Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

X. Geng
M. Chen
Huawei
Z. Li
China Mobile
R. Rahman
Cisco Systems

October 22, 2018

DetNet Configuration YANG Model
draft-ietf-detnet-yang-00

Abstract

This document defines a YANG data model for Deterministic Networking (DetNet), which includes the DetNet topology YANG module, DetNet flow configuration YANG module and DetNet Transport QoS YANG Module.

The YANG modules in this document conform 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

Table of Contents

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:

o 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).

o 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.

o 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        +----+
              |    | <----------------> | X |    management/control
              +----+                   +-+-+       plane entity
                                         ^
                                         |   configuration
                                         |      model
                                  +------------+
                                  v      |     |
                                 +-+     |     v  network
                                 +-+     v    +-+  nodes
                                        +-+   +-+
                                        +-+
```

Figure 1. Three Information Models

This document defines DetNet configuration YANG [RFC7950] [RFC6991]data models, which include:

o DetNet topology model

DetNet topology model is designed for DetNet topology/ capability discovery and device configuration, it is an augmentation of the ietf-te-toplogy model [I-D.ietf-teas-yang-te-topo]. The detail of DetNet topology model is defined in Section 3.

o DetNet flow configuration model

DetNet flow model is designed for DetNet flow path configuration and flow status reporting. The detail of DetNet flow configuration model is defined in Section 4.

o DetNet transport QoS model

DetNet transport QoS model is designed for QoS attributes configuration of transport tunnels to achieve end-to-end bounded latency and zero congestion loss. The detail of DetNet transport QoS model is defined in Section 5.

2. Terminologies

This documents uses the terminologies defined in [I-D.ietf-detnet-architecture].

3. DetNet Topology Model

A DetNet topology is composed of a set of DetNet nodes and DetNet links. DetNet nodes represent the network devices that can transport DetNet services, which are connected by DetNet links. A DetNet Link Terminate Point(LTP) is the connection point between a DetNet node and a DetNet link, which represents the port or interface of a network node. The concept of DetNet node/link/LTP are similar as TE node/link/LTP that are defined in [I-D.ietf-teas-yang-te-topo].

Figure 2 shows a simple DetNet topology: A is a DetNet node, B is DetNet a LTP, and C is a DetNet link.

```
                        +---+          +---+
                        | A |o(B)--(C)--|   |
                        +---+          +---+
```

Figure 2. An example of DetNet Topology

DetNet topology model (ietf-detnet-topology) augments ietf-te-topology model [I-D.ietf-teas-yang-te-topo] to cover the following

attributes, which are necessary for supporting DetNet congestion protection and service protection functions:

- o Bandwidth related attributes (e.g., bandwidth reserved for DetNet);
- o Buffer/queue management related attributes (e.g., queue management algorithm, etc.);
- o PREOF (Packet Replication, Ordering and Elimination Function) capabilities and parameters (e.g., maximum out-of-order packets, etc.);
- o Delay related attributes (e.g., node processing delay, queuing delay, link delay, etc.);

The above attributes are categorized into three types: node attributes, link attributes and LTP attributes. The detailed descriptions and model definitions are specified in section 4.1, 4.2 and 4.3, respectively.

3.1. DetNet Node Attributes

Section 4.3 of [I-D.finn-detnet-bounded-latency] gives a DetNet time model, which defines that the delay within a node includes five parts: processing delay, regulation delay, queuing delay, output delay and preemption delay. The processing delay, queuing delay and regulation delay are variable in general, but for DetNet, these delays should be bounded, which is the basic assumption of deterministic networking. These bounded delay parameters are necessary to perform DetNet path computation. Among this delay attributes, processing delay and regulation delay are node relevant, and the queuing delay is LTP relevant. In addition, in order to simplify the model and implementation, the processing delay and regulation delay are combined as processing delay, and the preemption delay is included in queuing delay. [Editor notes: more comments and inputs need here].

For the DetNet node attributes, the following variables are introduced:

- o Maximum DetNet packet processing delay
- o Minimum DetNet packet processing delay
- o Maximum DetNet packet processing delay variation

The modeling structure is shown below:

```
  augment /nw:networks/nw:network/nw:node/tet:te/tet:te-node-attributes:
     +--rw detnet-node-attributes
        +--rw minimum-packet-processing-delay?             uint32
        +--rw maximum-packet-processing-delay?             uint32
        +--rw maximum-packet-processing-delay-variation?   uint32
```

3.2. DetNet Link Attributes

DetNet link attributes include link delay and link bandwidth for DetNet. This document introduces the following link related attributes:

o Link delay: link delay is a constant that only depends on the physical connection. It has been defined in ietf-te-topology [I-D.ietf-teas-yang-te-topo], and DetNet can reuse it directly.

o Maximum DetNet reservable bandwidth: the maximum reservable bandwidth that is allocated to DetNet. For a 10G link, if 50% of the bandwidth is allocated to DetNet, then the maximum DetNet reservable bandwidth is 5G. That means there are 5G bandwidth that can be used by DetNet flows.

o Reserved DetNet bandwidth: the bandwidth that has been reserved for DetNet flows.

o Available DetNet bandwidth: the bandwidth that is available for new DetNet flows.

The DetNet link attributes are modeled within a link, and the YANG module structure is shown below:

```
  augment /nw:networks/nw:network/nt:link/tet:te/tet:te-link-attributes:
     +--rw detnet-link-attributes
        +--rw maximum-reservable-bandwidth
        |  +--rw te-bandwidth
        |     +--rw (technology)?
        |        +--:(generic)
        |           +--rw generic?   te-bandwidth
        +--rw reserved-detnet-bandwidth
        |  +--rw te-bandwidth
        |     +--rw (technology)?
        |        +--:(generic)
        |           +--rw generic?   te-bandwidth
        +--rw available-detnet-bandwidth
           +--rw te-bandwidth
              +--rw (technology)?
                 +--:(generic)
                    +--rw generic?   te-bandwidth
```

3.3. DetNet Link Terminate Point Attributes

The concept of LTP is introduced in [I-D.ietf-teas-yang-te-topo], and this section introduces attributes for DetNet LTP.

PREOF (Packet Replication/Elimination/Ordering Function) is for DetNet service protection, which includes :

- o In-order delivery function: defined in Section 3.2.2.1 of [I-D.ietf-detnet-architecture]
- o Packet replication function: defined in Section 3.2.2.2 of [I-D.ietf-detnet-architecture]
- o Packet elimination function: defined in Section 3.2.2.3 of [I-D.ietf-detnet-architecture]

The above functions are modeled as a set of capabilities and relevant parameters, which are listed below:

- o in-order-capability: indicates whether a LTP has the in-order delivery capability.
- o maximum-number-of-out-of-order-packets: indicates the maximum number of out-of-order packets that an LTP can support, it depends on the reserved buffer size for packet reordering.
- o replication-capability: indicates whether a LTP has the packet replication capability.
- o elimination-capability: indicates whether a LTP has the packet elimination capability.

In addition, DetNet LTP also includes queuing management algorithms and queuing delay attributes. In the context of DetNet, the delay of queuing is bounded, and the bound depends on what queuing management method is used and how many buffers are allocated. More information can be found in [I-D.finn-detnet-bounded-latency]. Queuing related attributes are listed below:

- o queuing-algorithm-capabilities: it is modeled as a list that includes all queuing algorithms that a LTP supports.
- o detnet-queues: it's modeled as a list that includes all queues of a DetNet LTP. For each queue, it has the following attributes:
- o queue-identifier: an identifier of a queue. It could be an internal identifier that is only used within a node. Or it could

be used by a centralized controller to specify in which specific queue a flow/packet is required to enter.

o queue-buffer-size: the size of a queue with unit of bytes.

o enabled-queuing-algorithm: indicates what queuing management algorithm is enabled.

o maximum-queuing-delay: the maximum queuing delay that a packet will undergo when transmitted through the queue.

o minimum-queuing-delay: the minimum queuing delay that a packet will undergo when transmitted through the queue.

o maximum-queuing-delay-variation: the maximum queuing delay variation that a packet will undergo when transmitted the queue.

The DetNet LTP attributes are modeled with a LTP, the YANG module structure is shown below:

```
augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te:
   +--rw detnet-terminate-point-attributes
      +--rw elimination-capability?           boolean
      +--rw replication-capability?           boolean
      +--rw in-ordering-capability
      |  +--rw in-ordering-capability?         boolean
      |  +--rw maximum-out-of-order-packets?   uint32
      +--rw queuing-algorithm-capabilities
      |  +--rw credit-based-shaping?            boolean
      |  +--rw time-aware-shaping?              boolean
      |  +--rw cyclic-queuing-and-forwarding?   boolean
      |  +--rw asynchronous-traffic-shaping?    boolean
      +--rw queues* [queue-identifier]
         +--rw queue-identifier                   uint32
         +--rw queue-buffer-size?                 uint32
         +--rw enabled-queuing-algorithm
         |  +--rw credit-based-shaping?            boolean
         |  +--rw time-aware-shaping?              boolean
         |  +--rw cyclic-queuing-and-forwarding?   boolean
         |  +--rw asynchronous-traffic-shaping?    boolean
         +--rw minimum-queuing-delay?             uint32
         +--rw maximum-queuing-delay?             uint32
         +--rw maximum-queuing-delay-variation?   uint32
```

4. DetNet Flow 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 4.1, 4.2, 4.3, respectively. Section 4.4 gives a simple example on how to use these attributes for DetNet flow configuration.

4.1. 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:

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

- o 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 misbehaviored flows from going into the network, which needs:

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

The YANG module structure is shown below:

```
    +--rw client-flow* [flow-id]
    |  +--rw flow-id                     uint32
    |  +--rw (flow-type)?
    |  |  +--:(l2-flow-identfication)
    |  |  |  +--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-identification)
    |  |     +--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
    |  +--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
```

4.2. 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:

o DetNetwork flow identification, refers to Section 7.1.3 of [I-D.ietf-detnet-flow-information-model].

o 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.)

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

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

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

   o 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:

```
  |  +--rw relay-node
  |     +--rw name?           string
  |     +--rw flow-rank
  |     +--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)
  |           |  |  +--rw ipv4-encaplustion
  |           |  |     +--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 ipv6-encaplustion
 |           |  |     +--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)
 |           |     +--rw mpls-encaplustion
 |           |        +--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
 |           +--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
```

4.3. 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:

o 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.

o 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:

```
|  +--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
```

The parameters for DetNet transport QoS are defined in Section 5.

4.4. DetNet Flow Configuration Example

This section gives an example about how to implement an end-2-end DetNet service with the collaboration of DetNet proxy, service and transport layers.

To simplify the explanation, several terms are introduced. This document defines DetNet Service Proxy Instance (DSPI), DetNet Service Instance (DSI) and DetNet Transport Instance for end-to-end DetNet flow configuration as showed in Figure 4. DSPI 1 at Edge Node 1 (E1) maps an application flow to a DetNet Flow (DF1), which is transmitted over a DetNet tunnel (Tn1). In DSI 2 of Relay Node 1 (R1), DetNet Flow 1(DF1) was replicated into two member flows: DetNet Flow 2 (DF2) transmitted by DetNet Tunnel 2 (Tnl2) and DetNet Flow 3 (DF3) by DetNet Tunnel 3(Tnl3). In DSPI 3 of Edge Node 2 (E2), DetNet Flow 2 (DF2) and DetNet Flow 3(DF3) were merged and mapped to application flow used by CE2.

DF: DetNet Flow
DSPI: DetNet Service Proxy Instance
DSI: DetNet Service Instance
DTI: DetNet Transport Instance
Tnl: Tunnel

```
                |<------- End to End DetNet Service ------>|
                |          DetNet           DetNet         |
         (AC)   |        |<-Tnl->|        |<-Tnl->|        |  (AC)
    End    |    V        V   1   V        V  2/3  V        V   |    End
    System |    +--------+       +--------+       +--------+   |  System
    +---+  |    |   E1   |=======|   R1   |=======|   E2   |   |   +---+
    |   |--|----|--------|       |--------|       |--------|---|---|   |
    |CE1|  |    | DSPI 1 |       |        |       | DSPI 2 |   |   |CE2|
    |   |       |+-------+       |        |       +-------+|       |   |
    +---+       || DSI 1 |       |  DSI 2 |       | DSI 3 ||       +---+
                ||       +       | +-------+      |       ||
                || +-----+       | |DTI 2 |..DF2..|       ||
                || |DTI 1|..DF1..| +-------+      |       ||
                || +-----+       | |DTI 3 |..DF3..|       ||
                |+-------+       | +-------+      +-------+|
                +--------+=======+--------+=======+--------+
                 Edge Node          Relay Node       Edge Node
                    |                                   |
                    |<-------- DetNet Service --------->|
```

Figure 3. End-to-end DetNet Flow Configuration

5. DetNet Transport QoS Model

The YANG data model of transport QoS is very important to achieve end-to-end bounded latency and zero congestion loss. There are three possible methods to deal with the DetNet transport QoS YANG:

1. DetNet service is not aware of any QoS/queuing/bounded-latency information, and all relative parameters are defined in separate YANG models;

2. DetNet service is not aware of any of Qos/queuing/bounded-latency information, but it should maintain an interface to the corresponding YANG models;

3. DetNet service should be aware of the Qos/queuing/bounded-latency information, because some Qos/queuing/bounded-latency mechanisms are required to be configured with flow information;

How to define transport QoS YANG is still under discussion and the transport QoS YANG model is not included in the current version of the draft.

[Editor notes: more comments and inputs need here].

6. DetNet Yang Structure

6.1. DetNet Topology Model Tree Diagram

```
  module: ietf-detnet-topology
  augment /nw:networks/nw:network/nw:network-types/tet:te-topology:
     +--rw detnet-topology!
  augment /nw:networks/nw:network/nw:node/tet:te/tet:te-node-attributes:
     +--rw detnet-node-attributes
        +--rw minimum-packet-processing-delay?             uint32
        +--rw maximum-packet-processing-delay?             uint32
        +--rw maximum-packet-processing-delay-variation?   uint32
  augment /nw:networks/nw:network/nt:link/tet:te/tet:te-link-attributes:
     +--rw detnet-link-attributes
        +--rw maximum-reservable-bandwidth
        |  +--rw te-bandwidth
        |     +--rw (technology)?
        |        +--:(generic)
        |           +--rw generic?   te-bandwidth
        +--rw reserved-detnet-bandwidth
        |  +--rw te-bandwidth
        |     +--rw (technology)?
        |        +--:(generic)
        |           +--rw generic?   te-bandwidth
        +--rw available-detnet-bandwidth
           +--rw te-bandwidth
              +--rw (technology)?
                 +--:(generic)
                    +--rw generic?   te-bandwidth
  augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te:
     +--rw detnet-terminate-point-attributes
        +--rw elimination-capability?           boolean
        +--rw replication-capability?           boolean
        +--rw in-ordering-capability
        |  +--rw in-ordering-capability?         boolean
        |  +--rw maximum-out-of-order-packets?   uint32
        +--rw queuing-algorithm-capabilities
        |  +--rw credit-based-shaping?            boolean
        |  +--rw time-aware-shaping?              boolean
        |  +--rw cyclic-queuing-and-forwarding?   boolean
        |  +--rw asynchronous-traffic-shaping?    boolean
        +--rw queues* [queue-identifier]
           +--rw queue-identifier                  uint32
           +--rw queue-buffer-size?                uint32
           +--rw enabled-queuing-algorithm
           |  +--rw credit-based-shaping?            boolean
           |  +--rw time-aware-shaping?              boolean
           |  +--rw cyclic-queuing-and-forwarding?   boolean
           |  +--rw asynchronous-traffic-shaping?    boolean
           +--rw minimum-queuing-delay?            uint32
           +--rw maximum-queuing-delay?            uint32
           +--rw maximum-queuing-delay-variation?   uint32
```

6.2. DetNet Flow Configuration Model Tree Diagram

```
module: ietf-detnet-flow-config
   +--rw detnet-flow
      +--rw (detnet-node-role)?
         +--:(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)
 |  +--rw relay-node
 |     +--rw name?           string
 |     +--rw flow-rank
 |     +--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)
 |           |  |  +--rw ipv4-encaplustion
 |           |  |     +--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 ipv6-encaplustion
 |           |  |     +--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)
 |           |     +--rw mpls-encaplustion
 |           |        +--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
 |           +--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)
         |  +--rw edge-node
         |     +--rw client-flow* [flow-id]
         |     |  +--rw flow-id                    uint32
         |     |  +--rw (flow-type)?
         |     |  |  +--:(l2-flow-identfication)
         |     |  |  |  +--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-identification)
         |     |  |     +--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
         |     |  +--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 detnet-service-instance
         |        +--rw name?           string
         |        +--rw flow-rank
         |        +--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)
         |              |  |  +--rw ipv4-encaplustion
         |              |  |     +--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 ipv6-encaplustion
         |              |  |     +--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)
         |              |     +--rw mpls-encaplustion
         |              |        +--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
         |              +--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
         +--:(end-station)
+--rw end-station
   +--rw client-flow* [flow-id]
   |  +--rw flow-id                    uint32
   |  +--rw (flow-type)?
   |  |  +--:(l2-flow-identfication)
   |  |  |  +--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-identification)
   |  |     +--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
   |  +--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 detnet-service-instance
                  +--rw name?           string
                  +--rw flow-rank
                  +--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)
```

```
           |  |  +--rw ipv4-encaplustion
```

```
                        |  |     +--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 ipv6-encaplustion
                        |  |     +--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)
                        |     +--rw mpls-encaplustion
                        |        +--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
                        +--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
```

7. DetNet YANG Model

7.1. DetNet Topology YANG Model

```
  <CODE BEGINS> file "ietf-detnet-topology@20180910.yang"
    module ietf-detnet-topology {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-topology";
    prefix "detnet-topology";

    import ietf-te-types {
      prefix "te-types";
    }
    import ietf-te-topology {
      prefix "tet";
    }

    import ietf-network {
      prefix "nw";
    }

    import ietf-network-topology {
      prefix "nt";
    }

    organization
      "IETF Deterministic Networking(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 augments the 'ietf-te-topology'
       module with DetNet related capabilities and
       parameters.";

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

```
    grouping detnet-queuing-algorithms {
      description
        "Relationship with IEEE 802.1 TSN YANG models is TBD.";
    }

    grouping detnet-node-attributes{
      description
        "DetNet node related attributes.";
      leaf minimum-packet-processing-delay{
        type uint32;
        description
          "Minimum packet processing delay
           in a node. The unit of the delay
           is microsecond(us)";
      }
      leaf maximum-packet-processing-delay{
        type uint32;
        description
          "Maximum packet processing delay
           in a node. The unit of the delay
           is microsecond(us)";
      }
      leaf maximum-packet-processing-delay-variation{
        type uint32;
        description
          "Maximum packet processing delay
           variation in a node. The unit of
           the delay variation is microsecond(us)";
      }
    }

    grouping detnet-link-attributes{
      description
        "DetNet link related attributes.";

      container maximum-reservable-bandwidth{
        uses te-types:te-bandwidth;
        description
          "This container specifies the maximum bandwidth
           that is reserved for DetNet on this link.";
      }

      container reserved-detnet-bandwidth{
        uses te-types:te-bandwidth;
        description
          "This container specifies the bandwidth that has
           been reserved for DetNet on this link.";
      }
```

```
        container available-detnet-bandwidth{
          uses te-types:te-bandwidth;
          description
            "This container specifies the bandwidth that is
             available for new DetNet flows on this link.";
        }
      }

      grouping detnet-terminate-point-attributes{
        description
          "DetNet terminate point related attributes.";

        leaf elimination-capability{
          type boolean;
          description
            "Indicates whether a node is able to do packet
             elimination.";
          reference
            "Section 3.2.2.3 of
             draft-ietf-detnet-architecture";

        }
        leaf replication-capability{
          type boolean;
          description
            "Indicates whether a node is able to do packet
             replication.";
          reference
            "Section 3.2.2.2 of
             draft-ietf-detnet-architecture";
        }
        container in-ordering-capability {
          description
            "Indicates the parameters needed for
             packet in-ordering.";
          reference
            "Section 3.2.2.1 of
             draft-ietf-detnet-architecture";

          leaf in-ordering-capability {
            type boolean;
          description
            "Indicates whether a node is able to do packet
             in-ordering.";
          }
          leaf maximum-out-of-order-packets {
          type uint32;
          description
```

```
            "The maximum number of out-of-order packets.";
          }
        }

        container queuing-algorithm-capabilities {
          description
            "All queuing algorithms that a LTP supports.";
          uses detnet-queuing-algorithms;
        }

        list queues {
          key "queue-identifier";
          description
            "A list of DetNet queues.";
          leaf queue-identifier {
            type uint32;
            description
              "The identifier of the queue.";
          }
          leaf queue-buffer-size {
            type uint32;
            description
              "The size of the queue with unit of bytes.";
          }

          container enabled-queuing-algorithm {
            description
              "The queuing algorithms that are enabled on the queue.";
               uses detnet-queuing-algorithms;
          }

          leaf minimum-queuing-delay{
            type uint32;
            description
              "The minimum queuing delay of the queue.
               The unit of the delay is microsecond(us)";
          }
          leaf maximum-queuing-delay{
            type uint32;
            description
              "The maximum queuing delay of the queue.
               The unit of the delay is microsecond(us)";
          }
          leaf maximum-queuing-delay-variation{
            type uint32;
            description
              "The maximum queuing delay variation of the queue.
               The unit of the delay variation is microsecond(us)";
```

```
        }
      }
    }

     augment "/nw:networks/nw:network/nw:network-types/tet:te-topology"{
      description
        "Introduce new network type for TE topology.";
      container detnet-topology {
        presence "Indicates DetNet topology.";
        description
          "Its presence identifies the DetNet topology type";
      }
    }

    augment "/nw:networks/nw:network/nw:node/tet:te/"
            + "tet:te-node-attributes" {
        when "../../../nw:network-types/tet:te-topology/"
        + "detnet-topology:detnet-topology" {
          description
            "Augmentation parameters apply only for networks with
             DetNet topology type.";
        }
      description
        "Augmentation parameters apply for DetNet node attributes.";
      container detnet-node-attributes {
        description
          "Attributes for DetNet node.";
        uses detnet-node-attributes;
      }
    }

    augment "/nw:networks/nw:network/nt:link/tet:te/"
              + "tet:te-link-attributes" {
      when "../../../nw:network-types/tet:te-topology/"
           + "detnet-topology:detnet-topology" {
        description
          "Augmentation parameters apply only for networks with
          DetNet topology type.";
      }
      description
        "Augmentation parameters apply for DetNet link attributes.";
      container detnet-link-attributes {
        description
          "Attributes for DetNet link.";
        uses detnet-link-attributes;
      }
    }
```

```
    augment "/nw:networks/nw:network/nw:node/nt:termination-point/"
               + "tet:te" {
        when "../../../../nw:network-types/tet:te-topology/"
              + "detnet-topology:detnet-topology" {
          description
            "Augmentation parameters apply only for networks with
             DetNet topology type.";
        }
      description
        "Augmentation parameters apply for DetNet
         link termination point.";
      container detnet-terminate-point-attributes {
        description
          "Attributes for DetNet link terminate point.";
        uses detnet-terminate-point-attributes;
      }
    }
  } //topology module


 <CODE ENDS>
```

7.2. DetNet Flow Configuration YANG Model

```
  <CODE BEGINS> file "ietf-detnet-flow@20180910.yang"
  module ietf-detnet-flow-config {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-flow-config";
    prefix "detnet-flow";

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

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

    import ietf-ethertypes {
      prefix "eth";
    }

    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";
     }

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

     grouping detnet-transport-qos {
       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-definition;
                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.";
         }
         leaf control-word {
           type uint32;
           mandatory true;
           description
             "The control word of the DetNet header.";
         }
       }

       grouping transport-tunnel-encap{
         description
           "Defines the transport tunnel encapsulation
            header.";
         choice tunnel-type {
           description
           "Tunnel type includes: IPv4, IPv6, MPLS.";
           case IPv4 {
             description
               "IPv4 tunnel.";
             container ipv4-encapsulation {
               description
                 "IPv4 encapsulation.";
               uses ipv4-header;
             }
           }
           case IPv6 {
             description
               "IPv6 tunnel.";
             container ipv6-encapsulation {
               description
                 "IPv6 encapsulation.";
               uses ipv6-header;
             }
           }
           case MPLS {
             description
               "MPLS tunnel.";
             container mpls-encapsulation {
               description
                 "MPLS encapsulation.";
               uses mpls-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 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.";
           }
         }
         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 the 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.";
```

```
          }
        }
      }
    } //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.";
        }

        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.";
       }
       container flow-rank{
         description
```

```
            "TBD based on the data plane solution.";
        }
        container service-rank{
          description
            "TBD based on the data plane solution.";
        }
        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
         "draft-ietf-detnet-flow-information-model";

       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";
          }
          choice flow-type{
            description
              "Client flow type: layer 2 flow, layer 3
               flow.";
            case l2-flow-identfication {
              description
                "Ethernet flow identification.";
              uses l2-flow-identification-at-uni;
            }
            case l3-flow-identification {
              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-proxy-instance {
        description
          "Maps between App-flows and DetNet flows";
        uses client-flows-at-uni;
        container detnet-service-instance {
```

```
         description
           "A DetNet service instance.";
         uses detnet-service-instance;
       }
     }

     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 {
             description
               "transit node container.";
             uses detnet-transport-qos;
           }
         }
         case relay-node{
           description
             "DetNet flow configuration parameters for
              relay nodes.";
           container relay-node {
             description
               "Relay node container.";
             uses detnet-service-instance;
           }
         }
         case edge-node{
           description
             "DetNet flow configuration parameters for
              edge nodes.";
           container edge-node {
             description
               "Edge node container.";
             uses detnet-service-proxy-instance;
           }
         }
         case end-station {
           description
             "DetNet flow configuration parameters for
              end stations.";
```

```
           container end-station {
             description
               "End station container.";
             uses detnet-service-proxy-instance;
           }
         }
       }
     }
   }
   <CODE ENDS>
```

8. DetNet Configuration Model Classification

This section defines three classes of DetNet configuration model: fully distributed configuration model, fully centralized configuration model, hybrid configuration model, based on different network architectures, showing how configuration information exchanges between various entities in the network.

8.1. Fully Distributed Configuration Model

In a fully distributed configuration model, UNI information is transmitted over DetNet UNI protocol from the user side to the network side; then UNI information and network configuration information propagate in the network over distributed control plane protocol. For example:

1) IGP collects topology information and DetNet capabilities of network([I-D.geng-detnet-info-distribution]);

2) Control Plane of the Edge Node(Ingress) receives a flow establishment request from UNI and calculates a/some valid path(s);

3) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

Current distributed control plane protocol,e.g., RSVP-TE[RFC3209], SRP[IEEE802.1Qcc], can only reserve bandwidth along the path, while the configuration of a fine-grained schedule, e.g.,Time Aware Shaping(TAS) defined in [IEEE802.1Qbv], is not supported.

The fully distributed configuration model is not covered by this draft. It should be discussed in the future DetNet control plane work.

8.2. Fully Centralized Configuration Model

In the fully centralized configuration model, UNI information is transmitted from Centralized User Configuration (CUC) to Centralized Network Configuration(CNC). Configurations of routers for DetNet flows are performed by CNC with network management protocol. For example:

1) CNC collects topology information and DetNet capability of network through Netconf;

2) CNC receives a flow establishment request from UNI and calculates a/some valid path(s);

3) CNC configures the devices along the path for flow transmission.

8.3. Hybrid Configuration Model

In the hybrid configuration model, controller and control plane protocols work together to offer DetNet service, and there are a lot of possible combinations. For example:

1) CNC collects topology information and DetNet capability of network through IGP/BGP-LS;

2) CNC receives a flow establishment request from UNI and calculates a/some valid path(s);

3) Based on the calculation result, CNC distributes flow path information to Edge Node(Ingress) and other information(e.g. replication/elimination) to the relevant nodes.

4) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

or

1) Controller collects topology information and DetNet capability of network through IGP/BGP-LS;

2) Control Plane of Edge Node(Ingress) receives a flow establishment request from UNI;

3) Edge Node(Ingress) sends the path establishment request to CNC through PCEP;

4) After Calculation, CNC sends back the path information of the flow to the Edge Node(Ingress) through PCEP;

5) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

There are also other variations that can be included in the hybrid model. This draft can not coverer all the control plane data needed in hybrid configuration models. Every solution has there own mechanism and corresponding parameters to make it work.

Editor's Note:

1. There are a lot of optional DetNet configuration models, and different scenario in different use case can choose one of them based on its conditions. Maybe next step of the work is to pick up one or more typical scenarios and give a practical solution.

2. [IEEE802.1Qcc] also defines three TSN configuration models: fully-centralized model, fully-distributed model, centralized Network / distributed User Model. This section defines the configuration model roughly the same, to keep the design of L2 and L3 in the same structure. Hybrid configuration model is slightly different from the 'centralized Network / distributed User Model'. The hybrid configuration model intends to contain more variations.

9. Open issues

There are some open issues that are still under disccusion:

- o 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.

- o How to suppport DetNet OAM is TBD.

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

10. IANA Considerations

This document makes no request of IANA.

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

11. Security Considerations

<TBD>

12. Acknowledgements

13. References

13.1. Normative References

[I-D.finn-detnet-bounded-latency]
Finn, N., Boudec, J., Mohammadpour, E., Varga, B., and J. Farkas, "DetNet Bounded Latency", draft-finn-detnet-bounded-latency-01 (work in progress), July 2018.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-08 (work in progress), September 2018.

[I-D.ietf-detnet-dp-sol-ip]
Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-00 (work in progress), July 2018.

[I-D.ietf-detnet-dp-sol-mpls]
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-00 (work in progress), July 2018.

[I-D.ietf-detnet-flow-information-model]
Farkas, J., Varga, B., rodney.cummings@ni.com, r., Jiang, Y., and Y. Zha, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-01 (work in progress), March 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

13.2. Informative References

[I-D.geng-detnet-info-distribution]
Geng, X., Chen, M., and Z. Li, "IGP-TE Extensions for DetNet Information Distribution", draft-geng-detnet-info-distribution-02 (work in progress), March 2018.

[I-D.ietf-detnet-use-cases]
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-19 (work in progress), October 2018.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-16 (work in progress), July 2018.

[I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-18 (work in progress), June 2018.

[I-D.thubert-tsvwg-detnet-transport]
Thubert, P., "A Transport Layer for Deterministic Networks", draft-thubert-tsvwg-detnet-transport-01 (work in progress), October 2017.

[I-D.varga-detnet-service-model]
Varga, B. and J. Farkas, "DetNet Service Model", draft-varga-detnet-service-model-02 (work in progress), May 2017.

[IEEE802.1CB]
"IEEE, "Frame Replication and Elimination for Reliability (IEEE Draft P802.1CB)", 2017, <http://www.ieee802.org/1/files/private/cb-drafts/>.", 2016.

[IEEE802.1Q-2014]
"IEEE, "IEEE Std 802.1Q Bridges and Bridged Networks", 2014, <http://ieeexplore.ieee.org/document/6991462/>.", 2014.

[IEEE802.1Qbu]
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 26: Frame Preemption", 2016, <http://ieeexplore.ieee.org/document/7553415/>.", 2016.

[IEEE802.1Qbv]
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", 2015, <http://ieeexplore.ieee.org/document/7572858/>.", 2016.

[IEEE802.1Qcc]
"IEEE, "Stream Reservation Protocol (SRP) Enhancements and Performance Improvements (IEEE Draft P802.1Qcc)", 2017, <http://www.ieee802.org/1/files/private/cc-drafts/>.".

[IEEE802.1Qch]
"IEEE, "Cyclic Queuing and Forwarding (IEEE Draft P802.1Qch)", 2017, <http://www.ieee802.org/1/files/private/ch-drafts/>.", 2016.

[IEEE802.1Qci]
"IEEE, "Per-Stream Filtering and Policing (IEEE Draft P802.1Qci)", 2016, <http://www.ieee802.org/1/files/private/ci-drafts/>.", 2016.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <https://www.rfc-editor.org/info/rfc4875>.

[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>.

Authors' Addresses

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com