

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2021

X. Geng
M. Chen
Huawei Technologies
Y. Ryoo
ETRI
D. Fedyk
LabN Consulting, L.L.C.
R. Rahman
Cisco Systems
Z. Li
China Mobile
October 12, 2020

Deterministic Networking (DetNet) Configuration YANG Model draft-ietf-detnet-yang-08

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 April 15, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Terminologies	3
3.	DetNet Configuration Module	3
3.1.	DetNet Application Flow Configuration Attributes	3
3.2.	DetNet Service Sub-layer Configuration Attributes	3
3.3.	DetNet Forwarding Sub-layer Configuration Attributes	3
4.	DetNet Flow Aggregation	4
5.	DetNet YANG Structure Considerations	5
6.	DetNet Configuration YANG Structures	5
7.	DetNet Configuration YANG Model	14
8.	Open Issues	41
9.	IANA Considerations	41
10.	Security Considerations	41
11.	Acknowledgements	41
12.	References	41
12.1.	Normative References	41
12.2.	Informative References	42
	Authors' Addresses	42

[1.](#) Introduction

DetNet (Deterministic Networking) provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low packet loss rates and assured maximum end-to-end delivery latency. A description of the general background and concepts of DetNet can be found in [[RFC8655](#)].

This document defines a YANG model for DetNet based on YANG data types and modeling language defined in [[RFC6991](#)] and [[RFC7950](#)].

DetNet service, which is designed for describing the characteristics of services being provided for application flows over a network, and DetNet configuration, which is designed for DetNet flow path establishment, flow status reporting, and DetNet functions configuration in order to achieve end-to-end bounded latency and zero congestion loss, are both included in this document.

2. Terminologies

This documents uses the terminologies defined in [[RFC8655](#)].

3. DetNet Configuration Module

DetNet configuration module includes DetNet App-flow configuration, DetNet Service Sub-layer configuration, and DetNet Forwarding Sub-layer configuration. The corresponding attributes used in different sub-layers are defined in [Section 3.1](#), 3.2, 3.3 respectively.

3.1. DetNet Application Flow Configuration Attributes

DetNet application flow 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 map a flow at the User Network Interface (UNI), the corresponding attributes are defined in [[I-D.ietf-detnet-flow-information-model](#)].

3.2. DetNet Service Sub-layer Configuration Attributes

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

- o DetNet flow identification
- 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). The corresponding arguments for service functions also needs to be defined.

3.3. DetNet Forwarding Sub-layer Configuration Attributes

As defined in [[RFC8655](#)], DetNet forwarding sub-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 forwarding sub-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 be used for resource 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.

4. DetNet Flow Aggregation

DetNet provides the capability of flow aggregation to improve scalability of DetNet data, management and control planes. Aggregated flows can be viewed by some DetNet nodes as individual DetNet flows. When aggregating DetNet flows, the flows should be compatible: if bandwidth reservations are used, the reservation should be a reasonable representation of the individual reservations; if maximum delay bounds are used, the system should ensure that the aggregate does not exceed the delay bounds of the individual flows.

The DetNet YANG model defined in this document supports DetNet flow aggregation with the following functions:

- o Aggregation flow encapsulation/decapsulation/identification
- o Mapping individual DetNet flows to an aggregated flow
- o Changing traffic specification parameters for aggregated flow

The following cases of DetNet aggregation are supported:

- o aggregate data flows into an application which is then mapped to a service sub-layer at the ingress node. Note the data flows may be other DetNet flows.
- o map each DetNet application to a single service sub-layer and allowing the aggregation of multiple applications at the ingress

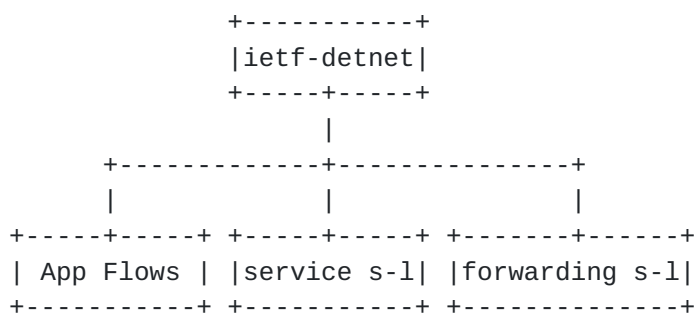
node, and vice versa for de-aggregation. A classifier may be required to de-aggregate the respective applications.

- o map each DetNet application uniquely to a single service sub-layer where those sub-layers may be encapsulated as a single service sub-layer and hence aggregating the applications at the ingress node, and vice versa for de-aggregation. In this case, the service sub-layer identifier may be sufficient to identify the application. A classifier may be required to de-aggregate the service sub-layers.
- o aggregate DetNet service sub-layers into an aggregated flow by using the same forwarding sub-layer at ingress node or relay node, and vice versa for de-aggregation.
- o aggregate DetNet flows with different forwarding sub-layer into an aggregated flow by using the same forwarding sub-layer at transit node, and vice versa for de-aggregation.

Traffic requirements and traffic specification may be tracked for individual or aggregate flows but reserving resources and tracking the services in the aggregated flow is out of scope.

5. DetNet YANG Structure Considerations

The picture shows that the general structure of the DetNet YANG Model:



There are three instances in DetNet YANG Model: App-flow instance, service sub-layer instance and forwarding sub-layer instance, respectively corresponding to four parts of DetNet functions defined in [section 3](#).

6. DetNet Configuration YANG Structures

```

module: ietf-detnet-config
  +--rw detnet
    +--rw traffic-profile* [profile-number]

```



```

|   +--rw profile-number                               uint16
|   +--rw traffic-requirements
|   |   +--rw min-bandwidth?                           uint64
|   |   +--rw max-latency?                             uint32
|   |   +--rw max-latency-variation?                   uint32
|   |   +--rw max-loss?                                uint8
|   |   +--rw max-consecutive-loss-tolerance?          uint32
|   |   +--rw max-misordering?                         uint32
|   +--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
|   +--ro member-applications*                         app-flow-ref
|   +--ro member-services*                             service-sub-layer-ref
|   +--ro member-groups*                               aggregation-grp-ref
|   +--ro member-forwarding-sublayers*                 forwarding-sub-layer-ref
+--rw app-flows
|   +--rw app-flow* [name]
|   |   +--rw name                                     string
|   |   +--rw app-flow-bidir-congruent?               boolean
|   |   +--ro outgoing-service?                       service-sub-layer-ref
|   |   +--ro incoming-service?                       service-sub-layer-ref
|   |   +--rw traffic-profile?                         traffic-profile-ref
|   |   +--rw ingress
|   |   |   +--rw name?                               string
|   |   |   +--ro app-flow-status?                    identityref
|   |   |   +--rw interface?                          if:interface-ref
|   |   |   +--rw (data-flow-type)?
|   |   |   |   +--:(tsn-app-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?                         uint8
|   |   |   |   +--:(ip-app-flow)
|   |   |   |   |   +--rw src-ip-prefix?               inet:ip-prefix
|   |   |   |   |   +--rw dest-ip-prefix?              inet:ip-prefix
|   |   |   |   |   +--rw next-header?                 uint8
|   |   |   |   |   +--rw traffic-class?               uint8
|   |   |   |   |   +--rw flow-label?                  inet:ipv6-flow-label
|   |   |   |   +--rw source-port
|   |   |   |   |   +--rw (port-range-or-operator)?
|   |   |   |   |   |   +--:(range)
|   |   |   |   |   |   |   +--rw lower-port          inet:port-number
|   |   |   |   |   |   |   +--rw upper-port          inet:port-number
|   |   |   |   |   |   +--:(operator)

```



```

| | | +--rw operator? packet-fields:operator
| | | +--rw port inet:port-number
| | +--rw destination-port
| | | +--rw (port-range-or-operator)?
| | | | +--:(range)
| | | | | +--rw lower-port inet:port-number
| | | | | +--rw upper-port inet:port-number
| | | | +--:(operator)
| | | | +--rw operator? packet-fields:operator
| | | | +--rw port inet:port-number
| | +--rw ipsec-spi? ipsec-spi
+--:(mpls-app-flow)
+--rw (label-space)?
+--:(context-label-space)
| +--rw mpls-label-stack
| +--rw entry* [id]
| | +--rw id uint8
| | +--rw label?
| | | rt-types:mpls-label
| | +--rw ttl? uint8
| | +--rw traffic-class? uint8
+--:(platform-label-space)
+--rw label? rt-types:mpls-label
+--rw egress
+--rw name? string
+--rw (application-type)?
+--:(ethernet)
| +--rw ethernet
| +--rw ethernet-place-holder? string
+--:(ip-mpls)
+--rw ip-mpls
+--rw (next-hop-options)
+--:(simple-next-hop)
| +--rw outgoing-interface?
| | if:interface-ref
| +--rw (flow-type)?
| +--:(ip)
| | +--rw next-hop-address?
| | | inet:ip-address
| +--:(mpls)
| +--rw mpls-label-stack
| +--rw entry* [id]
| | +--rw id uint8
| | +--rw label?
| | | rt-types:mpls-label
| | +--rw ttl? uint8
| | +--rw traffic-class? uint8
+--:(next-hop-list)

```



```

|               +--rw next-hop-list
|               +--rw next-hop* [hop-index]
|                   +--rw hop-index          uint8
|                   +--rw outgoing-interface?
|                       |
|                       if:interface-ref
|                   +--rw (flow-type)?
|                       +--:(ip)
|                           | +--rw next-hop-address?
|                           |     inet:ip-address
|                       +--:(mpls)
|                           +--rw mpls-label-stack
|                               +--rw entry* [id]
|                                   +--rw id
|                                       |
|                                       uint8
|                                   +--rw label?
|                                       |
|                                       rt-types:
|                                       mpls-label
|                                   +--rw ttl?
|                                       |
|                                       uint8
|                                   +--rw traffic-class?
|                                       uint8
+--rw service-aggregation-group* [group-name]
|   +--rw group-name      aggregation-group
|   +--rw outgoing
|       | +--rw traffic-profile?      traffic-profile-ref
|       | +--rw service-protection
|       | | +--rw service-protection-type?  service-protection-type
|       | | +--rw sequence-number-length?  sequence-number-field
|       | +--rw aggregation-header
|       | | +--rw mpls-label-stack
|       | | | +--rw entry* [id]
|       | | | +--rw id          uint8
|       | | | +--rw label?      rt-types:mpls-label
|       | | | +--rw ttl?        uint8
|       | | | +--rw traffic-class?  uint8
|       | +--ro services*      service-sub-layer-ref
|   +--rw incoming
|       +--rw aggregation-header
|           | +--rw mpls-label-stack
|           | | +--rw entry* [id]
|           | | +--rw id          uint8
|           | | +--rw label?      rt-types:mpls-label
|           | | +--rw ttl?        uint8
|           | | +--rw traffic-class?  uint8
|           +--ro services*      service-sub-layer-ref
+--rw service-sub-layer
|   +--rw service-sub-layer-list* [name]
|       +--rw name                string

```



```

|   +--rw service-rank?                uint8
|   +--rw (service-type)
|   |   +--:(non-grouped)
|   |   |   +--rw non-grouped
|   |   |   |   +--rw traffic-profile?          traffic-profile-ref
|   |   |   |   +--rw service-operation-type?
|   |   |   |       service-operation-type
|   |   +--:(grouped)
|   |   |   +--rw grouped
|   |   |   |   +--rw group-ref?    aggregation-grp-ref
|   +--rw service-protection
|   |   +--rw service-protection-type?    service-protection-type
|   |   +--rw sequence-number-length?    sequence-number-field
|   +--rw service-operation-type?    service-operation-type
|   +--rw incoming
|   |   +--rw (incoming-options)
|   |   |   +--:(ingress-application)
|   |   |   |   +--rw app-flow*    app-flow-ref
|   |   |   +--:(detnet-service-identification)
|   |   |   |   +--rw (detnet-flow-type)?
|   |   |   |   |   +--:(ip-detnet-flow)
|   |   |   |   |   |   +--rw src-ip-prefix?    inet:ip-prefix
|   |   |   |   |   |   +--rw dest-ip-prefix?    inet:ip-prefix
|   |   |   |   |   |   +--rw next-header?        uint8
|   |   |   |   |   |   +--rw traffic-class?        uint8
|   |   |   |   |   |   +--rw flow-label?          inet:ipv6-flow-label
|   |   |   |   |   |   +--rw source-port
|   |   |   |   |   |   |   +--rw (port-range-or-operator)?
|   |   |   |   |   |   |   |   +--:(range)
|   |   |   |   |   |   |   |   |   +--rw lower-port    inet:port-number
|   |   |   |   |   |   |   |   |   +--rw upper-port    inet:port-number
|   |   |   |   |   |   |   |   +--:(operator)
|   |   |   |   |   |   |   |   |   +--rw operator?
|   |   |   |   |   |   |   |   |   |   packet-fields:operator
|   |   |   |   |   |   |   |   +--rw port            inet:port-number
|   |   |   |   +--rw destination-port
|   |   |   |   |   +--rw (port-range-or-operator)?
|   |   |   |   |   |   +--:(range)
|   |   |   |   |   |   |   +--rw lower-port    inet:port-number
|   |   |   |   |   |   |   +--rw upper-port    inet:port-number
|   |   |   |   |   |   +--:(operator)
|   |   |   |   |   |   |   +--rw operator?
|   |   |   |   |   |   |   |   packet-fields:operator
|   |   |   |   |   |   |   +--rw port            inet:port-number
|   |   |   |   +--rw ipsec-spi?                ipsec-spi
|   |   +--:(mpls-detnet-flow)
|   |   |   +--rw (label-space)?
|   |   |   |   +--:(context-label-space)

```



```

|         |         |         | +--rw mpls-label-stack
|         |         |         |   +--rw entry* [id]
|         |         |         |     +--rw id                               uint8
|         |         |         |     +--rw label?
|         |         |         |       |         rt-types:mpls-label
|         |         |         |     +--rw ttl?                               uint8
|         |         |         |     +--rw traffic-class?          uint8
|         |         |         |   +--:(platform-label-space)
|         |         |         |     +--rw label?    rt-types:mpls-label
|         |         |   +--:(aggregated-service)
|         |         |     +--rw service-sub-layer*    service-sub-layer-ref
|         |         |   +--:(aggregated-forwarding)
|         |         |     +--rw forwarding-sub-layer*
|         |         |       forwarding-sub-layer-ref
|   +--rw outgoing
|     +--rw (outgoing-options)
|       +--:(detnet-service-outgoing)
|         +--rw service-outgoing-list*
|           [service-outgoing-index]
|             +--rw service-outgoing-index    uint8
|             +--rw (header-type)?
|               +--:(detnet-mpls-header)
|                 +--rw mpls-label-stack
|                   +--rw entry* [id]
|                     +--rw id                               uint8
|                     +--rw label?
|                       |         rt-types:mpls-label
|                     +--rw ttl?                               uint8
|                     +--rw traffic-class?          uint8
|               +--:(detnet-ip-header)
|                 +--rw src-ip-address?    inet:ip-address
|                 +--rw dest-ip-address?   inet:ip-address
|                 +--rw next-header?       uint8
|                 +--rw traffic-class?     uint8
|                 +--rw flow-label?
|                   |         inet:ipv6-flow-label
|                 +--rw source-port?       inet:port-number
|                 +--rw destination-port?  inet:port-number
|             +--rw next-layer* [index]
|               +--rw index                uint8
|               +--rw forwarding-sub-layer?
|                 forwarding-sub-layer-ref
|       +--:(detnet-service-aggregation)
|         +--rw aggregation-service-sub-layer?
|           |         service-sub-layer-ref
|         +--rw service-label
|           +--rw mpls-label-stack
|             +--rw entry* [id]

```



```

|         |         +--rw id                uint8
|         |         +--rw label?            rt-types:mpls-label
|         |         +--rw ttl?              uint8
|         |         +--rw traffic-class?    uint8
|         +---:(egress-proxy)
|         |   +--rw app-flow*    app-flow-ref
|         +---:(detnet-service-operation)
|         |   +--rw service-sub-layer*    service-sub-layer-ref
|         +---:(detnet-forwarding-operation)
|         |   +--rw forwarding-sub-layer*
|         |       forwarding-sub-layer-ref
+--rw forwarding-sub-layer
  +--rw forwarding-sub-layer-list* [name]
    +--rw name                string
    +--rw traffic-profile?    traffic-profile-ref
    +--rw forwarding-operation-type?    forwarding-operations-type
    +--rw incoming
      +--rw (incoming-options)
      |   +---:(detnet-service-forwarding)
      |   |   +--ro service-sub-layer*    service-sub-layer-ref
      |   +---:(detnet-forwarding-identification)
      |   |   +--rw interface?            if:interface-ref
      |   +--rw (detnet-flow-type)?
      |   |   +---:(ip-detnet-flow)
      |   |   |   +--rw src-ip-prefix?    inet:ip-prefix
      |   |   |   +--rw dest-ip-prefix?   inet:ip-prefix
      |   |   |   +--rw next-header?      uint8
      |   |   |   +--rw traffic-class?    uint8
      |   |   |   +--rw flow-label?       inet:ipv6-flow-label
      |   |   |   +--rw source-port
      |   |   |   |   +--rw (port-range-or-operator)?
      |   |   |   |   |   +---:(range)
      |   |   |   |   |   |   +--rw lower-port    inet:port-number
      |   |   |   |   |   |   +--rw upper-port    inet:port-number
      |   |   |   |   |   +---:(operator)
      |   |   |   |   |   |   +--rw operator?
      |   |   |   |   |   |   |   packet-fields:operator
      |   |   |   |   |   +--rw port            inet:port-number
      |   |   |   +--rw destination-port
      |   |   |   |   +--rw (port-range-or-operator)?
      |   |   |   |   |   +---:(range)
      |   |   |   |   |   |   +--rw lower-port    inet:port-number
      |   |   |   |   |   |   +--rw upper-port    inet:port-number
      |   |   |   |   |   +---:(operator)
      |   |   |   |   |   |   +--rw operator?
      |   |   |   |   |   |   |   packet-fields:operator
      |   |   |   |   |   +--rw port            inet:port-number
      |   |   +--rw ipsec-spi?            ipsec-spi

```



```

|         +---:(mpls-detnet-flow)
|         +---rw (label-space)?
|         +---:(context-label-space)
|         |   +---rw mpls-label-stack
|         |   |   +---rw entry* [id]
|         |   |   |   +---rw id                               uint8
|         |   |   |   +---rw label?
|         |   |   |   |   rt-types:mpls-label
|         |   |   |   +---rw ttl?                               uint8
|         |   |   |   +---rw traffic-class?   uint8
|         |   +---:(platform-label-space)
|         |   |   +---rw label?   rt-types:mpls-label
|   +---:(aggregated-forwarding)
|   |   +---rw forwarding-sub-layer*
|   |   |   forwarding-sub-layer-ref
+---rw outgoing
+---rw (outgoing-options)
+---:(detnet-forwarding-outgoing)
|   +---rw (next-hop-options)
|   |   +---:(simple-next-hop)
|   |   |   +---rw outgoing-interface?   if:interface-ref
|   |   |   +---rw (flow-type)?
|   |   |   |   +---:(ip)
|   |   |   |   |   +---rw (operation-type)?
|   |   |   |   |   +---:(ip-forwarding)
|   |   |   |   |   |   +---rw next-hop-address?
|   |   |   |   |   |   |   inet:ip-address
|   |   |   |   +---:(mpls-over-ip-encapsulation)
|   |   |   |   |   +---rw src-ip-address?
|   |   |   |   |   |   inet:ip-address
|   |   |   |   |   +---rw dest-ip-address?
|   |   |   |   |   |   inet:ip-address
|   |   |   |   |   +---rw next-header?           uint8
|   |   |   |   |   +---rw traffic-class?         uint8
|   |   |   |   |   +---rw flow-label?
|   |   |   |   |   |   inet:ipv6-flow-label
|   |   |   |   |   +---rw source-port?
|   |   |   |   |   |   inet:port-number
|   |   |   |   |   +---rw destination-port?
|   |   |   |   |   |   inet:port-number
|   |   |   +---:(mpls)
|   |   |   |   +---rw mpls-label-stack
|   |   |   |   |   +---rw entry* [id]
|   |   |   |   |   |   +---rw id                               uint8
|   |   |   |   |   |   +---rw label?
|   |   |   |   |   |   |   rt-types:mpls-label
|   |   |   |   |   |   +---rw ttl?                               uint8
|   |   |   |   |   |   +---rw traffic-class?   uint8

```



```

|   +--:(next-hop-list)
|   |   +--rw next-hop-list
|   |   |   +--rw next-hop* [hop-index]
|   |   |   |   +--rw hop-index          uint8
|   |   |   |   +--rw outgoing-interface?
|   |   |   |   |       if:interface-ref
|   |   |   +--rw (flow-type)?
|   |   |   |   +--:(ip)
|   |   |   |   |   +--rw (operation-type)?
|   |   |   |   |   |   +--:(ip-forwarding)
|   |   |   |   |   |   |   +--rw next-hop-address?
|   |   |   |   |   |   |   |       inet:ip-address
|   |   |   |   |   |   +--:(mpls-over-ip-
|   |   |   |   |   |   |   encapsulation)
|   |   |   |   |   |   +--rw src-ip-address?
|   |   |   |   |   |   |   |       inet:ip-address
|   |   |   |   |   |   +--rw dest-ip-address?
|   |   |   |   |   |   |   |       inet:ip-address
|   |   |   |   |   |   +--rw next-header?
|   |   |   |   |   |   |   |       uint8
|   |   |   |   |   |   +--rw traffic-class?
|   |   |   |   |   |   |   |       uint8
|   |   |   |   |   |   +--rw flow-label?
|   |   |   |   |   |   |   |       inet:ipv6-flow-label
|   |   |   |   |   |   +--rw source-port?
|   |   |   |   |   |   |   |       inet:port-number
|   |   |   |   |   |   +--rw destination-port?
|   |   |   |   |   |   |   |       inet:port-number
|   |   |   |   +--:(mpls)
|   |   |   |   |   +--rw mpls-label-stack
|   |   |   |   |   |   +--rw entry* [id]
|   |   |   |   |   |   |   +--rw id          uint8
|   |   |   |   |   |   |   +--rw label?
|   |   |   |   |   |   |   |       rt-types:mpls-label
|   |   |   |   |   |   +--rw ttl?          uint8
|   |   |   |   |   |   +--rw traffic-class? uint8
|   +--:(detnet-service-aggregation)
|   |   +--rw aggregation-service-sub-layer?
|   |   |   service-sub-layer-ref
|   +--rw optional-forwarding-label
|   |   +--rw mpls-label-stack
|   |   |   +--rw entry* [id]
|   |   |   |   +--rw id          uint8
|   |   |   |   +--rw label?      rt-types:mpls-label
|   |   |   |   +--rw ttl?      uint8
|   |   |   |   +--rw traffic-class? uint8
|   +--:(detnet-forwarding-aggregation)
|   |   +--rw aggregation-forwarding-sub-layer?

```



```

| | forwarding-sub-layer-ref
| +--rw forwarding-label
|   +--rw mpls-label-stack
|     +--rw entry* [id]
|       +--rw id          uint8
|       +--rw label?      rt-types:mpls-label
|       +--rw ttl?        uint8
|       +--rw traffic-class? uint8
+--:(detnet-service-operation)
| +--rw service-sub-layer* service-sub-layer-ref
+--:(detnet-forwarding-operation)
  +--rw forwarding-sub-layer*
    forwarding-sub-layer-ref

```

7. DetNet Configuration YANG Model

<CODE BEGINS>

```

module ietf-detnet-config {
  namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-config";
  prefix "ietf-detnet";

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

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

  import ietf-ethertypes {
    prefix "eth";
  }

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

  import ietf-packet-fields {
    prefix "packet-fields";
  }
  import ietf-interfaces {
    prefix "if";
  }

  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
 <<mailto:janos.farkas@ericsson.com>>

Editor: Xuesong Geng
 <<mailto:gengxuesong@huawei.com>>

Editor: Mach Chen
 <<mailto:mach.chen@huawei.com>>

Editor: Yeoncheol Ryoo
 <<mailto:dbduscjf@etri.re.kr>>

Editor: Don Fedyk
 <<mailto:dfedyk@labn.net>>;

Editor: Reshad Rahman
 <<mailto:rrahman@cisco.com>>

Editor: Zhenqiang Li
 <<mailto:lizhenqiang@chinamobile.com>>;

description

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

```
revision 2020-03-04 {  
  description  
    "initial revision";  
  reference  
    "RFC XXXX: draft-ietf-detnet-yang-02";  
}
```

```
identity status {  
  description  
    "Base identity from which all application-status  
    actions are derived";  
}
```

```
identity none {  
  base status;  
  description  
    "Application no ingress/egress";
```



```
    reference
      "draft-ietf-detnet-flow-information-model-06 Section 5.8";
  }

  identity ready {
    base status;
    description
      "Application ingress/egress ready";
    reference
      "draft-ietf-detnet-flow-information-model-06 Section 5.8";
  }

  identity failed {
    base status;
    description
      "Application ingres/egresss failed";
    reference
      "draft-ietf-detnet-flow-information-model-06 Section 5.8";
  }

  identity out-of-service {
    base status;
    description
      "Application Administratively blocked";
    reference
      "draft-ietf-detnet-flow-information-model-06 Section 5.8";
  }

  identity partial-failed {
    base status;
    description
      "Application One or more Egress ready, and one or more Egress
      failed. The DetNet flow can be used if the Ingress is
      Ready.";
    reference
      "draft-ietf-detnet-flow-information-model-06 Section 5.8";
  }

  typedef app-flow-ref {
    type leafref {
      path "/ietf-detnet:detnet"
        + "/ietf-detnet:app-flows"
        + "/ietf-detnet:app-flow"
        + "/ietf-detnet:name";
    }
  }

  typedef service-sub-layer-ref {
```



```
    type leafref {
      path "/ietf-detnet:detnet"
        + "/ietf-detnet:service-sub-layer"
        + "/ietf-detnet:service-sub-layer-list"
        + "/ietf-detnet:name";
    }
  }

typedef forwarding-sub-layer-ref {
  type leafref {
    path "/ietf-detnet:detnet"
      + "/ietf-detnet:forwarding-sub-layer"
      + "/ietf-detnet:forwarding-sub-layer-list"
      + "/ietf-detnet:name";
  }
}

typedef aggregation-grp-ref {
  type leafref {
    path "/ietf-detnet:detnet"
      + "/ietf-detnet:service-aggregation-group"
      + "/ietf-detnet:group-name";
  }
}

typedef traffic-profile-ref {
  type leafref {
    path "/ietf-detnet:detnet"
      + "/ietf-detnet:traffic-profile"
      + "/ietf-detnet:profile-number";
  }
}

typedef ipsec-spi {
  type uint32 {
    range "1..max";
  }
  description
    "SPI";
}

typedef service-operation-type {
  type enumeration {
    enum service-initiation {
      description
        "Operation for DetNet service sub-layer encapsulation";
    }
    enum service-termination {
```



```
        description
            "Operation for DetNet service sub-layer decapsulation";
    }
    enum service-relay {
        description
            "Operation for DetNet service sub-layer swap";
    }
    enum non-detnet {
        description
            "No operation for DetNet service sub-layer";
    }
}
}

typedef forwarding-operations-type {
    type enumeration {
        enum forward {
            description
                "Operation forward to next-hop";
        }
        enum impose-and-forward {
            description
                "Operation impose outgoing label(s) and forward to
                next-hop";
        }
        enum pop-and-forward {
            description
                "Operation pop incoming label and forward to next-hop";
        }
        enum pop-impose-and-forward {
            description
                "Operation pop incoming label, impose one or more
                outgoing label(s) and forward to next-hop";
        }
        enum swap-and-forward {
            description
                "Operation swap incoming label, with outgoing label and
                forward to next-hop";
        }
        enum pop-and-lookup {
            description
                "Operation pop incoming label and perform a lookup";
        }
    }
}
description
    "MPLS operations types";
}
```



```
typedef service-protection-type {
  type enumeration {
    enum none {
      description
        "no service protection provide";
    }
    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";
    }
}

typedef sequence-number-generation-type {
    type enumeration {
        enum copy-from-app-flow {
            description
                "Copy the app-flow sequence number to the DetNet-flow";
        }
        enum generate-by-detnet-flow {
            description
                "Generate the sequence number by DetNet flow";
        }
    }
}

typedef sequence-number-field {
    type enumeration {
        enum zero-sn {
            description
                "there is no DetNet sequence number field.";
        }
        enum short-sn {
            value 16;
            description
                "there is 16bit DetNet sequence number field";
        }
        enum long-sn {
            value 28;
            description
                "there is 28bit DetNet sequence number field";
        }
    }
}

typedef aggregation-group {
    type string;
    description
        "The name of the aggregation group";
}

grouping ip-header {
    description
        "The IPv4/IPv6 packet header information";
    leaf src-ip-address {
        type inet:ip-address;
    }
}
```



```
    description
      "The source IP address of the header";
  }

  leaf dest-ip-address {
    type inet:ip-address;
    description
      "The destination IP address of the header";
  }

  leaf next-header {
    type uint8;
    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 value 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";
  }
}

grouping l2-header {
  description
    "The Ethernet or TSN packet header information";
  leaf source-mac-address {
    type yang:mac-address;
    description
      "The source MAC address value of the ethernet header";
```



```
}

leaf destination-mac-address {
  type yang:mac-address;
  description
    "The destination MAC address value of the ethernet header";
}

leaf ethertype {
  type eth:ethertype;
  description
    "The ethernet packet type value of the ethernet header";
}

leaf vlan-id {
  type uint16;
  description
    "The Vlan value of the ethernet header";
}

leaf pcp {
  type uint8;
  description
    "The priority value of the ethernet header";
}
}

grouping destination-ip-port-identification {
  description
    "The TCP/UDP port(source/destination) identification information";
  container destination-port {
    uses packet-fields:port-range-or-operator;
  }
}

grouping source-ip-port-identification {
  description
    "The TCP/UDP port(source/destination) identification information";
  container source-port {
    uses packet-fields:port-range-or-operator;
  }
}

grouping ip-flow-identification {
  description
    "The IPv4/IPv6 packet header identification information";
  leaf src-ip-prefix {
    type inet:ip-prefix;
```



```
    description
      "The source IP address of the header";
  }

  leaf dest-ip-prefix {
    type inet:ip-prefix;
    description
      "The destination IP address of the header";
  }

  leaf next-header {
    type uint8;
    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 value of the header";
  }

  uses source-ip-port-identification;

  uses destination-ip-port-identification;

  leaf ipsec-spi {
    type ipsec-spi;
    description
      "Security parameter index of SA entry";
  }
}

grouping mpls-flow-identification {
  description
    "The MPLS packet header identification information";
  choice label-space {
    description
      "";
    case context-label-space {
      uses rt-types:mpls-label-stack;
    }
  }
}
```



```
    case platform-label-space {
      leaf label {
        type rt-types:mpls-label;
      }
    }
  }
}

grouping traffic-specification {
  container 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 traffic-requirements {
    container traffic-requirements {
        description
            "FlowRequirements: defines the attributes of the App-flow
             regarding bandwidth, latency, latency variation, loss, and
             misordering tolerance.";
        leaf min-bandwidth {
            type uint64;
            description
                "MinBandwidth is the minimum bandwidth that has to be
                 guaranteed for the DetNet service. MinBandwidth is
                 specified in octets per second.";
        }

        leaf max-latency {
            type uint32;
            description
                "MaxLatency is the maximum latency from Ingress to Egress(es)
                 for a single packet of the DetNet flow. MaxLatency is
                 specified as an integer number of nanoseconds";
        }

        leaf max-latency-variation {
            type uint32;
            description
                "MaxLatencyVariation is the difference between the minimum and
                 the maximum end-to-end one-way latency. MaxLatencyVariation
                 is specified as an integer number of nanoseconds.";
        }

        leaf max-loss {
            type uint8;
            description
                "MaxLoss defines the maximum Packet Loss Ratio (PLR) parameter
                 for the DetNet service between the Ingress and Egress(es) of
                 the DetNet domain.";
        }

        leaf max-consecutive-loss-tolerance {
            type uint32;
```



```
    description
      "Some applications have special loss requirement, such as
      MaxConsecutiveLossTolerance. The maximum consecutive loss
      tolerance parameter describes the maximum number of
      consecutive packets whose loss can be tolerated. The maximum
      consecutive loss tolerance can be measured for example based
      on sequence number";
  }

  leaf max-misordering {
    type uint32;
    description
      "MaxMisordering describes the tolerable maximum number of
      packets that can be received out of order. The maximum
      allowed misordering can be measured for example based on
      sequence number. The value zero for the maximum allowed
      misordering indicates that in order delivery is required,
      misordering cannot be tolerated.";
  }
}

grouping data-flow-spec {
  description
    "app-flow identification";
  choice data-flow-type {
    case tsn-app-flow {
      uses l2-header;
    }

    case ip-app-flow {
      uses ip-flow-identification;
    }

    case mpls-app-flow {
      uses mpls-flow-identification;
    }
  }
}

grouping detnet-flow-spec {
  description
    "detnet-flow identification";
  choice detnet-flow-type {
    case ip-detnet-flow {
      uses ip-flow-identification;
    }
  }
}
```



```
        case mpls-detnet-flow {
            uses mpls-flow-identification;
        }
    }
}

grouping app-flows-ref {
    description
        "incoming or outgoing app-flow reference group";
    leaf-list app-flow {
        type app-flow-ref;
        description
            "List of ingress or egress app-flows";
    }
}

grouping service-sub-layer-ref {
    description
        "incoming or outgoing service sub-layer reference group";
    leaf-list service-sub-layer {
        type service-sub-layer-ref;
        description
            "List of incoming or outgoing service sub-layer
            that has to aggregate or disaggregate";
    }
}

grouping forwarding-sub-layer-ref {
    description
        "incoming or outgoing forwarding sub-layer reference group";
    leaf-list forwarding-sub-layer {
        type forwarding-sub-layer-ref;
        description
            "List of incoming or outgoing forwarding sub-layer
            that has to aggregate or disaggregate";
    }
}

grouping detnet-header {
    description
        "DetNet header info for DetNet encapsulation or swap";
    choice header-type {
        case detnet-mpls-header {
            description
                "MPLS label stack for DetNet MPLS encapsulation
                for forwarding";
            uses rt-types:mpls-label-stack;
        }
    }
}
```



```
    case detnet-ip-header {
      description
        "IPv4/IPv6 packet header for DetNet IP encapsulation";
      uses ip-header;
    }
  }
}

grouping aggregation-header {
  description
    "DetNet aggregation header  DetNet encapsulation";
  container aggregation-header {
    description
      "MPLS label stack for DetNet MPLS encapsulation or
      forwarding";
    uses rt-types:mpls-label-stack;
  }
}

grouping detnet-app-next-hop-content {
  description
    "Generic parameters of DetNet next hops.";
  choice next-hop-options {
    mandatory true;
    description
      "Options for next hops.
      It is expected that further cases will be added through
      augments from other modules, e.g., for recursive
      next hops.";
    case simple-next-hop {
      description
        "This case represents a simple next hop consisting of the
        next-hop address and/or outgoing interface.
        Modules for address families MUST augment this case with a
        leaf containing a next-hop address of that address
        family.";
      leaf outgoing-interface {
        type if:interface-ref;
      }

      choice flow-type {
        case ip {
          leaf next-hop-address {
            type inet:ip-address;
          }
        }
      }

      case mpls {
```



```
        uses rt-types:mpls-label-stack;
    }
}

case next-hop-list {
    container next-hop-list {
        description
            "Container for multiple next hops.";
        list next-hop {
            key "hop-index";
            description
                "An entry in a next-hop list.
                 Modules for address families MUST augment this list
                 with a leaf containing a next-hop address of that
                 address family.";
            leaf hop-index {
                type uint8;
                description
                    "";
            }

            leaf outgoing-interface {
                type if:interface-ref;
            }

            choice flow-type {
                case ip {
                    leaf next-hop-address {
                        type inet:ip-address;
                    }
                }

                case mpls {
                    uses rt-types:mpls-label-stack;
                }
            }
        }
    }
}

grouping detnet-forwarding-next-hop-content {
    description
        "Generic parameters of DetNet next hops.";
    choice next-hop-options {
        mandatory true;
    }
}
```



```
description
  "Options for next hops.
  It is expected that further cases will be added through
  augments from other modules, e.g., for recursive
  next hops.";
case simple-next-hop {
  description
    "This case represents a simple next hop consisting of the
    next-hop address and/or outgoing interface.
    Modules for address families MUST augment this case with a
    leaf containing a next-hop address of that address
    family.";
  leaf outgoing-interface {
    type if:interface-ref;
  }

  choice flow-type {
    case ip {
      choice operation-type {
        case ip-forwarding {
          leaf next-hop-address {
            type inet:ip-address;
          }
        }
        case mpls-over-ip-encapsulation {
          uses ip-header;
        }
      }
    }

    case mpls {
      uses rt-types:mpls-label-stack;
    }
  }
}

case next-hop-list {
  container next-hop-list {
    description
      "Container for multiple next hops.";
    list next-hop {
      key "hop-index";
      description
        "An entry in a next-hop list.

        Modules for address families MUST augment this list
        with a leaf containing a next-hop address of that
```



```
        address family.";
    leaf hop-index {
        type uint8;
        description
            "";
    }

    leaf outgoing-interface {
        type if:interface-ref;
    }

    choice flow-type {
        case ip {
            choice operation-type {
                case ip-forwarding {
                    leaf next-hop-address {
                        type inet:ip-address;
                    }
                }

                case mpls-over-ip-encapsulation {
                    uses ip-header;
                }
            }
        }

        case mpls {
            uses rt-types:mpls-label-stack;
        }
    }
}

}
}
}

container detnet {
    list traffic-profile {
        key "profile-number";
        description
            "A traffic profile";
        leaf profile-number {
            type uint16;
            description
                "An Aggregation group ID. Zero means the service is not
                part of a group";
        }
    }
}
```



```
    uses traffic-requirements;

    uses traffic-specification;

    leaf-list member-applications {
        type app-flow-ref;
        config false;
        description
            "Applicaions attached to this profile";
    }

    leaf-list member-services {
        type service-sub-layer-ref;
        config false;
        description
            "Services attached to this profile";
    }

    leaf-list member-groups {
        type aggregation-grp-ref;
        config false;
        description
            "Groups attached to this profile";
    }

    leaf-list member-forwarding-sublayers {
        type forwarding-sub-layer-ref;
        config false;
        description
            "Forwarding sub-layer attached to this profile";
    }
}

container app-flows {
    description
        "The DetNet app-flow configuration";
    list app-flow {
        key "name";
        description
            "";
        leaf name {
            type "string";
            description
                "The name to identify the DetNet app-flow";
        }

        leaf app-flow-bidir-congruent {
            type boolean;
```



```
    description
      "Defines the data path requirement of the App-flow whether
       it must share the same data path and physical path
       for both directions through the network,
       e.g., to provide congruent paths in the two directions.";
  }

  leaf outgoing-service {
    type service-sub-layer-ref;
    config false;
    description
      "Binding to this applications outgoing
       service";
  }

  leaf incoming-service {
    type service-sub-layer-ref;
    config false;
    description
      "Binding to this applications incoming
       service";
  }

  leaf traffic-profile {
    type traffic-profile-ref;
    description
      "The Traffic Profile for this group";
  }

  container ingress {
    // key "name"; This should be a list for aggregation
    description
      "Ingress DetNet application flows or a compound flow";
    leaf name {
      type string;
      description
        "Ingress DetNet application";
    }

    leaf app-flow-status {
      type identityref {
        base status;
      }
      config false;
      description
        "Status of ingress application flow";
    }
  }
```



```
    leaf interface {
      type if:interface-ref;
    }

    uses data-flow-spec;
  } //End of app-ingress

  container egress {
    description
      "Route's next-hop attribute.";
    // key "name"; This should be a list for aggregation
    leaf name {
      type string;
      description
        "Egress DetNet application";
    }

    choice application-type {
      container ethernet {
        leaf ethernet-place-holder {
          type string;
          description
            "Place holder for matching ethernet";
        }
      }

      container ip-mpls {
        uses detnet-app-next-hop-content;
      }
    }
  }
}

list service-aggregation-group {
  key "group-name";
  description
    "A group of services";
  leaf group-name {
    type aggregation-group;
    description
      "An Aggregation group name. Empty means the service is not
      part of a group";
  }
}

container outgoing {
  leaf traffic-profile {
    type traffic-profile-ref;
```



```
    description
      "The Traffic Profile for this group";
  }

  container service-protection {
    leaf service-protection-type {
      type service-protection-type;
      description
        "The DetNet service protection type such as PRF, PEF,
        PEOF, PERF, and PEORF";
    }

    leaf sequence-number-length {
      type sequence-number-field;
      description
        "Sequence number field can choice 0 bit, 16bit, 28 bit
        field";
    }
  }

  uses aggregation-header;

  leaf-list services {
    type service-sub-layer-ref;
    config false;
    description
      "List of registered services";
  }
}

container incoming {
  uses aggregation-header;

  leaf-list services {
    type service-sub-layer-ref;
    config false;
    description
      "List of registered services";
  }
}

container service-sub-layer {
  description
    "The DetNet service sub-layer configuration";
  list service-sub-layer-list {
    key "name";
    description
```



```
    "";  
    leaf name {  
        type string;  
        description  
            "The name of the DetNet service sub-layer";  
    }  
  
    leaf service-rank {  
        type uint8;  
        description  
            "The DetNet rank for this service";  
    }  
  
    choice service-type {  
        mandatory true;  
        container non-grouped {  
            leaf traffic-profile {  
                type traffic-profile-ref;  
                description  
                    "The Traffic Profile for this service";  
            }  
  
            leaf service-operation-type {  
                type service-operation-type;  
            }  
        }  
  
        container grouped {  
            leaf group-ref {  
                type aggregation-grp-ref;  
                description  
                    "The aggregation group this service belongs to";  
            }  
        }  
    }  
  
    container service-protection {  
        leaf service-protection-type {  
            type service-protection-type;  
            description  
                "The DetNet service protection type such as PRF, PEF,  
                PEOF, PERF, and PEORF";  
        }  
  
        leaf sequence-number-length {  
            type sequence-number-field;  
            description  
                "Sequence number field can choice 0 bit, 16bit, 28 bit
```



```
        filed";
    }
}

leaf service-operation-type {
    type service-operation-type;
}

container incoming {
    description
        "The DetNet service sub-layer incoming configuration.";
    choice incoming-options {
        mandatory true;
        description
            "";
        case ingress-application {
            uses app-flows-ref;
        }

        case detnet-service-identification {
            uses detnet-flow-spec;
        }

        case aggregated-service {
            uses service-sub-layer-ref;
        }

        case aggregated-forwarding {
            uses forwarding-sub-layer-ref;
        }
    }
}

container outgoing {
    description
        "The DetNet service sub-layer outgoing configuration.";
    choice outgoing-options {
        mandatory true;
        description
            "";
        case detnet-service-outgoing {
            //uses detnet-service-next-hop-content;
            list service-outgoing-list {
                key "service-outgoing-index";
                leaf service-outgoing-index {
                    type uint8;
                }
            }
        }
    }
}
```



```
    uses detnet-header;

    list next-layer {
      key "index";
      description
        "lower-layer info";
      leaf index {
        type uint8;
      }

      leaf forwarding-sub-layer {
        type forwarding-sub-layer-ref;
      }
    }
  }
}

case detnet-service-aggregation {
  leaf aggregation-service-sub-layer {
    type service-sub-layer-ref;
  }

  container service-label {
    uses rt-types:mpls-label-stack;
  }
}

case egress-proxy {
  uses app-flows-ref;
}

case detnet-service-operation {
  uses service-sub-layer-ref;
}

case detnet-forwarding-operation {
  uses forwarding-sub-layer-ref;
}
}
}
}

container forwarding-sub-layer {
  description
    "The DetNet forwarding sub-layer configuration";
  list forwarding-sub-layer-list {
    key "name";
```



```
description
  "";
leaf name {
  type string;
  description
    "The name of the DetNet forwarding sub-layer";
}

leaf traffic-profile {
  type traffic-profile-ref;
  description
    "The Traffic Profile for this group";
}

leaf forwarding-operation-type {
  type forwarding-operations-type;
}

container incoming {
  description
    "The DetNet forwarding sub-layer incoming configuration.";
  choice incoming-options {
    mandatory true;
    description
      "";
    case detnet-service-forwarding {
      leaf-list service-sub-layer {
        type service-sub-layer-ref;
        config false;
        description
          "";
      }
    }

    case detnet-forwarding-identification {
      leaf interface {
        type if:interface-ref;
        description
          "";
      }

      uses detnet-flow-spec;
    }

    case aggregated-forwarding {
      uses forwarding-sub-layer-ref;
    }
  }
}
```


8. Open Issues

There are some open issues that are still under discussion:

- o Aggregation.
- o Going along the the updated data plane model.
- o Terminologies.

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

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

<TBD>

11. Acknowledgements

12. References

12.1. Normative References

- [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>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", [RFC 8655](#), DOI 10.17487/RFC8655, October 2019, <<https://www.rfc-editor.org/info/rfc8655>>.

12.2. Informative References

[I-D.ietf-detnet-flow-information-model]
Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D.
Fedyk, "DetNet Flow Information Model", [draft-ietf-detnet-flow-information-model-10](#) (work in progress), May 2020.

Authors' Addresses

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Yeoncheol Ryoo
ETRI

Email: dbduscjf@etri.re.kr

Don Fedyk
LabN Consulting, L.L.C.

Email: dfedyk@labn.net

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

