Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

Z. Du
P. Liu
L. Geng
China Mobile

November 2, 2020

Micro-burst Decreasing in Layer3 Network for Low-Latency Traffic
draft-du-detnet-layer3-low-latency-01

Abstract

This document introduces a method to decrease the micro-bursts in Layer3 network for low-latency traffic.

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

Copyright Notice

Table of Contents

1. Problem Statement

Currently, the DetNet architecture in RFC 8655 [RFC8655] is supposed to work in campus-wide networks and private WANs, and hasn't covered the large-scale ISP network scenario. However, the low-latency requirement exists in both L2 and L3 networks, and in both small and large networks.

As talked in [I-D.qiang-detnet-large-scale-detnet], deploying deterministic services in a large-scale network brings a lot of new challenges. A novel method called LDN is introduced in [I-D.qiang-detnet-large-scale-detnet], which explores the deterministic forwarding over a large-scale network.

According to RFC 8655 [RFC8655], DetNet operates at the IP layer and delivers service over lower-layer technologies such as MPLS and IEEE 802.1 Time-Sensitive Networking (TSN). However, the TSN mechanisms are designed for L2 network originally, and cannot be directly used in the large-scale layer 3 network because of various reasons. For example, some TSN mechanisms need synchronization of the network equipments, which is easier in a small network, but hard in a large network; some mechanisms need a per-flow state in the forwarding plane, which is un-scalable; and some TSN mechanisms need a constant and forecastable traffic characteristics, which is more complicated in a large network which includes much more flows joining in or leaving randomly and the traffic characteristics are more dynamic.

The current forwarding mechanism in an IP router is based on statistical multiplexing, and cannot provide the deterministic service because of various reasons. Even be given a high priority, a deterministic packet can experience a long congestion delay or be lost in a relatively light-loaded network, which is called micro-burst in the network.

Figure 1 show the problem of the current scheduling mechanism of an IP network. Before the scheduling in an IP network, the critical packets are well paced, but after the scheduling, the packets will be gathered even the total traffic rate is unchanged. When an IP outgoing interface receives multiple critical flows from several incoming interfaces, the situation becomes more serious. However, an IP router will try to send them as soon as possible, so occasionally, in some later hops, micro-bursts will emerge.

```
       _     _     _     _     _     _     _     _     _     _     _
      | |   | |   | |   | |   | |   | |   | |   | |   | |   | |   | |
   -------------------------------------------------------------------
                      Before scheduling in an IP network

       _  _  _  _  _  _                _  _  _  _  _
      | || || || || || |              | || || || || |
   -------------------------------------------------------------------
                      After scheduling in an IP network
```

Figure 1: Change of the traffic characteristics in an IP network

This document proposes a method to support the low latency traffic bearing in an IP network by avoiding micro-bursts in the network as much as possible.

2. Mechanism to Decrease Micro-bursts

The mechanism needs the cooperation of the edge node and the forwarding node in an IP network.

2.1. Process of Edge Node

The edge node of the IP network can recognize each critical flows just as in the TSN network, and then give them individually a good shaping. In fact, in TSN mechanisms, no micro-busrt will emerge for critical traffic, and each TSN mechanism is proved to be effective under certain conditions.

This document suggests the edge node to shape the critical traffic by using the CBS method in IEEE 802.1Qav, or the shaping methods in IEEE 802.1Qcr. Generally, the shaping methods can generate a paced traffic for each critical flow.

The parameters of the shaper, such as the sending rate, can be configured for each flow by some means.

2.2. Process of Forwarding Node

For the forwarding node, it is uneasy to recognize each critical flow because of the high pressure of forwarding a large amount of packets. It is suggested that no per-flow state is maintained in the forwarding node. It is to say that, in the forwarding node, the critical flows should be aggregated and handled together.

This document suggests that the forwarding node can deploy a specific queue at each outgoing interface. The queue will buffer all critical traffic that need to go out through that interface, and will pace them by using methods mentioned in Section 2.1.

The shaping method in TSN is used here instead of the original forwarding method in an IP router, which can make the critical traffic be forwarded orderly instead of as soon as possible. Therefore, micro-bursts can be decreased in the network.

If all the forwarding nodes can do their jobs properly, i.e., they can well pace the critical traffic, no or rare micro-bursts for the critical traffic will take place. In this way, the critical traffic will have a relatively low latency in the IP network with less uncertainty.

As no per-flow state is maintained in the forwarding node, the sending rate of the shaper is hard to decide. In this document, the sending rate is suggested to be generated referring to the incoming rate of the queue. The purpose is to maintain a proper buffer depth for the queue.

3. IANA Considerations

TBD.

4. Security Considerations

TBD.

5. Acknowledgements

TBD.

6. References

6.1. 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>.

6.2. Informative References

[I-D.qiang-detnet-large-scale-detnet]
Qiang, L., Geng, X., Liu, B., Eckert, T., Geng, L., and G. Li, "Large-Scale Deterministic IP Network", draft-qiang-detnet-large-scale-detnet-05 (work in progress), September 2019.

[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>.

Authors' Addresses

Zongpeng Du
China Mobile
No.32 XuanWuMen West Street
Beijing 100053
China

Email: duzongpeng@foxmail.com

Peng Liu
China Mobile
No.32 XuanWuMen West Street
Beijing 100053
China

Email: liupengyjy@chinamobile.com

Liang Geng
China Mobile
No.32 XuanWuMen West Street
Beijing 100053
China

Email: gengliang@chinamobile.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

X. Geng
M. Chen
F. Yang
Huawei Technologies

November 2, 2020

Segment Routing for Redundancy Protection
draft-geng-spring-sr-redundancy-protection-00

Abstract

Redundancy protection is one of the mechanisms to achieve service protection, following the principle of PREOF (Packet Replication/ Elimination/Ordering Function). To empower the Segment Routing with the capability of redundancy protection, two types of Segment including Redundancy Segment and Merging Segment are introduced. The instantiation of Redundancy and Merging Segments can be applied to both segment routing over MPLS (SR-MPLS) and segment routing over IPv6 (SRv6).

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 .

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Service protection defined in [RFC8655] is initially required from the use cases in a variety of industries described in [RFC8578]. Together with other two techniques Resource allocation and Explicit routes, targets to provide the deterministic flow transmission. Meanwhile, with the emerge of Cloud VR, Cloud Game, HDV (high-definition video) applications running in the Internet, SLA (Service Level Agreement) guarantee becomes an important issue which requires new technologies and solutions for network.

Redundancy Protection is one of the mechanisms to achieve service protection, following the principle of PREOF (Packet Replication/Elimination/Ordering Function) defined in [RFC8655]. Specifically, replicates the packets of flows into two or more copies, transports different copies through different path in parallel, eliminates and orders the packets at end to provide redundancy protection.

Segment Routing (SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". A segment can be associated to an arbitrary processing of the packet in the node identified by the segment.

This document extends the capabilities in SR paradigm to support the redundancy protection, including the definitions of new Segments and a variation of Segment Routing Policy. The new mechanism applies equally to both segment routing with MPLS data plane (SR-MPLS) and segment routing with IPv6 data plane (SRv6).

2. Terminology and Conventions

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 [I-D.ietf-spring-srv6-network-programming] and[RFC2119].

Redundancy Node: the start point of redundancy protection, which is a network device that could implement packet replication

Merging Node: the end point of redundancy protection, which is a network node that could implement packet elimination and ordering (optionally)

Redundancy Policy: an extended SR policy which includes more than one active segment lists to support redundancy protection

Flow Identification: information in SR data service to indicate one flow

Sequence Number: information in SR data service to indicate the packet sequence of one flow

Editor's Note: Similar mechanism is defined as "Service Protection" in the [RFC8655]. In this document, we define a new term "Redundancy Protection" to distinguish with other service protection method. Some of the terms are similar as [RFC8655].

3. Redundancy Protection in Segment Routing Scenario

```
                |                                        |
                |<-------------- SR Domain ------------->|
                |                                        |
                |            +-----+R3+-----+            |
              +---+        +-+-+          +-+-+        +---+
       -------|R1 |--------|Red|          |Mer|--------|R2 |-------
              +---+        +-+-+          +-+-+        +---+
                             +-----+R4+-----+
```

Figure 1: Example Scenario of Redundancy Protection in SR Domain

This figure shows the process of redundancy protection when a flow is sent into SR domain:

1) R1 receives the packet flow and encapsulates with segment to destination R2, either instantiated in a stack of MPLS labels or Segment Routing Extension Header (SRH) defined in [RFC8754];

2) When the packet flow arrives in Red node, known as Redundancy Node, one flow is replicated to two copies with the same flow identifier; For each packet in one flow, sequence number is marked to indicate the packet sequence; the flow identifier and sequence number of each packet can alternatively be marked at the ingress edge R1 of SR domain;

3) Two replicated flows go through different paths till Mer node, known as Merging Node; When there is any failures happened in one path, the service continues to deliver through the other path without break;

4) The first received packet of the flow is transmitted from Merging Node to R2, and the redundant packets are eliminated;

5) Sometimes, the packet will arrive out of order because of redundancy protection, the function of reordering may be necessary in the Merging Node;

In this example, service protection is supported by utilizing at least two packet flows transmitted over two candidate paths. For a unidirectional flow, Red node supports replication function, and Mer node supports elimination and ordering functions.

4. Segment to Support Redundancy Protection

To achieve the Packet Replication/ Elimination/Ordering Function, Redundancy Segment and Merging Segment are introduced.

4.1. Redundancy Segment

Redundancy Segment is associated with a Redundancy Policy on redundancy node. Redundancy Segment is associated with service instructions, indicating the following operations:

- Steering the packet into the corresponding redundancy policy

- Packet replication based on the redundancy policy, e.g., the number of replication copies

- Encapsulate the packet with necessary meta data (e.g., flow identification, sequence number) if it is not included in the original packet

In the case of SRv6, a new behavior End.R for Redundancy Segment is defined.

When N receives a packet whose IPv6 DA is S and S is a Redundancy SID, N does:

```
S01.  When an SRH is processed {
S02.     If (Segments Left>0)   {
S03.          Create two headers IPv6+SRH-1 and IPv6+SRH-2
S04.          Insert different policy-instructed segment lists into SRH-1 and SRH-2
S05.          Add Flow Identification and Sequence Number to SRH-1 and SRH-2
S06.          Remove the incoming outer IPv6+SRH header
S07.          Create a duplication of the incoming packet payload
S08.          Encapsulate the original packet with IPv6+SRH-1 header
S09.          Encapsulate the duplicate packet with IPv6+SRH-2 header
S10.          Set IPv6 SA as the local address of this node
S11.          Set IPv6 DA of IPv6+SRH-1 to the first segment of SRv6 Policy in SRH-1
S12.          Set IPv6 DA of IPv6+SRH-2 to the first segment of SRv6 Policy in SRH-2
S13.     }
S14.     ELSE {
S15.          Drop the packet
S16.     }
```

4.2. Merging Segment

Merging Segment is associated with service instructions, indicates the following operations:

- Packet merging and elimination: forward the first received packets and eliminate the redundant packets

- Packet ordering(optional): reorder the packets if the packet arrives out of order

In the case of SRv6, a new behavior End.M for Merging Segment is defined.

When N receives a packet whose IPv6 DA is S and S is a Merging SID, N does:

```
When an SRH is processed {
    If (Segments Left>0) & "the packet is not a redundant packet" {
        Do not decrement SL nor update the IPv6 DA with SRH[SL]
        Create a new outer IPv6+SRH-3 header
        Insert the policy-instructed segment lists in the newly created SRH-3
        Remove the incoming outer IPv6+SRH header
        Set IPv6 DA of IPv6+SRH-3 to the first segment of SRv6 Policy in SRH-3
    }
    ELSE {
        Drop the packet
    }
```

5. Meta Information to Support Redundancy Protection

To distinguish the different copies replicated at Redundancy node, and distinguish the different packets in the same flow to perform elimination and ordering, the definition of Flow Identification and Sequence Number is required.

Flow Identification and Sequence Number can be defined as MPLS labels in SR over MPLS data plane, or as option TLV in SRH header in SR over IPv6 data plane. This information must be carried along the path to Merging node. Merging node removes Flow Identifier and Sequence Number once the elimination and ordering is accomplished.

6. Segment Routing Policy to Support Redundancy Protection

Redundancy Policy is a variation of SR Policy, which is identified through the tuple <redundancy node, redundancy ID, merging node>. Redundancy Policy extends SR policy to include more than one ordered lists of segments between redundancy node and merging node to steer the same flow through different paths in SR domain. In Redundancy Policy, Redundancy Segment MUST be specified, and the last segment of each ordered list of segments SHOULD be Merging Segment.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

TBD

9. Acknowledgements

TBD

10. References

10.1. Normative References

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

[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>.

10.2. Informative References

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

[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>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

Authors' Addresses

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Fan Yang
Huawei Technologies

Email: shirley.yangfan@huawei.com

DetNet N. Finn
Internet-Draft Huawei Technologies Co. Ltd
Intended status: Informational J-Y. Le Boudec
Expires: May 6, 2021 E. Mohammadpour
EPFL
J. Zhang
Huawei Technologies Co. Ltd
B. Varga
J. Farkas
Ericsson
November 2, 2020

DetNet Bounded Latency
draft-ietf-detnet-bounded-latency-02

Abstract

This document presents a timing model for Deterministic Networking (DetNet), so that existing and future standards can achieve the DetNet quality of service features of bounded latency and zero congestion loss. It defines requirements for resource reservation protocols or servers. It calls out queuing mechanisms, defined in other documents, that can provide the DetNet quality of service.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking (TSN, [IEEE8021TSN]) to provide the DetNet services of bounded latency and zero congestion loss depends upon A) configuring and allocating network resources for the exclusive use of DetNet/TSN flows; B) identifying, in the data plane, the resources to be utilized by any given packet, and C) the detailed behavior of those resources, especially transmission queue selection, so that latency bounds can be reliably assured. Thus, DetNet is an example of an IntServ Guaranteed Quality of Service [RFC2212]

As explained in [RFC8655], DetNet flows are characterized by 1) a maximum bandwidth, guaranteed either by the transmitter or by strict input metering; and 2) a requirement for a guaranteed worst-case end-to-end latency. That latency guarantee, in turn, provides the opportunity for the network to supply enough buffer space to guarantee zero congestion loss.

To be of use to the applications identified in [RFC8578], it must be possible to calculate, before the transmission of a DetNet flow commences, both the worst-case end-to-end network latency, and the amount of buffer space required at each hop to ensure against congestion loss.

This document references specific queuing mechanisms, defined in other documents, that can be used to control packet transmission at each output port and achieve the DetNet qualities of service. This document presents a timing model for sources, destinations, and the DetNet transit nodes that relay packets that is applicable to all of those referenced queuing mechanisms.

Using the model presented in this document, it should be possible for an implementor, user, or standards development organization to select a particular set of queuing mechanisms for each device in a DetNet network, and to select a resource reservation algorithm for that network, so that those elements can work together to provide the DetNet service.

This document does not specify any resource reservation protocol or server. It does not describe all of the requirements for that protocol or server. It does describe requirements for such resource reservation methods, and for queuing mechanisms that, if met, will enable them to work together.

2. Terminology and Definitions

This document uses the terms defined in [RFC8655].

3. DetNet bounded latency model

3.1. Flow creation

This document assumes that following paradigm is used for provisioning DetNet flows:

1. Perform any configuration required by the DetNet transit nodes in the network for the classes of service to be offered, including one or more classes of DetNet service. This configuration is done beforehand, and not tied to any particular flow.

2. Characterize the new DetNet flow, particularly in terms of required bandwidth.

3. Establish the path that the DetNet flow will take through the network from the source to the destination(s). This can be a point-to-point or a point-to-multipoint path.

4. Select one of the DetNet classes of service for the DetNet flow.

5. Compute the worst-case end-to-end latency for the DetNet flow, using one of the methods, below (Section 3.1.1, Section 3.1.2). In the process, determine whether sufficient resources are available for that flow to guarantee the required latency and to provide zero congestion loss.

6. Assuming that the resources are available, commit those resources to the flow. This may or may not require adjusting the parameters that control the filtering and/or queuing mechanisms at each hop along the flow’s path.

This paradigm can be implemented using peer-to-peer protocols or using a central server. In some situations, a lack of resources can require backtracking and recursing through this list.

Issues such as un-provisioning a DetNet flow in favor of another, when resources are scarce, are not considered, here. Also not addressed is the question of how to choose the path to be taken by a DetNet flow.

3.1.1. Static flow latency calculation

The static problem:

> Given a network and a set of DetNet flows, compute an end-to-end latency bound (if computable) for each flow, and compute the resources, particularly buffer space, required in each DetNet transit node to achieve zero congestion loss.

In this calculation, all of the DetNet flows are known before the calculation commences. This problem is of interest to relatively static networks, or static parts of larger networks. It provides bounds on delay and buffer size. The calculations can be extended to provide global optimizations, such as altering the path of one DetNet flow in order to make resources available to another DetNet flow with tighter constraints.

The static flow calculation is not limited only to static networks; the entire calculation for all flows can be repeated each time a new DetNet flow is created or deleted. If some already-established flow would be pushed beyond its latency requirements by the new flow, then the new flow can be refused, or some other suitable action taken.

This calculation may be more difficult to perform than that of the dynamic calculation (Section 3.1.2), because the flows passing through one port on a DetNet transit node affect each others' latency. The effects can even be circular, from Flow A to B to C and back to A. On the other hand, the static calculation can often accommodate queuing methods, such as transmission selection by strict priority, that are unsuitable for the dynamic calculation.

3.1.2. Dynamic flow latency calculation

The dynamic problem:

> Given a network whose maximum capacity for DetNet flows is bounded by a set of static configuration parameters applied to the DetNet transit nodes, and given just one DetNet flow, compute the worst-case end-to-end latency that can be experienced by that flow, no matter what other DetNet flows (within the network's configured parameters) might be created or deleted in the future. Also, compute the resources, particularly buffer space, required in each DetNet transit node to achieve zero congestion loss.

This calculation is dynamic, in the sense that flows can be added or deleted at any time, with a minimum of computation effort, and without affecting the guarantees already given to other flows.

The choice of queuing methods is critical to the applicability of the dynamic calculation. Some queuing methods (e.g. CQF, Section 6.6) make it easy to configure bounds on the network's capacity, and to make independent calculations for each flow. Some other queuing methods (e.g. strict priority with the credit-based shaper defined in [IEEE8021Q] section 8.6.8.2) can be used for dynamic flow creation, but yield poorer latency and buffer space guarantees than when that same queuing method is used for static flow creation (Section 3.1.1).

3.2. Relay node model

A model for the operation of a DetNet transit node is required, in order to define the latency and buffer calculations. In Figure 1 we see a breakdown of the per-hop latency experienced by a packet passing through a DetNet transit node, in terms that are suitable for computing both hop-by-hop latency and per-hop buffer requirements.

```
         DetNet transit node A            DetNet transit node B
      +-------------------------+       +------------------------+
      |              Queuing    |       |              Queuing   |
      |   Regulator subsystem   |       |   Regulator subsystem  |
      |   +-+-+-+-+ +-+-+-+-+   |       |   +-+-+-+-+ +-+-+-+-+  |
   -->+   | | | | | | | | | +   +------>+   | | | | | | | | | +  +--->
      |   +-+-+-+-+ +-+-+-+-+   |       |   +-+-+-+-+ +-+-+-+-+  |
      |                         |       |                        |
      +-------------------------+       +------------------------+
      |<->|<------>|<------->|<->|<---->|<->|<------>|<------>|<->|<--
   2,3  4      5        6      1    2,3   4      5        6     1   2,3
                   1: Output delay      4: Processing delay
                   2: Link delay        5: Regulation delay
                   3: Preemption delay  6: Queuing delay.
```

Figure 1: Timing model for DetNet or TSN

In Figure 1, we see two DetNet transit nodes (typically, bridges or routers), with a wired link between them. In this model, the only queues, that we deal with explicitly, are attached to the output port; other queues are modeled as variations in the other delay times. (E.g., an input queue could be modeled as either a variation in the link delay [2] or the processing delay [4].) There are six delays that a packet can experience from hop to hop.

1. Output delay
   The time taken from the selection of a packet for output from a queue to the transmission of the first bit of the packet on the physical link. If the queue is directly attached to the physical port, output delay can be a constant. But, in many implementations, the queuing mechanism in a forwarding ASIC is separated from a multi-port MAC/PHY, in a second ASIC, by a multiplexed connection. This causes variations in the output delay that are hard for the forwarding node to predict or control.

2. Link delay
   The time taken from the transmission of the first bit of the packet to the reception of the last bit, assuming that the transmission is not suspended by a preemption event. This delay has two components, the first-bit-out to first-bit-in delay and

the first-bit-in to last-bit-in delay that varies with packet size. The former is typically measured by the Precision Time Protocol and is constant (see [RFC8655]). However, a virtual "link" could exhibit a variable link delay.

3. Preemption delay
   If the packet is interrupted in order to transmit another packet or packets, (e.g. [IEEE8023] clause 99 frame preemption) an arbitrary delay can result.

4. Processing delay
   This delay covers the time from the reception of the last bit of the packet to the time the packet is enqueued in the regulator (Queuing subsystem, if there is no regulation). This delay can be variable, and depends on the details of the operation of the forwarding node.

5. Regulator delay
   This is the time spent from the insertion of the last bit of a packet into a regulation queue until the time the packet is declared eligible according to its regulation constraints. We assume that this time can be calculated based on the details of regulation policy. If there is no regulation, this time is zero.

6. Queuing subsystem delay
   This is the time spent for a packet from being declared eligible until being selected for output on the next link. We assume that this time is calculable based on the details of the queuing mechanism. If there is no regulation, this time is from the insertion of the packet into a queue until it is selected for output on the next link.

Not shown in Figure 1 are the other output queues that we presume are also attached to that same output port as the queue shown, and against which this shown queue competes for transmission opportunities.

The initial and final measurement point in this analysis (that is, the definition of a "hop") is the point at which a packet is selected for output. In general, any queue selection method that is suitable for use in a DetNet network includes a detailed specification as to exactly when packets are selected for transmission. Any variations in any of the delay times 1-4 result in a need for additional buffers in the queue. If all delays 1-4 are constant, then any variation in the time at which packets are inserted into a queue depends entirely on the timing of packet selection in the previous node. If the delays 1-4 are not constant, then additional buffers are required in the queue to absorb these variations. Thus:

- Variations in output delay (1) require buffers to absorb that variation in the next hop, so the output delay variations of the previous hop (on each input port) must be known in order to calculate the buffer space required on this hop.

- Variations in processing delay (4) require additional output buffers in the queues of that same DetNet transit node. Depending on the details of the queueing subsystem delay (6) calculations, these variations need not be visible outside the DetNet transit node.

4. Computing End-to-end Delay Bounds

4.1. Non-queuing delay bound

End-to-end delay bounds can be computed using the delay model in Section 3.2. Here, it is important to be aware that for several queuing mechanisms, the end-to-end delay bound is less than the sum of the per-hop delay bounds. An end-to-end delay bound for one DetNet flow can be computed as

```
end_to_end_delay_bound = non_queuing_delay_bound +
queuing_delay_bound
```

The two terms in the above formula are computed as follows.

First, at the h-th hop along the path of this DetNet flow, obtain an upperbound per-hop_non_queuing_delay_bound[h] on the sum of the bounds over the delays 1,2,3,4 of Figure 1. These upper bounds are expected to depend on the specific technology of the DetNet transit node at the h-th hop but not on the T-SPEC of this DetNet flow. Then set non_queuing_delay_bound = the sum of per-hop_non_queuing_delay_bound[h] over all hops h.

Second, compute queuing_delay_bound as an upper bound to the sum of the queuing delays along the path. The value of queuing_delay_bound depends on the T-SPEC of this flow and possibly of other flows in the network, as well as the specifics of the queuing mechanisms deployed along the path of this flow. The computation of queuing_delay_bound is described in Section 4.2 as a separate section.

4.2. Queuing delay bound

For several queuing mechanisms, queuing_delay_bound is less than the sum of upper bounds on the queuing delays (5,6) at every hop. This occurs with (1) per-flow queuing, and (2) per-class queuing with regulators, as explained in Section 4.2.1, Section 4.2.2, and Section 6.

For other queuing mechanisms the only available value of queuing_delay_bound is the sum of the per-hop queuing delay bounds. In such cases, the computation of per-hop queuing delay bounds must account for the fact that the T-SPEC of a DetNet flow is no longer satisfied at the ingress of a hop, since burstiness increases as one flow traverses one DetNet transit node.

4.2.1. Per-flow queuing mechanisms

With such mechanisms, each flow uses a separate queue inside every node. The service for each queue is abstracted with a guaranteed rate and a latency. For every flow, a per-node delay bound as well as an end-to-end delay bound can be computed from the traffic specification of this flow at its source and from the values of rates and latencies at all nodes along its path. The per-flow queuing is used in IntServ. Details of calculation for IntServ are described in Section 6.5.

4.2.2. Per-class queuing mechanisms

With such mechanisms, the flows that have the same class share the same queue. A practical example is the credit-based shaper defined in section 8.6.8.2 of [IEEE8021Q]. One key issue in this context is how to deal with the burstiness cascade: individual flows that share a resource dedicated to a class may see their burstiness increase, which may in turn cause increased burstiness to other flows downstream of this resource. Computing delay upper bounds for such cases is difficult, and in some conditions impossible [charny2000delay][bennett2002delay]. Also, when bounds are obtained, they depend on the complete configuration, and must be recomputed when one flow is added. (The dynamic calculation, Section 3.1.2.)

A solution to deal with this issue is to reshape the flows at every hop. This can be done with per-flow regulators (e.g. leaky bucket shapers), but this requires per-flow queuing and defeats the purpose of per-class queuing. An alternative is the interleaved regulator, which reshapes individual flows without per-flow queuing ([Specht2016UBS], [IEEE8021Qcr]). With an interleaved regulator, the packet at the head of the queue is regulated based on its (flow) regulation constraints; it is released at the earliest time at which this is possible without violating the constraint. One key feature of per-flow or interleaved regulator is that, it does not increase worst-case latency bounds [le_boudec_theory_2018]. Specifically, when an interleaved regulator is appended to a FIFO subsystem, it does not increase the worst-case delay of the latter.

Figure 2 shows an example of a network with 5 nodes, per-class queuing mechanism and interleaved regulators as in Figure 1. An end-to-end delay bound for flow f, traversing nodes 1 to 5, is calculated as follows:

```
   end_to_end_latency_bound_of_flow_f = C12 + C23 + C34 + S4
```

In the above formula, Cij is a bound on the delay of the queuing subsystem in node i and interleaved regulator of node j, and S4 is a bound on the delay of the queuing subsystem in node 4 for flow f. In fact, using the delay definitions in Section 3.2, Cij is a bound on sum of the delays 1,2,3,6 of node i and 4,5 of node j. Similarly, S4 is a bound on sum of the delays 1,2,3,6 of node 4. A practical example of queuing model and delay calculation is presented Section 6.4.

```
                      f
           ----------------------------->
         +---+   +---+   +---+   +---+   +---+
         | 1 |---| 2 |---| 3 |---| 4 |---| 5 |
         +---+   +---+   +---+   +---+   +---+
            \__C12_/\__C23_/\__C34_/\_S4_/
```

Figure 2: End-to-end delay computation example

REMARK: The end-to-end delay bound calculation provided here gives a much better upper bound in comparison with end-to-end delay bound computation by adding the delay bounds of each node in the path of a flow [TSNwithATS].

4.3. Ingress considerations

A sender can be a DetNet node which uses exactly the same queuing methods as its adjacent DetNet transit node, so that the delay and buffer bounds calculations at the first hop are indistinguishable from those at a later hop within the DetNet domain. On the other hand, the sender may be DetNet unaware, in which case some conditioning of the flow may be necessary at the ingress DetNet transit node.

This ingress conditioning typically consists of a FIFO with an output regulator that is compatible with the queuing employed by the DetNet transit node on its output port(s). For some queuing methods, simply requires added extra buffer space in the queuing subsystem. Ingress conditioning requirements for different queuing methods are mentioned in the sections, below, describing those queuing methods.

4.4. Interspersed non-DetNet transit nodes

It is sometimes desirable to build a network that has both DetNet aware transit nodes and DetNet non-aware transit nodes, and for a DetNet flow to traverse an island of non-DetNet transit nodes, while still allowing the network to offer delay and congestion loss guarantees. This is possible under certain conditions.

In general, when passing through a non-DetNet island, the island causes delay variation in excess of what would be caused by DetNet nodes. That is, the DetNet flow is "lumpier" after traversing the non-DetNet island. DetNet guarantees for delay and buffer requirements can still be calculated and met if and only if the following are true:

1. The latency variation across the non-DetNet island must be bounded and calculable.

2. An ingress conditioning function (Section 4.3) may be required at the re-entry to the DetNet-aware domain. This will, at least, require some extra buffering to accommodate the additional delay variation, and thus further increases the delay bound.

The ingress conditioning is exactly the same problem as that of a sender at the edge of the DetNet domain. The requirement for bounds on the latency variation across the non-DetNet island is typically the most difficult to achieve. Without such a bound, it is obvious that DetNet cannot deliver its guarantees, so a non-DetNet island that cannot offer bounded latency variation cannot be used to carry a DetNet flow.

5. Achieving zero congestion loss

When the input rate to an output queue exceeds the output rate for a sufficient length of time, the queue must overflow. This is congestion loss, and this is what deterministic networking seeks to avoid.

To avoid congestion losses, an upper bound on the backlog present in the regulator and queuing subsystem of Figure 1 must be computed during resource reservation. This bound depends on the set of flows that use these queues, the details of the specific queuing mechanism and an upper bound on the processing delay (4). The queue must contain the packet in transmission plus all other packets that are waiting to be selected for output.

A conservative backlog bound, that applies to all systems, can be derived as follows.

The backlog bound is counted in data units (bytes, or words of multiple bytes) that are relevant for buffer allocation. For every class we need one buffer space for the packet in transmission, plus space for the packets that are waiting to be selected for output. Excluding transmission and preemption times, the packets are waiting in the queue since reception of the last bit, for a duration equal to the processing delay (4) plus the queuing delays (5,6).

Let

- total_in_rate be the sum of the line rates of all input ports that send traffic of any class to this output port. The value of total_in_rate is in data units (e.g. bytes) per second.

- nb_input_ports be the number input ports that send traffic of any class to this output port

- max_packet_length be the maximum packet size for packets of any class that may be sent to this output port. This is counted in data units.

- max_delay456 be an upper bound, in seconds, on the sum of the processing delay (4) and the queuing delays (5,6) for a packet of any class at this output port.

Then a bound on the backlog of traffic of all classes in the queue at this output port is

```
backlog_bound = nb_input_ports * max_packet_length +
total_in_rate* max_delay456
```

6. Queuing techniques

In this section, for simplicity of delay computation, we assume that the T-SPEC or arrival curve [NetCalBook] for each flow at source is leaky bucket. Also, at each relay node, the service for each queue is abstracted with a guaranteed rate and a latency.

6.1. Queuing data model

Sophisticated queuing mechanisms are available in Layer 3 (L3, see, e.g., [RFC7806] for an overview). In general, we assume that "Layer 3" queues, shapers, meters, etc., are precisely the "regulators" shown in Figure 1. The "queuing subsystems" in this figure are not the province solely of bridges; they are an essential part of any DetNet transit node. As illustrated by numerous implementation examples, some of the "Layer 3" mechanisms described in documents such as [RFC7806] are often integrated, in an implementation, with the "Layer 2" mechanisms also implemented in the same node. An integrated model is needed in order to successfully predict the interactions among the different queuing mechanisms needed in a network carrying both DetNet flows and non-DetNet flows.

Figure 3 shows the general model for the flow of packets through the queues of a DetNet transit node. Packets are assigned to a class of service. The classes of service are mapped to some number of regulator queues. Only DetNet/TSN packets pass through regulators. Queues compete for the selection of packets to be passed to queues in the queuing subsystem. Packets again are selected for output from the queuing subsystem.

```
                                    |
   +--------------------------------V----------------------------------+
   |                    Class of Service Assignment                    |
   +--+------+----------+---------+-----------+-----+-------+-------+--+
      |      |          |         |           |     |       |       |
   +--V-+ +--V-+     +--V--+   +--V--+     +--V--+  |       |       |
   |Flow| |Flow|     |Flow |   |Flow |     |Flow |  |       |       |
   |  0 | |  1 | ... |  i  |   | i+1 | ... |  n  |  |       |       |
   | reg| | reg|     | reg |   | reg |     | reg |  |       |       |
   +--+-+ +--+-+     +--+--+   +--+--+     +--+--+  |       |       |
      |      |          |         |           |     |       |       |
   +--V------V----------V--+   +--V-----------V--+  |       |       |
   |  Trans.  selection    |   | Trans. select.  |  |       |       |
   +----------+------------+   +-----+-----------+  |       |       |
              |                      |              |       |       |
           +--V--+                +--V--+        +--V--+ +--V--+ +--V--+
           | out |                | out |        | out | | out | | out |
           |queue|                |queue|        |queue| |queue| |queue|
           |  1  |                |  2  |        |  3  | |  4  | |  5  |
           +--+--+                +--+--+        +--+--+ +--+--+ +--+--+
              |                      |              |       |       |
   +----------V----------------------V--------------V-------V-------V--+
   |                      Transmission selection                       |
   +----------+----------------------+--------------+-------+-------+--+
              |                      |              |       |       |
              V                      V              V       V       V
        DetNet/TSN queue       DetNet/TSN queue    non-DetNet/TSN queues
```

Figure 3: IEEE 802.1Q Queuing Model: Data flow

Some relevant mechanisms are hidden in this figure, and are performed in the queue boxes:

- o Discarding packets because a queue is full.

- Discarding packets marked "yellow" by a metering function, in preference to discarding "green" packets.

Ideally, neither of these actions are performed on DetNet packets. Full queues for DetNet packets should occur only when a flow is misbehaving, and the DetNet QoS does not include "yellow" service for packets in excess of committed rate.

The Class of Service Assignment function can be quite complex, even in a bridge [IEEE8021Q], since the introduction of per-stream filtering and policing ([IEEE8021Q] clause 8.6.5.1). In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a DetNet transit node can utilize any of the following information to assign a packet to a particular class of service (queue):

- Input port.

- Selector based on a rotating schedule that starts at regular, time-synchronized intervals and has nanosecond precision.

- MAC addresses, VLAN ID, IP addresses, Layer 4 port numbers, DSCP. ([I-D.ietf-detnet-ip], [I-D.ietf-detnet-mpls]) (Work items are expected to add MPC and other indicators.)

- The Class of Service Assignment function can contain metering and policing functions.

- MPLS and/or pseudowire ([RFC6658]) labels.

The "Transmission selection" function decides which queue is to transfer its oldest packet to the output port when a transmission opportunity arises.

6.2. Preemption

In [IEEE8021Q] and [IEEE8023], the transmission of a frame can be interrupted by one or more "express" frames, and then the interrupted frame can continue transmission. This frame preemption is modeled as consisting of two MAC/PHY stacks, one for packets that can be interrupted, and one for packets that can interrupt the interruptible packets. The Class of Service (queue) determines which packets are which. Only one layer of preemption is supported -- a transmitter cannot have more than one interrupted frame in progress. DetNet flows typically pass through the interrupting MAC. For those DetNet flows with T-SPEC, latency bound can be calculated by the methods provided in the following sections that accounts for the affect of preemption, according to the specific queuing mechanism that is used in DetNet nodes. Best-effort queues pass through the interruptible MAC, and can thus be preempted.

6.3. Time Aware Shaper

In [IEEE8021Q], the notion of time-scheduling queue gates is described in section 8.6.8.4. On each node, the transmission selection for packets is controlled by time-synchronized gates; each output queue is associated with a gate. The gates can be either open or close. The states of the gates are determined by the gate control list (GCL). The GCL specifies the opening and closing times of the gates. Since the design of GCL should satisfy the requirement of latency upper bounds of all time-sensitive flows, those flows travers a network should have bounded latency, if the traffic and nodes are conformant.

It should be noted that scheduled traffic service relies on a synchronized network and coordinated GCL configuration. Synthesis of GCL on multiple nodes in network is a scheduling problem considering all TSN/DetNet flows traversing the network, which is a non-deterministic polynomial-time hard (NP-hard) problem. Also, at this writing, scheduled traffic service supports no more than eight traffic classes, typically using up to seven priority classes and at least one best effort class.

6.4. Credit-Based Shaper with Asynchronous Traffic Shaping

In the cosidered queuing model, there are four types of flows, namely, control-data traffic (CDT), class A, class B, and best effort (BE) in decreasing order of priority. Flows of classes A and B are together referred to AVB flows. This model is a subset of Time-Sensitive Networking as described next.

Based on the timing model described in Figure 1, the contention occurs only at the output port of a relay node; therefore, the focus of the rest of this subsection is on the regulator and queuing subsystem in the output port of a relay node. The output port performs per-class scheduling with eight classes (queuing subsystems): one for CDT, one for class A traffic, one for class B traffic, and five for BE traffic denoted as BE0-BE4. The queuing policy for each queuing subsystem is FIFO. In addition, each node output port also performs per-flow regulation for AVB flows using an interleaved regulator (IR), called Asynchronous Traffic Shaper [IEEE8021Qcr]. Thus, at each output port of a node, there is one interleaved regulator per-input port and per-class; the interleaved regulator is mapped to the regulator depicted in Figure 1. The detailed picture of scheduling and regulation architecture at a node output port is given by Figure 4. The packets received at a node input port for a given class are enqueued in the respective interleaved regulator at the output port. Then, the packets from all the flows, including CDT and BE flows, are enqueued in queuing subsytem; there is no regulator for such classes.

```
         +--+   +--+ +--+   +--+
         |  |   |  | |  |   |  |
         |IR|   |IR| |IR|   |IR|
         |  |   |  | |  |   |  |
         +-++XXX++-+ +-++XXX++-+
           |     |     |     |
           |     |     |     |
   +---+ +-v-XXX-v-+ +-v-XXX-v-+ +-----+ +-----+ +-----+ +-----+ +-----+
   |   | |         | |         | |Class| |Class| |Class| |Class| |Class|
   |CDT| | Class A | | Class B | | BE4 | | BE3 | | BE2 | | BE1 | | BE0 |
   |   | |         | |         | |     | |     | |     | |     | |     |
   +-+-+ +----+----+ +----+----+ +--+--+ +--+--+ +--+--+ +--+--+ +--+--+
     |        |           |         |       |       |       |       |
     |      +-v-+       +-v-+       |       |       |       |       |
     |      |CBS|       |CBS|       |       |       |       |       |
     |      +-+-+       +-+-+       |       |       |       |       |
     |        |           |         |       |       |       |       |
   +-v--------v-----------v---------v-------V-------v-------v-------v--+
   |                     Strict Priority selection                     |
   +--------------------------------+----------------------------------+
                                    |
                                    V
```

Figure 4: The architecture of an output port inside a relay node with interleaved regulators (IRs) and credit-based shaper (CBS)

Each of the queuing subsystems for class A and B, contains Credit-Based Shaper (CBS). The CBS serves a packet from a class according to the available credit for that class. The credit for each class A or B increases based on the idle slope, and decreases based on the send slope, both of which are parameters of the CBS (Section 8.6.8.2 of [IEEE8021Q]). The CDT and BE0-BE4 flows are served by separate queuing subsystems. Then, packets from all flows are served by a transmission selection subsystem that serves packets from each class based on its priority. All subsystems are non-preemptive. Guarantees for AVB traffic can be provided only if CDT traffic is bounded; it is assumed that the CDT traffic has leaky bucket arrival curve with two parameters r_h as rate and b_h as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r_h t + b_h.

Additionally, it is assumed that the AVB flows are also regulated at their source according to leaky bucket arrival curve. At the source, the traffic satisfies its regulation constraint, i.e. the delay due to interleaved regulator at source is ignored.

At each DetNet transit node implementing an interleaved regulator, packets of multiple flows are processed in one FIFO queue; the packet at the head of the queue is regulated based on its leaky bucket parameters; it is released at the earliest time at which this is possible without violating the constraint. The regulation parameters for a flow (leaky bucket rate and bucket size) are the same at its source and at all DetNet transit nodes along its path.

6.4.1. Delay Bound Calculation

A delay bound of the queuing subsystem ([4] in Figure 1) for an AVB flow of class A or B can be computed if the following condition holds:

sum of leaky bucket rates of all flows of this class at this transit node <= R, where R is given below for every class.

If the condition holds, the delay bounds for a flow of class X (A or B) is d_X and calculated as:

d_X = T_X + (b_t_X-L_min_X)/R_X - L_min_X/c

where L_min_X is the minimum packet lengths of class X (A or B); c is the output link transmission rate; b_t_X is the sum of the b term (bucket size) for all the flows of the class X. Parameters R_X and T_X are calculated as follows for class A and class B, separately:

If the flow is of class A:

R_A = I_A (c-r_h)/ c

T_A = L_nA + b_h + r_h L_n/c)/(c-r_h)

where L_nA is the maximum packet length of class B and BE packets; L_n is the maximum packet length of classes A,B, and BE.

If the flow is of class B:

R_B = I_B (c-r_h)/ c

T_B = (L_BE + L_A + L_nA I_A/(c_h-I_A) + b_h + r_h L_n/c)/(c-r_h) where L_A is the maximum packet length of class A; L_BE is the maximum packet length of class BE.

Then, an end-to-end delay bound of class X (A or B)is calculated by the formula Section 4.2.2, where for Cij:

```
   Cij = d_X
```

More information of delay analysis in such a DetNet transit node is described in [TSNwithATS].

6.4.2. Flow Admission

The delay bound calculation requires some information about each node. For each node, it is required to know the idle slope of CBS for each class A and B (I_A and I_B), as well as the transmission rate of the output link (c). Besides, it is necessary to have the information on each class, i.e. maximum packet length of classes A, B, and BE. Moreover, the leaky bucket parameters of CDT (r_h,b_h) should be known. To admit a flow/flows, their delay requirements should be guaranteed not to be violated. As described in Section 3.1, the two problems, static and dynamic, are addressed separately. In either of the problems, the rate and delay should be guaranteed. Thus,

The static admission control:

The leaky bucket parameters of all flows are known, therefore, for each flow f, a delay bound can be calculated. The computed delay bound for every flow should not be more than its delay requirement. Moreover, the sum of the rate of each flow (r_f) should not be more than the rate allocated to each class (R). If these two conditions hold, the configuration is declared admissible.

The dynamic admission control:

For dynamic admission control, we allocate to every node and class A or B, static value for rate (R) and maximum burstiness (b_t). In addition, for every node and every class A and B, two counters are maintained:

R_acc is equal to the sum of the leaky-bucket rates of all flows of this class already admitted at this node; At all times, we must have:

$$R\_acc <= R, \quad \text{(Eq. 1)}$$

b_acc is equal to the sum of the bucket sizes of all flows of this class already admitted at this node; At all times, we must have:

$$b\_acc <= b\_t. \quad \text{(Eq. 2)}$$

A new flow is admitted at this node, if Eqs. (1) and (2) continue to be satisfied after adding its leaky bucket rate and bucket size to R_acc and b_acc. A flow is admitted in the network, if it is admitted at all nodes along its path. When this happens, all variables R_acc and b_acc along its path must be incremented to reflect the addition of the flow. Similarly, when a flow leaves the network, all variables R_acc and b_acc along its path must be decremented to reflect the removal of the flow.

The choice of the static values of R and b_t at all nodes and classes must be done in a prior configuration phase; R controls the bandwidth allocated to this class at this node, b_t affects the delay bound and the buffer requirement. R must satisfy the constraints given in Annex L.1 of [IEEE8021Q].

6.5. IntServ

Integrated service (IntServ) is an architecture that specifies the elements to guarantee quality of service (QoS) on networks.

The flow, at the source, has a leaky bucket arrival curve with two parameters r as rate and b as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r t + b.

If a resource reservation on a path is applied, a node provides a guaranteed rate R and maximum service latency of T. This can be interpreted in a way that the bits might have to wait up to T before being served with a rate greater or equal to R. The delay bound of the flow traversing the node is T + b / R.

Consider an IntServ path including a sequence of nodes, where the i-th node provides a guaranteed rate R_i and maximum service latency of T_i. Then, the end-to-end delay bound for a flow on this can be calculated as sum(T_i) + b / min(R_i).

If more information about the flow is known, e.g. the peak rate, the delay bound is more complicated; the detail is available in Section 1.4.1 of [NetCalBook].

6.6. Cyclic Queuing and Forwarding

Annex T of [IEEE8021Q] describes Cyclic Queuing and Forwarding (CQF), which provides bounded latency and zero congestion loss using the time-scheduled gates of [IEEE8021Q] section 8.6.8.4. For a given DetNet class of service, a set of two or more buffers is provided at the output queue layer of Figure 3. A cycle time T_c is configured for each class c, and all of the buffer sets in a class swap buffers simultaneously throughout the DetNet domain at that cycle rate, all in phase. In such a mechanism, the regulator, mentioned in Figure 1, is not required.

In the case of two-buffer CQF, each class c has two buffers, namely buffer1 and buffer2. In a cycle (i) when buffer1 accumulates received packets from the node’s reception ports, buffer2 transmits the already stored packets from the previous cycle (i-1). In the next cycle (i+1), buffer2 stores the received packets and buffer1 transmits the packets received in cycle (i). The duration of each cycle is T_c.

The per-hop latency is trivially determined by the cycle time T_c: the packet transmitted from a node at a cycle (i), is transmitted from the next node at cycle (i+1). Hence, the maximum delay experienced by a given packet is from the beginning of cycle (i) to the end of cycle (i+1), or 2T_c; also, the minimum delay is from the end of cycle (i) to the beginning of cycle (i+1), i.e., zero. Then, if the packet traverses h hops, the maximum delay is:

$$(h+1)\ T_c$$

and the minimum delay is:

$$(h-1)\ T_c$$

which gives a latency variation of 2T_c.

The cycle length T_c should be carefully chosen; it needs to be large enough to accomodate all the DetNet traffic, plus at least one maximum interfering packet, that can be received within one cycle. Also, the value of T_c includes a time interval, called dead time (DT), which is the sum of the delays 1,2,3,4 defined in Figure 1. The value of DT guarantees that the last packet of one cycle in a node is fully delivered to a buffer of the next node is the same cycle. A two-buffer CQF is recommended if DT is small compared to T_c. For a large DT, CQF with more buffers can be used.

Ingress conditioning (Section 4.3) may be required if the source of a DetNet flow does not, itself, employ CQF. Since there are no per-flow parameters in the CQF technique, per-hop configuration is not required in the CQF forwarding nodes.

7. References

7.1. Normative References

[I-D.ietf-detnet-ip]
Varga, B., Farkas, J., Berger, L., Fedyk, D., Malis, A., and S. Bryant, "DetNet Data Plane: IP", draft-ietf-detnet-ip-05 (work in progress), February 2020.

[I-D.ietf-detnet-mpls]
Varga, B., Farkas, J., Berger, L., Fedyk, D., Malis, A., Bryant, S., and J. Korhonen, "DetNet Data Plane: MPLS", draft-ietf-detnet-mpls-05 (work in progress), February 2020.

[RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <https://www.rfc-editor.org/info/rfc2212>.

[RFC6658] Bryant, S., Ed., Martini, L., Swallow, G., and A. Malis, "Packet Pseudowire Encapsulation over an MPLS PSN", RFC 6658, DOI 10.17487/RFC6658, July 2012, <https://www.rfc-editor.org/info/rfc6658>.

[RFC7806] Baker, F. and R. Pan, "On Queuing, Marking, and Dropping", RFC 7806, DOI 10.17487/RFC7806, April 2016, <https://www.rfc-editor.org/info/rfc7806>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

[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>.

7.2. Informative References

[bennett2002delay]
J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, and J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding", <https://dl.acm.org/citation.cfm?id=581870>.

[charny2000delay]
A. Charny and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling", <https://link.springer.com/chapter/10.1007/3-540-39939-9_1>.

[IEEE8021Q]
IEEE 802.1, "IEEE Std 802.1Q-2018: IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks", 2018, <http://ieeexplore.ieee.org/document/8403927>.

[IEEE8021Qcr]
IEEE 802.1, "IEEE P802.1Qcr: IEEE Draft Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment: Asynchronous Traffic Shaping", 2017, <http://www.ieee802.org/1/files/private/cr-drafts/>.

[IEEE8021TSN]
IEEE 802.1, "IEEE 802.1 Time-Sensitive Networking (TSN) Task Group", <http://www.ieee802.org/1/>.

[IEEE8023]
IEEE 802.3, "IEEE Std 802.3-2018: IEEE Standard for Ethernet", 2018, <http://ieeexplore.ieee.org/document/8457469>.

[le_boudec_theory_2018]
J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", <https://ieeexplore.ieee.org/document/8519761>.

[NetCalBook]
J.-Y. Le Boudec and P. Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <https://ica1www.epfl.ch/PS_files/NetCal.htm>.

[Specht2016UBS]
J. Specht and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", <https://ieeexplore.ieee.org/abstract/document/7557870>.

[TSNwithATS]
E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "End-to-end Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping", <https://arxiv.org/abs/1804.10608/>.

Authors' Addresses

Norman Finn
Huawei Technologies Co. Ltd
3101 Rio Way
Spring Valley, California 91977
US

Phone: +1 925 980 6430
Email: nfinn@nfinnconsulting.com

Jean-Yves Le Boudec
EPFL
IC Station 14
Lausanne EPFL 1015
Switzerland

Email: jean-yves.leboudec@epfl.ch

Ehsan Mohammadpour
EPFL
IC Station 14
Lausanne EPFL 1015
Switzerland

Email: ehsan.mohammadpour@epfl.ch

Jiayi Zhang
Huawei Technologies Co. Ltd
Q27, No.156 Beiqing Road
Beijing 100095
China

Email: zhangjiayi11@huawei.com

Balazs Varga
Ericsson
Konyves Kalman krt. 11/B
Budapest 1097
Hungary

Email: balazs.a.varga@ericsson.com

Janos Farkas
Ericsson
Konyves Kalman krt. 11/B
Budapest 1097
Hungary

Email: janos.farkas@ericsson.com

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 will expire on April 15, 2021.

Copyright Notice

Table of Contents

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 Appliction 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:

- DetNet flow identification

- 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:

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

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

4. DetNet Flow Aggregation

DetNet provides the capability of flow aggregation to improve scaleability of DetNet data, management and control planes. Aggregated flows can be viewed by the 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:

- Aggregation flow encapsulation/decapsulation/identification

- Mapping individual DetNet flows to an aggregated flow

- Changing traffic specification parameters for aggregated flow

The following cases of DetNet aggregation are supported:

- 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.

- 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.

- 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.

- 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.

- 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:

```
                 +-----------+
                 |ietf-detnet|
                 +-----+-----+
                       |
         +-------------+---------------+
         |             |               |
   +-----+-----+ +-----+-----+ +-------+------+
   | App Flows | |service s-l| |forwarding s-l|
   +-----------+ +-----------+ +--------------+
```

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 filed can choice 0 bit, 16bit, 28 bit
               filed";
            }
          }

        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;
            }
          }
        }

        container outgoing {
          description
            "The DetNet forwarding sub-layer outbound configuration.";
          choice outgoing-options {
            mandatory true;
            description
              "";
            case detnet-forwarding-outgoing {
              uses detnet-forwarding-next-hop-content;
            }

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

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

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

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

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

            case detnet-forwarding-operation {
              uses forwarding-sub-layer-ref;
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 20, 2021

X. Geng
M. Chen
Huawei Technologies
Y. Ryoo
ETRI
D. Fedyk
LabN Consulting, L.L.C.
R. Rahman
Individual
Z. Li
China Mobile

November 16, 2020

Deterministic Networking (DetNet) Configuration YANG Model
draft-ietf-detnet-yang-09

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 will expire on May 20, 2021.

Copyright Notice

Table of Contents

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 Appliction 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:

- DetNet flow identification

- 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 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 scaleability of DetNet data, management and control planes. Aggregated flows can be viewed by the 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:

- 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.

- 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.

- 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.

- 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.

- 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:

```
                     +-----------+
                     |ietf-detnet|
                     +-----+-----+
                           |
             +-------------+---------------+
             |             |               |
       +-----+-----+ +-----+-----+ +-------+------+
       | App Flows | |service s-l| |forwarding s-l|
       +-----------+ +-----------+ +--------------+
```

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
   +--rw detnet
      +--rw traffic-profile* [profile-name]
      |  +--rw profile-name                   string
      |  +--rw traffic-requirements
      |  |  +--rw min-bandwidth?                    uint64
      |  |  +--rw max-latency?                      uint32
      |  |  +--rw max-latency-variation?            uint32
      |  |  +--rw max-loss?                         uint32
      |  |  +--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-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?
      |     |     |  |       ethertypes:ethertype
      |     |     |  +--rw vlan-id?
      |     |     |  |       dot1q-types:vlanid
      |     |     |  +--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?     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?     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-sub-layer
      |  +--rw service-sub-layer-list* [name]
      |     +--rw name                      string
      |     +--rw service-rank?             uint8
      |     +--rw traffic-profile?          traffic-profile-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-type
      |     |  +--rw (incoming-type)
      |     |     +--:(app-flow)
      |     |     |  +--rw app-flow
      |     |     |     +--rw flow-list*   app-flow-ref
      |     |     +--:(service)
      |     |     |  +--rw service
      |     |     |     +--rw service-sub-layer*
      |     |     |             service-sub-layer-ref
      |     |     |
      |     |     +--:(forwarding)
      |     |     |  +--rw forwarding
      |     |     |     +--rw forwarding-sub-layer*
      |     |     |             forwarding-sub-layer-ref
      |     |     +--:(service-identification)
      |     |        +--rw 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?     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?     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
      |     +--rw outgoing-type
      |        +--rw (outgoing-type)
      |           +--:(forwarding-sub-layer)
      |           |  +--rw forwarding-sub-layer
      |           |     +--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
      |           +--:(service-sub-layer)
      |           |  +--rw service-sub-layer
      |           |     +--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
      |           +--:(upper-app-flow)
      |           |  +--rw upper-app-flow
      |           |     +--rw flow-list*   app-flow-ref
      |           +--:(upper-service-sub-layer)
      |           |  +--rw upper-service-sub-layer
      |           |     +--rw service-sub-layer*
      |           |             service-sub-layer-ref
      |           +--:(upper-forwarding-sub-layer)
      |              +--rw upper-forwarding-sub-layer
      |                 +--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-type
            |  +--rw (incoming-type)
            |     +--:(service-sub-layer)
            |     |  +--rw service-sub-layer
            |     |     +--ro sub-layer-list*   service-sub-layer-ref
            |     +--:(upper-forwarding-sub-layer)
            |     |  +--rw forwarding-sub-layer*
            |     |          forwarding-sub-layer-ref
            |     +--:(lower-forwarding-sub-layer)
            |        +--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?     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?     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
            |
            +--rw outgoing-type
               +--rw (outgoing-type)
                  +--:(interface)
                  |  +--rw interface
                  |     +--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
                  +--:(service)
                  |  +--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
                  +--:(forwarding)
                  |  +--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
                  +--:(upper-service)
                  |  +--rw service-sub-layer*
                  |          service-sub-layer-ref
                  +--:(upper-forwarding)
                     +--rw forwarding-sub-layer*
                             forwarding-sub-layer-ref
```

7. DetNet Configuration YANG Model

```
<CODE BEGINS>
module ietf-detnet{
  namespace "urn:ietf:params:xml:ns:yang:ietf-detnet";
  prefix ietf-detnet;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-ethertypes {
    prefix ethertypes;
  }
  import ietf-routing-types {
    prefix rt-types;
  }
  import ietf-packet-fields {
    prefix packet-fields;
  }
  import ietf-interfaces {
    prefix if;
  }
  import ieee802-dot1q-types{
    prefix dot1q-types;
  }

  organization
    "IETF DetNet Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/detnet/>
     WG List:  <mailto: detnet@ietf.org>
     WG Chair: Lou Berger
                 <mailto:lberger@labn.net>

                 Janos Farkas
                 <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 2020-11-12 {
    description
      "initial revision";
    reference
      "RFC XXXX: draft-ietf-detnet-yang-09";
  }

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

  identity none {
    base app-status;
    description
      "Application no ingress/egress";
    reference
      "draft-ietf-detnet-flow-information-model Section 5.8";
  }

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

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

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

  identity partial-failed {
    base app-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 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 traffic-profile-ref {
    type leafref {
      path "/ietf-detnet:detnet"
         + "/ietf-detnet:traffic-profile"
         + "/ietf-detnet:profile-name";
    }
  }

  typedef ipsec-spi {
    type uint32 {
      range "1..max";
    }
    description
      "IPsec Security Parameters Index";
    reference
      "IETF RFC 6071";
  }

  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";
      }
    }
  }

  grouping ip-header {
    description
      "The IPv4/IPv6 packet header information";
    leaf src-ip-address {
      type inet:ip-address;
      description
        "The source IP address in the header";
    }
    leaf dest-ip-address {
      type inet:ip-address;
      description
        "The destination IP address in 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 ethertypes:ethertype;
      description
        "The Ethernet packet type value of the Ethernet header";
    }
    leaf vlan-id {
      type dot1q-types:vlanid;
      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
        "IPsec Security Parameters Index of the Security Association";
      reference
        "IETF RFC 6071";
    }
  }

  grouping mpls-flow-identification {
    description
      "The MPLS packet header identification information";
    choice label-space {
      description
        "Designates the label space being used.";
      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 Section 4.1";
      leaf interval {
        type uint32;
        units microseconds;
        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.";
      reference
        "draft-ietf-detnet-flow-information-model Section 4.2";
      leaf min-bandwidth {
        type uint64;
        units bytes-per-second;
        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;
        units microseconds;
        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 uint32;
        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;
        units packets;
        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;
        units packets;
        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 flow-list{
      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-layers
         that have 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-layers
         that have 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 or
         forwarding";
        uses rt-types:mpls-label-stack;
      }
      case detnet-ip-header {
        description
          "IPv4/IPv6 packet header for DetNet IP encapsulation";
        uses ip-header;
      }
    }
  }

  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
                  "The value if the index of for a hop.";
              }
              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
                  "The value if the index of for a hop.";
              }
              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-name";
        description
          "A traffic profile";
        leaf profile-name {
          type string;
          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
            "Applications attached to this profile";
        }
        leaf-list member-services {
          type service-sub-layer-ref;
          config false;
          description
            "Services 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";
        reference
          "draft-ietf-detnet-flow-information-model Section Section 4.1";
        list app-flow {
          key "name";
          description
            "A unique (management) identifier of the App-flow.";
          leaf name {
            type string;
            description
              "A unique (management) identifier of the App-flow.";
            reference
              "draft-ietf-detnet-flow-information-model
               Sections 4.1, 5.1";
          }
          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.";
            reference
              "draft-ietf-detnet-flow-information-model Section 4.2";
          }
          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 app-status;
              }
              config false;
              description
                "Status of ingress application flow";
              reference
                "draft-ietf-detnet-flow-information-model
```

```
               Sections 4.1, 5.8";
          }
          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;
            }
          }
        }
      }
    }
    container service-sub-layer {
      description
        "The DetNet service sub-layer configuration";
      list service-sub-layer-list {
        key "name";
        description
          "Services are indexed by name";
        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";
          reference
```

```
            "draft-ietf-detnet-flow-information-model Section 5.7";
        }
        leaf traffic-profile {
          type traffic-profile-ref;
          description
            "The Traffic Profile for this service";
        }
        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";
            reference
              "draft-ietf-detnet-data-plane-framework Section 4.3";
          }
          leaf sequence-number-length {
            type sequence-number-field;
            description
              "Sequence number field length can be one of 0 (none),
               16 bits or 28 bits.";
          }
        }
        leaf service-operation-type {
          type service-operation-type;
        }
        container incoming-type {
          description
            "The DetNet service sub-layer incoming configuration.";
          choice incoming-type {
            mandatory true;
            description
              "";
            container app-flow {
              description
                "This service sub-layer is related to
                 the app-flows of the upper layer
                 and provide ingress proxy or ingress aggregation
                 at the ingress node.";
              uses app-flows-ref;
            }
            container service {
              description
                "This service sub-layer is related to
                 the service sub-layer of the upper layer
                 and provide service-to-service aggregation
                 at the ingress node or relay node.";
              uses service-sub-layer-ref;
            }
            container forwarding {
              description
                "This service sub-layer is related to
                 the forwarding sub-layer of the upper layer
                 and provide forwarding-to-service aggregation
                 at the ingress node or relay node.";
              uses forwarding-sub-layer-ref;
            }
            container service-identification {
              description
                "This service sub-layer is related to
                 the service or forwarding sub-layer of the lower layer
                 and provide DetNet service relay or termination
                 at the relay node or egress node.";
              uses detnet-flow-spec;
            }
          }
        }
        container outgoing-type {
          description
            "The DetNet service sub-layer outgoing configuration.";
          choice outgoing-type {
            mandatory true;
            description
              "";
            container forwarding-sub-layer {
              description
                "This service sub-layer is sent to the forwarding
                 sub-layers of the lower layer for DetNet service
                 forwarding or service-to-forwarding aggregation at
                 the ingress node or relay node.  When the operation
                 type is service-initiation, The service sub-layer
                 encapsulates the DetNet Control-Word and services
                 label, which are for individual DetNet flow when the
                 incoming type is app-flow and for aggregated DetNet
                 flow when the incoming type is service or
                 forwarding.  The service sub-layer swaps the service
                 label when the operation type is service-relay.";
              list service-outgoing-list {
                key "service-outgoing-index";
                description
                  "list of the outgoing service
                   that separately for each node
                   where services will be eliminated";
                leaf service-outgoing-index {
                  type uint8;
                }
                uses detnet-header;
                list next-layer {
                  key "index";
                  description
                    "list of the forwarding-sub-layer
                     for replicate to multiple paths";
                  leaf index {
                    type uint8;
                  }
                  leaf forwarding-sub-layer {
                    type forwarding-sub-layer-ref;
                    description
                      "forwarding-sub-layer reference point";
                  }
                }
              }
            }
            container service-sub-layer {
              description
                "This service sub-layer is sent to the service
                 sub-layers of the lower layer for service-to-service
                 aggregation at the ingress node or relay node.  The
                 service sub-layer encapsulates the DetNet
                 Control-Word and S-label when the operation type is
                 service-initiation, and swaps the S-label when the
                 operation type is service-relay.";
              leaf aggregation-service-sub-layer {
                type service-sub-layer-ref;
                description
                  "reference point of the service-sub-layer
                   at which this service will be aggregated";
              }
              container service-label {
                uses rt-types:mpls-label-stack;
              }
            }
            container upper-app-flow {
              description
                "This service sub-layer is sent to the app-flow of
                 the upper layer for egress proxy at the egress node,
                 and decapsulates the DetNet Control-Word and S-label
                 for individual DetNet service.  This outgoing type
                 only can be chosen when the operation type is
                 service-termination.";
              uses app-flows-ref;
            }
            container upper-service-sub-layer {
              description
```

```
                "This service sub-layer is sent to the service
                 sub-layer of the upper layer for service-to-service
                 disaggregation at the relay node or egress node, and
                 decapsulates the DetNet Control-Word and A-label for
                 aggregated DetNet service.  This outgoing type only
                 can be chosen when the operation type is
                 service-termination.";
              uses service-sub-layer-ref;
            }
            container upper-forwarding-sub-layer {
              description
                "This service sub-layer is sent to the forwarding
                 sub-layer of the upper layer for
                 forwarding-to-service disaggregation at the relay
                 node or egress node, and decapsulates the DetNet
                 Control-Word and A-label for aggregated DetNet
                 service.  This outgoing type only can be chosen when
                 the operation type is service-termination";
              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-type {
          description
            "The DetNet forwarding sub-layer incoming configuration.";
          choice incoming-type {
            mandatory true;
            description
              "Cases of incoming types";
            container service-sub-layer {
              description
                "This forwarding sub-layer is related to the service
                 sub-layers of the upper layer and provide DetNet
                 forwarding or service-to-forwarding aggregation at
                 the ingress node or relay node.";
              leaf-list sub-layer-list {
                type service-sub-layer-ref;
                config false;
                description
                  "";
              }
            }
            case upper-forwarding-sub-layer {
              description
                "This forwarding sub-layer is related to the
                 forwarding sub-layer of the upper layer and provide
                 forwarding-to-forwarding aggregation at the ingress
                 node or relay node or transit node.";
              uses forwarding-sub-layer-ref;
            }
            case lower-forwarding-sub-layer {
            //case forwarding-identification {
              description
                "This forwarding sub-layer is related to all of the
                 lower layer and provide DetNet forwarding swap or
                 termination at the transit node or relay node or
                 egress node.";
              leaf interface {
                type if:interface-ref;
                description
                  "This is the interface associated with the forwarding
                   sub-layer";
              }
              uses detnet-flow-spec;
            }
          }
        }
        container outgoing-type {
          description
            "The DetNet forwarding sub-layer outbound configuration.";
          choice outgoing-type {
            mandatory true;
            description
              "";
            container interface {
              description
                "This forwarding sub-layer is sent to the interface
                 for send to next-hop at the ingress node or relay
                 node or transit node.";
              uses detnet-forwarding-next-hop-content;
            }
            case service {
              description
                "This forwarding sub-layer is sent to the service
                 sub-layers of the lower layer for
                 forwarding-to-service aggregation at the ingress
                 node or relay node.";
              leaf aggregation-service-sub-layer {
                type service-sub-layer-ref;
              }
              container optional-forwarding-label {
                uses rt-types:mpls-label-stack;
              }
            }
            case forwarding {
              description
                "This forwarding sub-layer is sent to the forwarding
                 sub-layers of the lower layer for
                 forwarding-to-forwarding aggregation at the ingress
                 node or relay node or transit node.";
              leaf aggregation-forwarding-sub-layer {
                type forwarding-sub-layer-ref;
              }
              container forwarding-label {
                uses rt-types:mpls-label-stack;
              }
            }
            case upper-service {
              description
                "This forwarding sub-layer is sent to the service
                 sub-layer of the upper layer and decapsulate the
                 F-label for DetNet service or service-to-forwarding
                 disaggregation at the relay node or egress node.
                 This outgoing type only can be chosen when the
                 operation type is pop-and-lookup";
              uses service-sub-layer-ref;
            }
            case upper-forwarding {
              description
                "This forwarding sub-layer is sent to the forwarding
                 sub-layer of the upper layer and decapsulate the
                 F-label for forwarding-to-forwarding disaggregation
                 at the transit node or relay node or egress node.
                 This outgoing type only can be chosen when the
                 operation type is pop-and-lookup";
              uses forwarding-sub-layer-ref;
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

8. Open Issues

There are some open issues that are still under discussion:

- Terminology.

- Security Considerations.

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-11 (work in progress), October 2020.

Appendix A. Examples

The following examples are provided.

- A simple DetNet application illustrting multiplexing of Application Flows.

- A case of Forwarding sub-layer aggregation using a single forwarding sublayer.

- A case of Service sub-layer aggregation with and aggrgation label.

A.1. Example JSON Configuration

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2020-10-02T23:59:00Z"
        }
      },
      {
        "name": "eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2020-10-02T23:59:00Z"
```

```
           }
         },
         {
           "name": "eth2",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth3",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth4",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         }
       ]
     },
     "ietf-detnet:detnet": {
       "app-flows": {
         "app-flow": [
           {
             "name": "app-0",
             "app-flow-bidir-congruent": false,
             "outgoing-service": "ssl-1",
             "traffic-profile": "pf-1",
             "ingress": {
               "app-flow-status": "ready",
               "interface": "eth0",
               "src-ip-prefix": "1.1.1.1/32",
               "dest-ip-prefix": "8.8.8.8/32",
               "traffic-class": 6
             }
           },
           {
             "name": "app-1",
             "app-flow-bidir-congruent": false,
             "outgoing-service": "ssl-1",
             "traffic-profile": "pf-1",
             "ingress": {
               "app-flow-status": "ready",
               "interface": "eth0",
               "src-ip-prefix": "1.1.1.1/32",
               "dest-ip-prefix": "8.8.8.8/32",
               "traffic-class": 7
             }
           }
         ]
       },
       "traffic-profile": [
         {
           "profile-name": "pf-1",
           "traffic-requirements": {
             "min-bandwidth": "100000000",
             "max-latency": 100000000,
             "max-latency-variation": 200000000,
             "max-loss": 2,
             "max-consecutive-loss-tolerance": 5,
             "max-misordering": 0
           },
           "traffic-specification": {
             "interval": 5,
             "max-packets-per-interval": 10,
             "max-payload-size": 1500,
             "average-packets-per-interval": 5,
             "average-payload-size": 1000
           },
           "member-applications": [
             "app-0",
             "app-1"
           ]
         },
         {
           "profile-name": "pf-2",
           "traffic-requirements": {
             "min-bandwidth": "200000000",
             "max-latency": 100000000,
             "max-latency-variation": 200000000,
             "max-loss": 2,
             "max-consecutive-loss-tolerance": 5,
             "max-misordering": 0
           },
           "traffic-specification": {
             "interval": 5,
             "max-packets-per-interval": 10,
             "max-payload-size": 1500,
             "average-packets-per-interval": 5,
             "average-payload-size": 1000
           },
           "member-services": [
             "ssl-1"
           ]
         },
         {
           "profile-name": "pf-3",
           "traffic-specification": {
             "interval": 5,
             "max-packets-per-interval": 10,
             "max-payload-size": 1500
           },
           "member-forwarding-sublayers": [
             "fsl-1"
           ]
         }
       ],
       "service-sub-layer": {
         "service-sub-layer-list": [
           {
             "name": "ssl-1",
             "service-rank": 10,
             "traffic-profile": "pf-2",
             "service-operation-type": "service-initiation",
             "service-protection": {
               "service-protection-type": "none",
               "sequence-number-length": "long-sn"
             },
             "incoming-type": {
               "app-flow": {
                 "flow-list": [
                   "app-0",
                   "app-1"
                 ]
               }
             },
             "outgoing-type": {
               "forwarding-sub-layer": {
                 "service-outgoing-list": [
                   {
                     "service-outgoing-index": 0,
                     "mpls-label-stack": {
                       "entry": [
                         {
                           "id": 0,
                           "label": 100
                         }
                       ]
                     },
                     "next-layer": [
                       {
                         "index": 0,
                         "forwarding-sub-layer": "fsl-1"
                       }
                     ]
                   }
                 ]
               }
             }
           }
         ]
       }
     }
   }
```

Figure 1: Example DetNet JSON configuration

A.2. Example XML Config: Aggregation using a Forwarding Sublayer

```
<interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ia="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-10-02T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth1</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-10-02T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth2</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-10-02T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth3</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-10-02T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth4</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-10-02T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
  </interfaces>
<detnet
  xmlns="urn:ietf:params:xml:ns:yang:ietf-detnet">
  <app-flows>
    <app-flow>
      <name>app-1</name>
      <app-flow-bidir-congruent>false</app-flow-bidir-congruent>
      <outgoing-service>ssl-1</outgoing-service>
       <traffic-profile>1</traffic-profile>
      <ingress>
        <interface>eth0</interface>
        <src-ip-prefix>1.1.1.1/32</src-ip-prefix>
        <dest-ip-prefix>8.8.8.8/32</dest-ip-prefix>
      </ingress>
    </app-flow>
    <app-flow>
      <name>app-2</name>
      <app-flow-bidir-congruent>false</app-flow-bidir-congruent>
      <outgoing-service>ssl-2</outgoing-service>
       <traffic-profile>1</traffic-profile>
      <ingress>
        <interface>eth1</interface>
        <src-ip-prefix>1.1.1.2/32</src-ip-prefix>
        <dest-ip-prefix>8.8.8.9/32</dest-ip-prefix>
      </ingress>
    </app-flow>
  </app-flows>
  <traffic-profile>
    <profile-name>1</profile-name>
    <traffic-requirements>
```

```
      <min-bandwidth>100000000</min-bandwidth>
      <max-latency>100000000</max-latency>
      <max-latency-variation>200000000</max-latency-variation>
      <max-loss>2</max-loss>
      <max-consecutive-loss-tolerance>5</max-consecutive-loss-tolerance>
      <max-misordering>0</max-misordering>
    </traffic-requirements>
    <member-applications>app-1</member-applications>
    <member-applications>app-2</member-applications>
  </traffic-profile>
  <traffic-profile>
    <profile-name>2</profile-name>
    <traffic-requirements>
      <min-bandwidth>100000000</min-bandwidth>
      <max-latency>100000000</max-latency>
      <max-latency-variation>200000000</max-latency-variation>
      <max-loss>2</max-loss>
      <max-consecutive-loss-tolerance>5</max-consecutive-loss-tolerance>
      <max-misordering>0</max-misordering>
    </traffic-requirements>
    <member-services>ssl-1</member-services>
    <member-services>ssl-2</member-services>
  </traffic-profile>
  <traffic-profile>
    <profile-name>3</profile-name>
    <traffic-specification>
      <interval>5</interval>
      <max-packets-per-interval>10</max-packets-per-interval>
      <max-payload-size>1500</max-payload-size>
    </traffic-specification>
    <member-forwarding-sublayers>afl-1</member-forwarding-sublayers>
  </traffic-profile>
  <service-sub-layer>
    <service-sub-layer-list>
      <name>ssl-1</name>
      <service-rank>10</service-rank>
      <traffic-profile>2</traffic-profile>
      <service-operation-type>service-initiation</service-
operation-type>
      <service-protection>
        <service-protection-type>none</service-protection-type>
        <sequence-number-length>long-sn</sequence-number-length>
      </service-protection>
     <incoming-type>
        <app-flow>
          <flow-list>app-1</flow-list>
        </app-flow>
      </incoming-type>
      <outgoing-type>
        <forwarding-sub-layer>
          <service-outgoing-list>
            <service-outgoing-index>0</service-outgoing-index>
            <mpls-label-stack>
              <entry>
                <id>0</id>
                <label>100</label>
              </entry>
            </mpls-label-stack>
            <next-layer>
              <index>0</index>
              <forwarding-sub-layer>afl-1</forwarding-sub-layer>
            </next-layer>
          </service-outgoing-list>
        </forwarding-sub-layer>
      </outgoing-type>
    </service-sub-layer-list>
    <service-sub-layer-list>
      <name>ssl-2</name>
      <service-rank>10</service-rank>
      <traffic-profile>2</traffic-profile>
      <service-operation-type>service-initiation</service-
operation-type>
      <service-protection>
        <service-protection-type>none</service-protection-type>
        <sequence-number-length>long-sn</sequence-number-length>
      </service-protection>
     <incoming-type>
        <app-flow>
          <flow-list>app-2</flow-list>
        </app-flow>
      </incoming-type>
      <outgoing-type>
        <forwarding-sub-layer>
          <service-outgoing-list>
            <service-outgoing-index>0</service-outgoing-index>
            <mpls-label-stack>
              <entry>
                <id>0</id>
                <label>103</label>
              </entry>
            </mpls-label-stack>
            <next-layer>
              <index>0</index>
              <forwarding-sub-layer>afl-1</forwarding-sub-layer>
            </next-layer>
          </service-outgoing-list>
        </forwarding-sub-layer>
      </outgoing-type>
    </service-sub-layer-list>
    </service-sub-layer>
    <forwarding-sub-layer>
    <forwarding-sub-layer-list>
      <name>afl-1</name>
      <traffic-profile>3</traffic-profile>
      <forwarding-operation-type>impose-and-forward</forwarding-
operation-type>
      <incoming-type>
        <service-sub-layer>
          <sub-layer-list>ssl-1</sub-layer-list>
          <sub-layer-list>ssl-2</sub-layer-list>
        </service-sub-layer>
      </incoming-type>
      <outgoing-type>
        <interface>
        <mpls-label-stack>
          <entry>
            <id>0</id>
            <label>10000</label>
          </entry>
        </mpls-label-stack>
          <outgoing-interface>eth2</outgoing-interface>
        </interface>
      </outgoing-type>
    </forwarding-sub-layer-list>
    </forwarding-sub-layer>
</detnet>
```

Figure 2: Example DetNet XML configuration

A.3. Example JSON Service Aggregation Configuration

```
   {
     "ietf-interfaces:interfaces": {
       "interface": [
         {
           "name": "eth0",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth1",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth2",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth3",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         },
         {
           "name": "eth4",
           "type": "iana-if-type:ethernetCsmacd",
           "oper-status": "up",
           "statistics": {
             "discontinuity-time": "2020-10-02T23:59:00Z"
           }
         }
       ]
     },
     "ietf-detnet:detnet": {
       "app-flows": {
         "app-flow": [
           {
             "name": "app-1",
             "app-flow-bidir-congruent": false,
             "outgoing-service": "ssl-1",
             "traffic-profile": "1",
             "ingress": {
               "interface": "eth0",
               "src-ip-prefix": "1.1.1.1/32",
               "dest-ip-prefix": "8.8.8.8/32"
             }
           },
           {
             "name": "app-2",
             "app-flow-bidir-congruent": false,
             "outgoing-service": "ssl-2",
             "traffic-profile": "1",
             "ingress": {
               "interface": "eth1",
               "src-ip-prefix": "1.1.1.2/32",
               "dest-ip-prefix": "8.8.8.9/32"
             }
           }
         ]
       },
       "traffic-profile": [
         {
           "profile-name": "1",
           "traffic-requirements": {
             "min-bandwidth": "100000000",
             "max-latency": 100000000,
             "max-latency-variation": 200000000,
             "max-loss": 2,
             "max-consecutive-loss-tolerance": 5,
             "max-misordering": 0
           },
           "member-applications": [
             "app-1",
             "app-2"
           ]
         },
         {
           "profile-name": "2",
           "traffic-requirements": {
             "min-bandwidth": "100000000",
             "max-latency": 100000000,
             "max-latency-variation": 200000000,
             "max-loss": 2,
             "max-consecutive-loss-tolerance": 5,
             "max-misordering": 0
           },
           "member-services": [
             "ssl-1",
             "ssl-2"
           ]
         },
         {
           "profile-name": "3",
           "traffic-specification": {
             "interval": 5,
             "max-packets-per-interval": 10,
             "max-payload-size": 1500
           },
           "member-forwarding-sublayers": [
             "afl-1"
           ]
         }
       ],
       "service-sub-layer": {
         "service-sub-layer-list": [
           {
             "name": "ssl-1",
             "service-rank": 10,
             "traffic-profile": "2",
             "service-protection": {
               "service-protection-type": "none",
               "sequence-number-length": "long-sn"
             },
             "service-operation-type": "service-initiation",
             "incoming-type": {
               "app-flow": {
                 "flow-list": [
                   "app-1"
                 ]
               }
             },
             "outgoing-type": {
               "service-sub-layer": {
                 "aggregation-service-sub-layer": "asl-1",
                 "service-label": {
                   "mpls-label-stack": {
                     "entry": [
                       {
                         "id": 0,
                         "label": 102
                       }
                     ]
                   }
                 }
               }
             }
           },
           {
             "name": "ssl-2",
             "service-rank": 10,
             "traffic-profile": "2",
             "service-operation-type": "service-initiation",
             "service-protection": {
               "service-protection-type": "none",
               "sequence-number-length": "long-sn"
             },
             "incoming-type": {
               "app-flow": {
                 "flow-list": [
                   "app-2"
                 ]
               }
             },
             "outgoing-type": {
               "service-sub-layer": {
                 "aggregation-service-sub-layer": "asl-1",
                 "service-label": {
                   "mpls-label-stack": {
                     "entry": [
                       {
                         "id": 0,
                         "label": 105
                       }
                     ]
                   }
                 }
               }
             }
           },
           {
             "name": "asl-1",
             "service-rank": 10,
             "service-protection": {
               "service-protection-type": "none",
               "sequence-number-length": "long-sn"
             },
             "incoming-type": {
               "service": {
                 "service-sub-layer": [
                   "ssl-1",
                   "ssl-2"
                 ]
               }
             },
             "outgoing-type": {
               "forwarding-sub-layer": {
                 "service-outgoing-list": [
                   {
                     "service-outgoing-index": 0,
                     "mpls-label-stack": {
                       "entry": [
                         {
                           "id": 0,
                           "label": 1000
                         }
                       ]
                     },
                     "next-layer": [
                       {
                         "index": 0,
                         "forwarding-sub-layer": "afl-1"
                       }
                     ]
                   }
                 ]
               }
             }
           }
         ]
       },
       "forwarding-sub-layer": {
         "forwarding-sub-layer-list": [
           {
             "name": "afl-1",
             "traffic-profile": "3",
             "forwarding-operation-type": "impose-and-forward",
             "outgoing-type": {
               "interface": {
                 "outgoing-interface": "eth2",
                 "mpls-label-stack": {
                   "entry": [
                     {
                       "id": 0,
                       "label": 20000
                     }
                   ]
                 }
               }
             }
           }
         ]
       }
     }
   }
```

Figure 3: Example DetNet JSON Service Aggregation

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
Individual

Email: reshad@yahoo.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2021

F. Theoleyre
CNRS
G. Papadopoulos
IMT Atlantique
G. Mirsky
ZTE Corp.
CJ. Bernardos
UC3M

October 25, 2020

Operations, Administration and Maintenance (OAM) features for DetNet
draft-theoleyre-detnet-oam-support-00

Abstract

Deterministic Networking (DetNet), as defined in RFC 8655, is aimed to provide a bounded end-to-end latency on top of the network infrastructure, comprising both Layer 2 bridged and Layer 3 routed segments. This document's primary purpose is to detail the specific requirements of the Operation, Administration, and Maintenance (OAM) recommended to maintain a deterministic network. With the implementation of the OAM framework in DetNet, an operator will have a real-time view of the network infrastructure regarding the network's ability to respect the Service Level Objective (SLO), such as packet delay, delay variation, and packet loss ratio, assigned to each data flow.

Status of This Memo

Copyright Notice

Table of Contents

1. TEMPORARY EDITORIAL NOTES

This document is an Internet Draft, so it is work-in-progress by nature. It contains the following work-in-progress elements:

- "TODO" statements are elements which have not yet been written by the authors for some reason (lack of time, ongoing discussions with no clear consensus, etc). The statement does indicate that the text will be written at some time.

2. Introduction

Deterministic Networking (DetNet) [RFC8655] has proposed to provide a bounded end-to-end latency on top of the network infrastructure, comprising both Layer 2 bridged and Layer 3 routed segments. Their work encompasses the data plane, OAM, time synchronization, management, control, and security aspects.

Operations, Administration, and Maintenance (OAM) Tools are of primary importance for IP networks [RFC7276]. DetNet OAM should provide a toolset for fault detection, localization, and performance measurement.

This document's primary purpose is to detail the specific requirements of the OAM features recommended to maintain a deterministic/reliable network. Specifically, it investigates the requirements for a deterministic network, supporting critical flows.

In this document, the term OAM will be used according to its definition specified in [RFC6291]. DetNet expects to implement an OAM framework to maintain a real-time view of the network infrastructure, and its ability to respect the Service Level Objectives (SLO), such as packet delay, delay variation, and packet loss ratio, assigned to each data flow.

2.1. Terminology

The following terms are used througout this document as defined below:

- OAM entity: a data flow to be monitored for defects and/or its performance metrics measured.

- Maintenance End Point (MEP): OAM systems traversed by a data flow when entering/exiting the network. In DetNet, it corresponds with the source and destination of a data flow. OAM messages can be exchanged between two MEPs.

- Maintenance Intermediate endPoint (MIP): an OAM system along the flow; a MIP MAY respond to an OAM message generated by the MEP.

- Control and management plane: the control and management planes are used to configure and control the network (long-term).

Relative to a data flow, the control and/or management plane can be out-of-band.

- o Active measurement methods (as defined in [RFC7799]) modify a normal data flow by inserting novel fields, injecting specially constructed test packets [RFC2544]). It is critical for the quality of information obtained using an active method that generated test packets are in-band with the monitored data flow. In other words, a test packet is required to cross the same network nodes and links and receive the same Quality of Service (QoS) treatment as a data packet.

- o Passive measurement methods [RFC7799] infer information by observing unmodified existing flows.

- o Hybrid measurement methods [RFC7799] is the combination of elements of both active and passive measurement methods.

2.2. Acronyms

OAM: Operations, Administration, and Maintenance

DetNet: Deterministic Networking

SLO: Service Level Objective

QoS: Quality of Service

SNMP: Simple Network Management Protocol

SDN: Software Defined Network

<TODO> we need here an exhaustive list, to be completed after the document has evolved.

2.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Role of OAM in DetNet

DetNet networks expect to provide communications with predictable low packet delay and packet loss. Most critical applications will define an SLO to be required for the data flows it generates.

To respect strict guarantees, DetNet can use an orchestrator able to monitor and maintain the network. Typically, a Software-Defined Network (SDN) controller places DetNet flows in the deployed network based on their the SLO. Thus, resources have to be provisioned a priori for the regular operation of the network. OAM represents the essential elements of the network operation and necessary for OAM resources that need to be accounted for to maintain the network operational.

Fault-tolerance also assumes that multiple paths could be provisioned so that an end-to-end circuit is maintained by adapting to the existing conditions. The central controller/orchestrator typically controls the Packet Replication, Elimination, and Ordering Functions (PREOF) on a node. OAM is expected to support monitoring and troubleshooting PREOF on a particular node and within the domain.

Note that PREOF can also be controlled by a set of distributed controllers, in those scenarios where DetNet solutions involve more than one single central controller.

4. Operation

OAM features will enable DetNet with robust operation both for forwarding and routing purposes.

4.1. Information Collection

Information about the state of the network can be collected using several mechanisms. Some protocols, e.g., Simple Network Management Protocol (SNMP), send queries. Others, e.g., YANG-based data models, generate notifications based on the publish-subscribe method. In either way, information about the state of the network being collected and sent to the controller.

Also, we can characterize methods of transporting OAM information relative to the path of data. For instance, OAM information may be transported out-of-band or in-band with the data flow.

4.2. Continuity Check

Continuity check is used to monitor the continuity of a path, i.e., that there exists a way to deliver the packets between two endpoints A and B.

4.3. Connectivity Verification

In addition to the Continuity Check, DetNet solutions have to verify the connectivity. This verification considers additional constraints, i.e., the absence of misconnection.

In particular, resources have to be reserved for a given flow, so they are booked for use without being impacted by other flows. Similarly, the destination does not receive packets from different flows through its interface.

It is worth noting that the test and data packets MUST follow the same path, i.e., the connectivity verification has to be conducted in-band without impacting the data traffic. Test packets MUST share fate with the monitored data traffic without introducing congestion in normal network conditions.

4.4. Route Tracing

Ping and traceroute are two ubiquitous tools that help localize and characterize a failure in the network. They help to identify a subset of the list of routers in the route. However, to be predictable, resources are reserved per flow in DetNet. Thus, DetNet needs to define route tracing tools able to track the route for a specific flow.

DetNet with IP data plane is NOT RECOMMENDED to use multiple paths or links, i.e., Equal-Cost Multipath (ECMP) [I-D.ietf-detnet-ip]. As the result, OAM in IP ECMP environment is outside the scope of this document.

4.5. Fault Verification/detection

DetNet expects to operate fault-tolerant networks. Thus, mechanisms able to detect faults before they impact the network performance are needed.

The network has to detect when a fault occurred, i.e., the network has deviated from its expected behavior. While the network must report an alarm, the cause may not be identified precisely. For instance, the end-to-end reliability has decreased significantly, or a buffer overflow occurs.

DetNet OAM mechanisms SHOULD allow a fault detection in real time. They MAY, when possible, predict faults based on current network conditions. They MAY also identify and report the cause of the actual/predicted network failure.

4.6. Fault Isolation/identification

The network has isolated and identified the cause of the fault. For instance, the replication process behaves not as expected to a specific intermediary router.

5. Administration

The network SHOULD expose a collection of metrics to support an operator making proper decisions, including:

- Queuing Delay: the time elapsed between a packet enqueued and its transmission to the next hop.

- Buffer occupancy: the number of packets present in the buffer, for each of the existing flows.

The following metrics SHOULD be collected:

- per virtual circuit to measure the end-to-end performance for a given flow. Each of the paths has to be isolated in multipath routing strategies.

- per path to detect misbehaving path when multiple paths are applied.

- per device to detect misbehaving node, when it relays the packets of several flows.

5.1. Collection of metrics

DetNet OAM SHOULD optimize the number of statistics / measurements to collected, frequency of collecting. Distributed and centralized mechanisms MAY be used in combination. Periodic and event-triggered collection information characterizing the state of a network MAY be used.

5.2. Worst-case metrics

DetNet aims to enable real-time communications on top of a heterogeneous multi-hop architecture. To make correct decisions, the controller needs to know the distribution of packet losses/delays for each flow, and each hop of the paths. In other words, the average end-to-end statistics are not enough. The collected information must be sufficient to allow the controller to predict the worst-case.

6. Maintenance

DetNet needs to implement a self-healing and self-optimization approach. The controller MUST be able to continuously retrieve the state of the network, to evaluate conditions and trends about the relevance of a reconfiguration, quantifying:

> the cost of the sub-optimality: resources may not be used optimally (e.g., a better path exists).
>
> the reconfiguration cost: the controller needs to trigger some reconfigurations. For this transient period, resources may be twice reserved, and control packets have to be transmitted.

Thus, reconfiguration may only be triggered if the gain is significant.

6.1. Replication / Elimination

When multiple paths are reserved between two maintenance endpoints, packet replication may be used to introduce redundancy and alleviate transmission errors and collisions. For instance, in Figure 1, the source node S is transmitting the packet to both parents, nodes A and B. Each maintenance endpoint will decide to trigger the packet replication, elimination or the ordering process when a set of metrics passes a threshold value.

```
                          ===> (A) => (C) => (E) ===
                        //        \\//   \\//       \\
              source (S)          //\\   //\\         (R) (root)
                        \\       //  \\ //  \\      //
                          ===> (B) => (D) => (F) ===
```

Figure 1: Packet Replication: S transmits twice the same data packet, to DP(A) and AP (B).

6.2. Resource Reservation

Because the QoS criteria associated with a path may degrade, the network has to provision additional resources along the path. We need to provide mechanisms to patch the network configuration.

6.3. Soft transition after reconfiguration

Since DetNet expects to support real-time flows, DetNet OAM MUST support soft-reconfiguration, where the novel resources are reserved before the ancient ones are released. Some mechanisms have to be proposed so that packets are forwarded through the novel track only when the resources are ready to be used, while maintaining the global state consistent (no packet reordering, duplication, etc.)

7. IANA Considerations

This document has no actionable requirements for IANA. This section can be removed before the publication.

8. Security Considerations

This section will be expanded in future versions of the draft.

9. Acknowledgments

TBD

10. Informative References

[I-D.ietf-detnet-ip]
Varga, B., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "DetNet Data Plane: IP", draft-ietf-detnet-ip-07 (work in progress), July 2020.

[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>.

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <https://www.rfc-editor.org/info/rfc2544>.

[RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, DOI 10.17487/RFC6291, June 2011, <https://www.rfc-editor.org/info/rfc6291>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <https://www.rfc-editor.org/info/rfc7276>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

Authors' Addresses

Fabrice Theoleyre
CNRS
300 boulevard Sebastien Brant - CS 10413
Illkirch - Strasbourg 67400
FRANCE

Phone: +33 368 85 45 33
Email: theoleyre@unistra.fr
URI: http://www.theoleyre.eu

Georgios Z. Papadopoulos
IMT Atlantique
Office B00 - 102A
2 Rue de la Chataigneraie
Cesson-Sevigne - Rennes 35510
FRANCE

Phone: +33 299 12 70 04
Email: georgios.papadopoulos@imt-atlantique.fr

Grek Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

DetNet Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: April 25, 2021

D. Trossen
Huawei
F.-J. Goetz
J. Schmitt
Siemens

October 23, 2020

DetNet Control Plane Signaling
draft-trossen-detnet-control-signaling-00.txt

Abstract

This document provides solutions for control plane signaling, in accordance with the control plane framework developed in the DetNet WG. The solutions cover distributed, centralized, and hybrid signaling scenarios in the TSN and SDN domain. We propose changes to RSVP IntServ for a better integration with Layer 2 technologies for resource reservation, outlining example API specifications for the realization of the revised RSVP (called RSVP-detnet in the document).

Status of this Memo

Copyright and License Notice

Table of Contents

1 Introduction

The authors in [ID.malis-detnet-controller-plane-framework-03] provide an overview of the DetNet control plane architecture along three possible classes, namely (i) fully distributed control plane utilizing dynamic signaling protocols, (ii) a centralized, SDN-like, control plane, and (iii) a hybrid control plane. We structure the following sections of this draft along those three classes in order to present example solutions for each class of the DetNet control plane architecture. Specifically, Section 2 will present a solution for a Bridged Ethernet deployment scenario. We will introduce changes to RSVP with the proposal for an RSVP-DetNet model that splits resource style and sender selection between sender and receiver, unlike in RSVP for IntServer, for an optimized realization of L2 integrations.

Section 3 will present a solution that realizes a centralized SDN-based approach for switched Ethernet deployment scenarios.

Section 4 will finally outline a hybrid solution in an SDN domain with path allocation through signaling and switching configuration as a centralized solution.

At this stage, Section 3 and 4 will be detailed in future updates to the draft.

1.1 Terminology

This document uses the terminology established in the DetNet Architecture [RFC8655], and the reader is assumed to be familiar with that document and its terminology.

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].

2 Distributed Control in Bridged TSN-based Ethernet Deployment

The first solution addresses deployments of bridged TSN-based customer Ethernet network, possibly interconnected through DetNet IP flows for multi-site deployment.

2.1 Overview

Figure 1 provides an example deployment of TSN-based Ethernet edge (customer) networks, using the RSVP/RAP combined signaling presented in the following sections. The customer sites are interconnected via edge routers that aggregate DetNet IP flow requirements from hosts for reservation of aggregated flows within the core network.

Starting point from an DetNet perspective is the RSVP IntServ model with guaranteed QoS. Resource Allocation Protocol (RAP), as defined in [RAP_IEEE], is an example of a target lower layer reservation mechanism. In this section, we focus on the necessary integration of the RSVP and RAP concepts to enable such (and similar) deployment.

```
                                 +-------+
                                +-------+
   +-----+ +-------+           +-------+           +-------+ +-----+
   |RSVP/| |DetNet |  +------+ | DetNet| +------+  |DetNet | |RSVP/|
   |RAP  |-|IP Flow|  |Edge  | |IP Flow| |Edge  |  |IP Flow|-|RAP  |
   |Node | | Edge  |--|Router|-| Core  |-|Router|--| Edge  | |Node |
   +-----+ |Network|  +------+ |Network| +------+  |Network| +-----+
           +-------+           +-------+           +-------+
```

Figure 1 : IP + Bridged Ethernet Within Customer Networks

2.2 RAP Reservation in TSN vs RSVP IntServ Model

Layer2 reservation in TSN-based networks is supported through RAP, providing a maximum of 8 classes of traffic where the frame priority code point (PCP) is used to select the Resource Allocation (RA) class at the ingress bridge. Streams within a single RA class are queued in a single traffic class where the latency of the stream is guaranteed per hop and per RA class.

This model contrasts with the RSVP IntServ [RFC2212] model, which provides a flow bandwidth driven latency model with a separate transmission queue per flow, not a class-based model like in the aforementioned RAP model.

This difference in models poses a number of challenges:

1. Is RSVP IntServ (as defined in [RFC2212]) the right starting point?

2. How to efficiently map the different reservation styles of RSVP onto RAP?

3. What is the nature of the RSVP-RAP interface?

4. How is the binding between L3 signaling (RSVP IntServer) and L2 signaling (RAP) realized, e.g., mapping of Stream-ID?

The following sub-sections elaborate on the various aspects in addressing those challenges.

2.3 Interactions between L2 and L3

Figure 2 provides an overview of the interactions between L2 and L3 elements in a network deployment as an elaboration of the elements in Figure 1.

The application utilizes an application-specific QoS API (qAPI) that controls and signals the establishment of deterministic end-to-end flows via the DetNet API (dAPI) between the L3 control plane entities in the L3 end systems and routers, utilizing the dUNI, based on RSVP, at Layer 3.

The DetNet lower API (dlAPI) maps the RSVP model onto the RAP model and signals the establishment of appropriate L2 segments via the TSN API (tAPI) between the L2 control plane entities of the end systems, routers, and L2 bridges, utilizing the tUNI, based on RAP, at Layer 2.

The L2 CP entities in turn control the establishment of appropriate resources on the HW bridge (with no specification for the handling of resource reservations on the end systems).

Within the DetNet router (on the right side of Figure 2), the second L2 control plane entity bridges to the second Ethernet sub-network.

```
   +-----------+
   |Application|
   +-----------+
     | qAPI  |
    C|       |S
     | duAPI |                 User Network              DetNet Router
   +-----------+                                        +-------------+
   |   L3 CP   |<**************************************>|    L3 CP    |
   +-----------+                                        +-------------+
      |dlAPI |                                           |   |   |   |
   M&A|      |S                                         M|  S|  M|  S|
      | tAPI |          Bridge           Bridge          |   |   |   |
   +-----------+     +-----------+    +-----------+     +-----+ +-----+
   |   L2 CP   |<===>|   L2 CP   |    |   L2 CP   |<===>|L2 CP| |L2 CP|
   +-----------+     +-----------+    +-----------+     +-----+ +-----+
         |                 |                |              |       |
        C|                C|               C|             C|      C|
         |                 |                |              |       |
   +-----------+     +-----------+    +-----------+     +-------------+
   |HW Endpoint|-----| HW Bridge |----| HW Bridge |-----|  HW Router  +
   +-----------+     +-----------+    +-----------+     +-------------+
   ***** L3 Signaling Control Protocol     DetNet UNI (dUNI RSVP)
   ===== L2 Reservation Control Protocol TSN UNI (tUNI IEEE P802.1Qdd)

   HW     Hardware                 CP     Control Plane
   qAPI   QoS API                  duAPI  DetNet upper API
   dlAPI  DetNet lower API         tAPI   TSN API
   C      Controls                 S      Signals
   M&A    Maps and Aggregation
```

Figure 2   : Interaction between L2 and L3

Based on the above interactions, the main body of work is the proposed change to RSVP, dubbed RSVP-DetNet, for an efficient mapping of L3 to L2, and motivated in Section 2.5, while we will present the various API specifications in Figure 2 throughout Section 2.6, including an example mapping between dlAPI and RAP in Section 2.6.3.

Before doing so, we will outline similarities and differences in the RSVP and RAP models at Layer 3 and 2, respectively.

2.4 Similarities and Differences between RSVP and RAP

The following sub-sections will outline various aspects to be considered when designing the RSVP-RAP interface, namely the assumptions on network nodes (Section 2.4.1), the mapping of the latency model used in both models (Section 2.4.2), the dealing with latency margins (Section 2.4.3), the dealing with Jitter and non-shaping nodes (Section 2.4.4), and the mapping of resource reservation styles (Section 2.4.5).

2.4.1 Assumptions on Network Nodes

RSVP assumes three different nodes over which a reservation can be done, namely

- Shaping node, which implements the RSVP signaling and shaping on the data plane,

- None shaping node, which implements the RSVP signaling and is capable of estimating the latency caused by this node

- Legacy node, which does neither implement RSVP nor any shaping.

RAP assumes properties common to all nodes within a reservation domain:

- All nodes take part in the signaling process

- Different data plane architectures are supported albeit limited to those defined in IEEE 802.1Q.

- Bridging between different (heterogeneous) data planes is achieved through a peer-to-peer model where every upstream node is a talker for the next downstream node.

2.4.2 Mapping of Latency Model

RSVP assumes a weighted fair queuing (WFQ) at the data plane, where a listener is able to influence therefore the latency through the reserved bandwidth per flow.

RAP assumes one traffic class with given interference per common RA class, resulting in a per hop latency for all stream within a single RA class. The E2E latency is just signaled by accumulating hop latency while the allowed interference determines the amount of allowed flow per RA class. Here, the listener is unable to influence the latency but the stream requirement is signaled upstream.

2.4.3 Dealing with Latency Margins

RSVP provides the notion of slack [RFC2212] per flow, which can be consumed by the processing node in the network to enable additional reservations.

In RAP, every listener of a stream propagates its required latency upstream to the talker. Latency margins are not handled directly by RAP, while the per hop latency of an RA class is preconfigured by management. In each node, the per RA class upstream required latency of all streams can be used to locally calculate the latency margins per hop. The management system can then use this information to adjust the per hop maximum latency at runtime.

2.4.4 Dealing with Jitter and Non-Shaping Nodes

RSVP has two different parameters to propagate the maximum non-conformance to the leaky bucket model introduced through jitter and non-shaping nodes. These can be accumulated by non-shaping nodes, i.e., those which implement the RSVP protocol but are not performing shaping at the data plane.

Within RAP, there is no distinction between shaping and non-shaping nodes since all nodes adhere to the data plane architecture defined in IEEE 802.1Q. Heterogeneous data planes are possible as long as assurances to the next hop can be upheld, while RA class attributes are used to propagate data plane behavior (e.g., shaper) to the next neighbor.

2.4.5 Mapping Resource Reservation Styles

RSVP uses the notion of 'sessions', which are able to maintain different kinds of end-to-end connectivity and resource styles, namely fixed (i) filter style, (ii) shared explicit style, and (iii) wildcard filter style - see [RFC2205]. It is important to note that in RSVP, both sender selection and resource styles are controlled by the receiver; we return to this issue in our next section.

RAP supports only distinct explicit mode of reservation, while in principle supporting reservation between one talker and multiple listeners or one listener and multiple scheduled talkers. Bridged Ethernet technology is also able to support the shared resource modes.

	Resource Style		
Sender Selection	Distinct	Shared	Coordinated Shared
Explicit	supported	supported	supported
Wildcard		supported	

Figure 3   : Resource Style and Sender Selection [RFC2205]

2.5. RSVP-DetNet

In this section, we motivate the introduction of a new signaling model for RSVP in combination with sub-nets like TSN. We outline first the rationale for its introduction before outlining the proposed changes.

2.5.1. Rationale

Continuing from Section 2.4.5. , in RSVP (for IntServ), the receiver initiates resource style and sender selection through the Resv message being sent upstream, while path state being setup through the Path message from the sender to the receiver upon receiving the Resv message.

When looking into an integration with lower layer APIs, such as the TSN API, we identify key differences in WHEN these lower layer APIs decide if a reservation is possible:

1. For a new Announce downstream, each L2 node decides that if this stream was reserved at this port, would there be enough resources available to do so?

2. For a new Attachment upstream, each L2 node will lock the required resources and bandwidth exclusively for this stream. For every L2 node local non-locked Announce, the L2 node will decide the same question as in item 1 and refresh and propagate the necessary states accordingly.

It is important to note that steps 1 and 2 only work if the 'resource style' is already known by the Announce propagation.

2.5.2. Splitting Control over Resource Style and Sender Selection

In order to allow for an efficient resource reservation at the lower network level by implementing the steps 1 and 2 in Section 2.5.1. , we propose to split the control over 'resource style' and 'sender selection' in that in RSVP-DetNet the sender controls the 'resource style' and the listener controls the 'sender selection'.

2.5.3. Coordinated Shared Resource Style

Independent from the efficient realization of lower layer resource reservation, we have also identified a requirement in industrial use cases to support a large amount of deterministic connections with small data usage. In those cases, separate reservation for each connection could be inefficient.

To address this, we propose to introduce another 'resource style' called 'Coordinated Shared', which would indicate the use of scheduling (of those many deterministic connections) at L2-Listener and L3-Receiver level. A first proposal for a solution in the TSN RAP protocol was presented to the IEEE in [CHEN-IEEE]

2.6. API Specifications

The following sub-sections describe the upper and lower Interfaces, following the proposed RSVP-DetNet changes, and provides an example mapping of the RSVP lower API to RAP in a TSN setup.

2.6.1. RSVP-DetNet upper API (duAPI)

The RSVP-DetNet interface is oriented on the interface specified by RSVP-IntServ (RFC 2205). Most of the changes are due to mapping resource reservation styles (see chapter 2.4.5).

Sender

Call: Open L3 Session (oriented to the RSVP-IntServ interface)

Request parameter:

- Flow destination IP address, Protocol ID, Destination Port

Response parameter:

- L3 Session ID (local handle)

Call: Add IP Flow

Request parameter

- L3 Session ID, Sender source IP address, Source Port, DSCP
- Traffic Specification: Maximum IP packet size (per flow, <= MTU), Minimum IP packet size, Burstiness, IP packet information rate
- Select one of the Resource Style: Distinct, Shared, Coordinated Shared
- Data TTL, PATH MTU size, Loss Rate

Notes for new parameter: The DSCP is required to map IP flows according their service class to offered service classes of the lower layer.

The traffic specification is enhanced by Minimum IP packet size for optimization interference calculation.

The resource style for an IP flow is announced by the sender within the path message.

The Loss Rate is accumulated per IP Flow.

Upcall: IP Flow

- L3 Session ID
- One of the Info_type: RESV_EVENT; PATH_ERROR

Receiver

Call: Open L3 Session

Request parameter

- Flow destination IP address, Protocol ID, Destination Port

Response parameter

- L3 Session ID

Call: Attach IP Flow

Request parameter

- L3 Session ID

- Select one of the IP flow Source Selection: Wildcard, List of explicit sources with Source Port

- Maximum packet size

- Extended Traffic Specification: Maximum Expected Latency

Notes for new parameter: The Source Selection is split form the RSVP-IntServ Reservation Style but still follows the rules defined by RSVP-IntServ. The extended traffic specification Maximum Expected Latency is propagated and merged to a minimum upstream form receiver to sender.

Upcall: IP Flow

- L3 Session ID

- One of the Info_type: RESV_EVENT; PATH_ERROR

General

Call: Close L3 Session

Request parameter

- L3 Session ID

2.6.2. RSVP-DetNet lower API (dlAPI)

The RSVP-DetNet lower API shall be lower layer network technology neutral.

Sender

Call: Add Flow

Request parameter

- L3 Session ID, Interface, L3 Flow handle, Flow destination IP address, DSCP

- Traffic Specification: Maximum IP packet size, Minimum IP packet size, Burstiness, IP packet information rate

- One of the Resource Styles: Distinct, Shared, Coordinated Shared

Response parameter

- Transport Flow Identification

Notes for new parameter:

The L3 Flow handle is a local parameter to correlate IP Flows to transport flows.

The Transport Flow Identifier correlates the IP flow to the lower layer transport flow e.g. TSN Stream ID.

Upcall: Flow

Response parameter

- L3 Session ID, Transport Flow Identification

- One of the Info_type: RESV_EVENT, RES_MODIFY_EVENT

Receiver

Call: Attach Flow

Request parameter

- L3 Session ID, Interface, L3 Flow handle, Transport Flow Identification, Maximum packet size

- Extended Traffic Specification: Maximum expected latency

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

Notes for new parameter:

(see notes above)

Upcall: Flow

Response parameter

- L3 Session ID, Transport Flow Identification

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

2.6.3. Example tAPI on RAP defined by IEEE 802.1Q

The following section defines a preliminary interface for RAP (IEEE P802.1Qdd). Currently RAP draft version 0.3 is available and the service interface of RAP is not yet stable.

Call: Open L2 Reservation Group

Request parameter

- Stream destination MAC address (unicast or multicast

- VLAN

- PCP

Response parameter

Low-Layer Group ID (local handle)

Notes:

RAP identifies its session by destination MAC address, VLAN and PCP.

Call: Close L2 Reservation Group

Request parameter

- Low-Layer Group ID

Talker

Call: Add Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

- Traffic Specification Template

- Resource Style (Distinct, Shared, Coordinated Shared)

- End-System/End-Station source MAC addres

- End-System/End-Station destination MAC address (extension for Coordinated Shared)

- Sync Domain ID (extension for Coordinated Shared)

- PATH Max frame size

- Talker Loss Rate

Notes:

Traffic Specification Template is queue drain algorithm dependent

For efficient mapping it has advantages when RAP supports all Resource Styles

The Resource Style "Coordinated Shared" allows only one destination and all nodes must belong to the same sync domain

PATH MTU frame size is determined downstream from Talker to Listener

The Loss Rate is accumulated per Stream.

Call: Modify Stream

Request parameter

- Low-Layer Group ID

- Stream Identification,

- Traffic Specification Template

Call: Release Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

Upcall: Stream

Response parameter

- Low-Layer Group ID, Stream Identification

- One of the Info_type: RESV_EVENT, RESV_MODIFY_EVENT

Listener

Call: Attach Stream

Request parameter

- Low-Layer Group ID,

- Stream Identification

- Maximum frame size

- Schedule Specification (extension for Coordinated Shared)

- Extended Traffic Specification: Maximum expected latency

Notes:

Schedule Specification includes the schedule for the group of Steams which are coordinated by synchronization

Maximum frame size will be delivered upstream from Listener to Talker

Call: Attach Modify Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

- Schedule Specification (extension only for Coordinated Shared)

Call: Release Stream

Request parameter

- Low-Layer Group ID

- Stream Identification

Upcall: Stream

Response parameter

- Low-Layer Group ID

- Stream Identification

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

3. Centralized Control Signaling in SDN Domain

For future work.

4. Hybrid Control Signaling in SDN Domain

For future work.

5. Security Considerations

Editor’s note: This section needs more details.

6. IANA Considerations

N/A

7. Conclusion

This draft outlines the possible control plane signaling in deterministic networking environments for distributed, centralized and hybrid deployments. For the first, we have proposed the introduction of RSVP-detnet for a better alignment of the Layer 3 signaling with that of emerging Layer 2 solutions, together with suggested API specifications for the realization of the L3 to L2 interfaces in endpoints.

8. References

8.1 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>.

[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>.

[RFC2212] Shenker, S., Partridge, C., and Guerin, R., "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[RFC2205] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jasmin, " Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

8.2 Informative References

[ID.malis-detnet-controller-plane-framework-04] A. Malis, X. Geng, M. Chen, F. Qin, B. Varga, "Deterministic Networking (DetNet) Controller Plane Framework", draft-malis-detnet-controller-plane-framework-04 (work in progress), 2020.

[CHEN-IEEE] F. Chen, F.J. Goetz, M. Kiessling, J. Schmitt, " Support for uStream Aggregation in RAP (ver 0.3)" (work in progess), Jan 2019, <http://www.ieee802.org/1/files/public/docs2019/dd-chen-flow-aggregation-0119-v03.pdf>

[RAP_IEEE] IEEE, "P802.1Qdd - Resource Allocation Protocol", (work in progress), <https://1.ieee802.org/tsn/802-1qdd/>

Authors' Addresses

Dirk Trossen
Huawei Technologies Duesseldorf GmbH
Riesstr. 25C
80992 Munich
Germany

Email: Dirk.Trossen@Huawei.com

Franz-Josef Goetz
Siemens AG
Gleiwitzer-Str. 555
90475 Nuremberg
Germany

Email: franz-josef.goetz@siemens.com

Juergen Schmitt
Siemens AG
Gleiwitzer Str. 555
90475 Nuremberg
Germany

Email: juergen.jues.schmitt@siemens.com