.";
  leaf source-mac-address {
    type yang:mac-address;
    description "The source MAC address used for flow identification.";
  }
}
leaf destination-mac-address {
  type yang:mac-address;
  description
    "The destination MAC address used for
    flow identification.";
}

leaf ethertype {
  type eth:ethertype;
  description
    "The Ethernet Type (or Length) value represented
    in the canonical order defined by IEEE 802.
    The canonical representation uses lowercase
    characters.";
  reference
    "IEEE 802-2014 Clause 9.2";
}

leaf vlan-id {
  type uint16 {
    range "1..4094";
  }
  description
    "Vlan Identifier used for L2 flow identification.";
}

container pcp {
  //Todo
  description
    "PCP used for L2 flow identification.";
}

grouping l3-flow-identification-at-uni {
  description
    "Layer 3 flow identification at UNI.";
  uses ip-flow-identification;
}

grouping traffic-specification {
  description
    "traffic-specification specifies how the Source
    transmits packets for the flow. This is the
    promise/request of the Source to the network.
    The network uses this traffic specification
    to allocate resources and adjust queue
    parameters in network nodes.";
  reference

leaf interval {
  type uint32;
  description
      "The period of time in which the traffic
      specification cannot be exceeded";
}

leaf max-packets-per-interval {
  type uint32;
  description
      "The maximum number of packets that the
      source will transmit in one Interval.";
}

leaf max-payload-size {
  type uint32;
  description
      "The maximum payload size that the source
      will transmit.";
}

leaf average-packets-per-interval {
  type uint32;
  description
      "The average number of packets that the
      source will transmit in one Interval";
}

leaf average-payload-size {
  type uint32;
  description
      "The average payload size that the source
      will transmit.";
}

grouping client-flows-at-uni {
  description
      "The attributes of the client flow at UNI. When
      flow aggregation is enabled at ingress, multiple
      client flows map to a DetNet service instance.";
  list client-flow {
    key "flow-id";
    description
      "A list of client flows.";
    leaf flow-id {
      type uint32;
      description
        "Flow identifier that is unique in a network
        device for client flow identification";
  }
leaf flow-rank {
    type boolean;
    description
    "Flow rank is used by the network
    to decide which flows can and cannot exist
    when network resources reach their limit. Rank is used to help to determine which flows
    can be dropped (i.e., removed from node
    configuration) if a port of a node becomes
    oversubscribed (e.g., due to network
    reconfiguration). The true value is more
    important than the false value (i.e., flows
    with false are dropped first).";
    reference
    "draft-ietf-detnet-flow-information-model";
}
choice flow-type {
    description
    "Client flow type: layer 2 flow, layer 3
    flow.";
    case l2-flow {
        description
        "Ethernet flow identification.";
        uses l2-flow-identification-at-uni;
    }
    case l3-flow {
        description
        "Layer 3 flow identification, including
        IPv4, IPv6 and MPLS.";
        uses l3-flow-identification-at-uni;
    }
}
container traffic-specification {
    description
    "The traffic specification of the client flow.";
    uses traffic-specification;
}
}

grouping detnet-service-decap {
    description
    "DetNet service decapsulation information.";
    leaf service-label-pop {
        type mpls-label-action-def;
        mandatory true;
        description
        "Service label pop information.";
    }
}

Geng, et al. Expires September 27, 2019
"Pop the DetNet service label."

leaf ttl-action {
  type ttl-action-def;
  description
    "The label ttl actions:
    no-action or copy to inner label/header.";
}

{
}

grouping detnet-service-proxy-instance {
  description
    "Mapping between App-flows and DetNet flows."
}

choice edge-node-type {
  description
    "There are two types of edge node: ingress node and
    egress node";
  case ingress-node {
    uses client-flows-at-uni;
    leaf service-function {
      type service-function-type;
      description
        "DetNet service function indication.";
    }
    uses detnet-sequence-number;
    uses out-segments;
  }
  case egress-node {
    uses in-segments;
    uses transport-tunnel-decap;
    uses detnet-service-decap;
  }
}

}

container detnet-flow {
  description
    "DetNet flow configuration and status reporting.";
  choice detnet-node-role{
    description
      "Depends on the role of a node to configure
      corresponding flow parameters.";
    case transit-node {
      description
        "DetNet flow configuration parameters for
        transit nodes.";
      container transit-node {
There are some open issues that are still under discussion:

- The Relationship with 802.1 TSN YANG models is TBD. TSN YANG models include: P802.1Qcw, which defines TSN YANG for Qbv, Qbu,
and Qci, and P802.1CBcv, which defines YANG for 802.1CB. The possible problem here is how to avoid possible overlap among yang models defined in IETF and IEEE. A common YANG model may be defined in the future to shared by both TSN and DetNet. More discussion are needed here.

- How to support DetNet OAM is TBD.

These issues will be resolved in the following versions of the draft.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

<TBD>

9. Acknowledgements

10. References

10.1. Normative References

[I-D.finn-detnet-bounded-latency]

[I-D.ietf-detnet-architecture]

[I-D.ietf-detnet-dp-sol-ip]

[I-D.ietf-detnet-dp-sol-mpls]


10.2. Informative References


[I-D.varga-detnet-service-model]

[IEEE802.1CB]

[IEEE802.1Q-2014]

[IEEE802.1Qbu]

[IEEE802.1Qbv]

[IEEE802.1Qcc]

[IEEE802.1Qch]

[IEEE802.1Qci]


Authors’ Addresses

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

Reshad Rahman
Cisco Systems

Email: rrahaman@cisco.com