Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

B. Liu
J. Dang, Ed.
Huawei

February 22, 2021

A Queuing Mechanism with Multiple Cyclic Buffers
draft-dang-queuing-with-multiple-cyclic-buffers-00

Abstract

This document presents a queuing mechanism with multiple cyclic buffers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2021.

Table of Contents

1. Introduction 2
2. Requirements Language 2
3. Terminology & Abbreviations 2
4. Problem Statement 2
5. Queuing with Multiple Cyclic Buffers 3
6. Security Considerations 5
7. IANA Considerations 5
8. Acknowledgements 5
9. Normative References 5
Authors' Addresses 5

1. Introduction

Network forwarding with bounded latency and zero congestion loss is important for various industrial applications. The DetNet working group draft "DetNet Bounded Latency"[draft-ietf-detnet-bounded-latency] describes requirements for queuing mechanisms. Among the referenced queuing mechanisms, Cyclic Queuing and Forwarding (CQF) requires no per-flow dynamic state at core nodes, which is scalable when the number of flows grows. To cope with long link delay, more than two cyclic buffers can be used.

This document discusses the details of the cyclic queuing mechanisms. We propose a queuing model and mechanism with multiple cyclic buffers, which can improve bandwidth utilization without sacrificing latency and jitter.

2. 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.

3. Terminology & Abbreviations

CQF: Cyclic Queuing and Forwarding

T_c: the length of a cycle

4. Problem Statement

IEEE 802.1Q[IEEE8021TSN] defines CQF. As described in "DetNet Bounded Latency"[draft-ietf-detnet-bounded-latency], CQF with two synchronized buffers works as follows. All nodes keep a same cycle starting time. In a cycle x, Node A sends all packets in a buffer to

Node B. In the same cycle x, Node B uses a buffer to accumulate all packets from A, and at the same time sends out the packets that have already been buffered in cycle x-1. In the next cycle x+1, Node B sends out all the packets that are received from Node A in cycle x. If a packet traverses h hops, the maximum latency is (h+1)T_c, and the minimum latency is (h-1)T_c. The jitter (latency variation) bound is 2T_c.

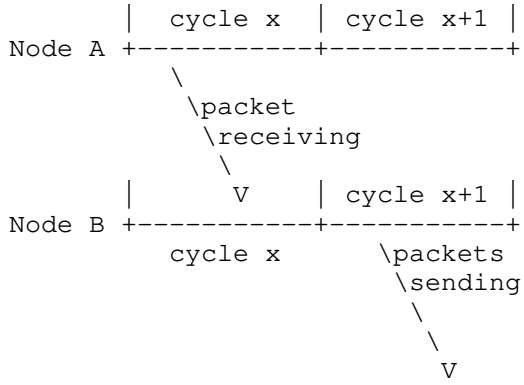


Figure-1: CQF with two synchronized buffers

The major disadvantages of CQF with two synchronized buffers are as follows. First, packets are sent and received in a same cycle by upstream and downstream nodes, respectively. So the link propagation delay must be smaller than the T_c. This prohibits the method from being used with long links, such as in WAN and MAN scenarios. Otherwise T_c must be larger than the link delay, resulting in high latency, jitter and buffer upper bound. Second, when the method is applicable, the sum of link delay, output delay, preemption delay and processing delay takes a portion of T_c, called dead time in [draft-ietf-detnet-bounded-latency], which cannot be used to send packets with deterministic services. This results in the conflict between good bandwidth utilization rate and good latency and jitter bound.

5. Queuing with Multiple Cyclic Buffers

This document proposes a cyclic queuing model that decouples link propagation delay with cycle length T_c. The model is shown in Figure-2.

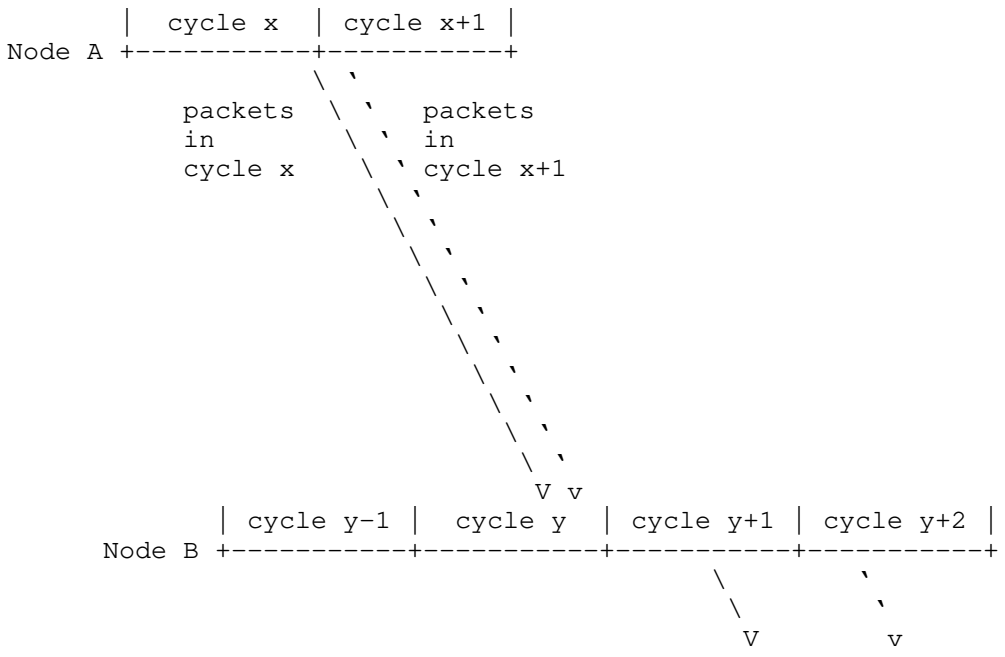


Figure-2: Proposed model for cyclic queuing

In this model, the cycle starting time points of different nodes can be either synchronized or not. However, the "phase difference" between neighbor nodes should be stable, and the variation should be bounded.

Since link delay and T_c are decoupled, a small T_c can be used with arbitrary link length without sacrificing bandwidth utilization. Any time range in a cycle can be used for deterministic traffic. In Figure-2, the last packet of the deterministic packets sent in cycle x by Node A is received by Node B in its local cycle y (all variations and jitters are considered). Node B stores those packets in a buffer associated with cycle y, and transmits the packets in this buffer in cycle y+1. The first packet of cycle x+1 may also be received by Node B in cycle y. So two or even more receiving buffers are needed simultaneously.

Since there is variation in link delay and output delay, the last packet in cycle x and the first packet in cycle x+1 may be difficult to distinguish to Node B, especially when the bandwidth utilization is high. To resolve the ambiguity, a cycle label can be put in a packet, which identifies which cycle the packet belongs with. Packets in different cycles carry different cycle labels. So Node B

can unambiguously distinguish the packet's sending cycle and map it in a correct local buffer even when the bandwidth is fully utilized.

There can be multiple ways to map a cycle label in a packet to a local cyclic buffer. For example, an ordered pair of neighboring nodes can learn a cycle mapping table. Node B may receive packets from different upstream nodes that carry different cycle labels. Node B can use the mapping table to swap the labels to a same local cycle label, and put the packets into a same local buffer. Or, an upstream node can swap a local label to a downstream label before transmitting the packet. Another way is to put a label stack in the packet, so every hop just pops a label and maps it to a local buffer. Further study is required to define how cycle labels are formated and processed and how mapping tables are learned.

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. Acknowledgements

TBD

9. Normative References

[draft-ietf-detnet-bounded-latency]
"DetNet Bounded Latency", <https://tools.ietf.org/html/draft-ietf-detnet-bounded-latency>.

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

[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>.

Authors' Addresses

Bingyang Liu
Huawei
No.156 Beiqing Road
Beijing, P.R. China 100095
China

Email: liubingyang@huawei.com

Joanna Dang (editor)
Huawei
No.156 Beiqing Road
Beijing, P.R. China 100095
China

Email: dangjuanna@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 26, 2021

Z. Du
P. Liu
China Mobile
February 22, 2021

Micro-burst Decreasing in Layer3 Network for Low-Latency Traffic
draft-du-detnet-layer3-low-latency-02

Abstract

This document introduces the problem of micro-bursts in layer3 network, and proposed 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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2021.

Table of Contents

1. Introduction 2
2. Gaps for Large-scale Layer 3 Deterministic Network 3
3. Rethinking the Problem in IP Forwarding 3
4. Method to Decrease Micro-bursts 5
4.1. Working Flow of the Method 5
4.2. Process of Edge Node 5
4.3. Process of Forwarding Node 6
5. Analysis of the Proposed Method 6
6. IANA Considerations 6
7. Security Considerations 6
8. Acknowledgements 7
9. References 7
9.1. Normative References 7
9.2. Informative References 7
Authors' Addresses 7

1. Introduction

The DetNet architecture in RFC 8655 [RFC8655] is supposed to work in campus-wide networks and private WANs, including the large-scale ISP network scenario, such as the 5G bearing network mentioned in RFC 8578 [RFC8578]. It is essential for the large-scale ISP network to be able to provide the low-latency service. The low-latency requirement exists in both L2 and L3 networks, and in both small and large networks.

However, 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 (Large-scale Deterministic Network) is introduced in [I-D.qiang-detnet-large-scale-detnet], which explores the deterministic forwarding over a large-scale network.

This document also explores the deterministic service in the large-scale layer 3 network, and proposed a method based on micro-burst decreasing, which can benefit the forwarding of low-latency traffic in a large-scale network.

2. Gaps for Large-scale Layer 3 Deterministic 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. Some of them are described as below.

Some TSN mechanisms need synchronization of the network equipments, which is easier in a small network, but hard in a large network. It brings in some complex maintenance jobs across a large distance that are not needed before.

Some TSN mechanisms need a per-flow state in the forwarding plane, which is un-scalable. Aggregation methods need to be considered.

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 main aspects of the problems are the simplicity and the scalability. The former can ensure that the mechanism is easy to deploy, and the second can ensure that the mechanism is able to bear a large number of deterministic services.

3. Rethinking the Problem in IP Forwarding

As a comparison, the current IP forwarding mechanism is considered to be a good example fulfilling the requirements of simplicity and scalability. However, traditional IP network is based on statistical multiplexing, and can only provide Best Effort service, short of SLA guaranteed mechanisms.

When we rethink the problem in the current IP forwarding mechanism, we can find that in the current IP network, a long delay in queuing, or some packet losses due to burst are acceptable; however, it is unacceptable in the deterministic forwarding. Therefore, they have different design principles in a low layer.

The current forwarding mechanism in an IP router, which is based on statistical multiplexing, 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 caused by micro-burst in the network.

Micro-burst is a special case of network congestion, which typically lasts a short period, at the granularity of millisecond. In a micro-burst, a lot of data are received on the interface suddenly, and the temporary bandwidth requirement would be tens of or hundreds of the average bandwidth requirement, or even exceed the interface bandwidth.

In most cases, the buffer on the equipment can handle the micro-bursts. However, in some corner cases, micro-bursts bring in a long delay (at the granularity of millisecond) or even packet loss.

The following paragraphs introduce the causes of the micro-burst.

Firstly, IP traffic has a instinct of burstiness no matter in the macro or micro aspect, i.e., it does not have a constant traffic model even after aggregations.

Secondly, IP network has a flexible topology, where the incoming traffic may exceed the bandwidth of the outgoing interface. For example, an interface with a large bandwidth may need to send traffic to an interface with a smaller bandwidth, and multiple flows from several incoming interfaces may need to occupy the same outgoing interface.

Thirdly, the IP node has been designed to send traffic as quickly as possible, and it is not aware whether the downstream node's buffer can handle the traffic. For example, Figure 1 below shows the problem of the current IP scheduling mechanism. Before the scheduling in an IP network, the 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 worse. However, an IP router will try to send them as soon as possible, so occasionally, in some later hops, micro-bursts will emerge.

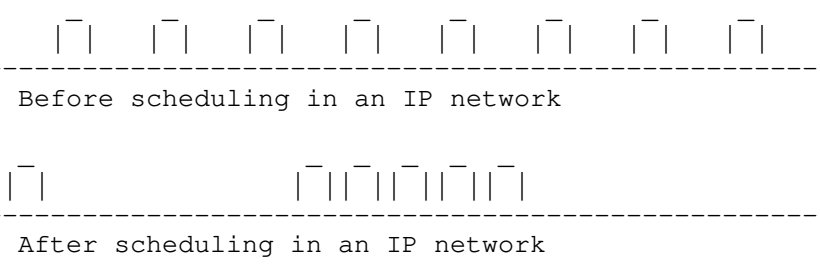


Figure 1: Change of the traffic characteristics in an IP network

This document proposes a method to support the low latency traffic bearing in an IP network, such as the 5G bearing network, by avoiding micro-bursts in the network as much as possible. The principle in this method is to forward critical and BE traffic separately, and do not distinguish different critical flows in the intermediate nodes on the forwarding plane.

4. Method to Decrease Micro-bursts

The method needs the cooperation of the edge nodes and the forwarding/core nodes in an IP network.

4.1. Working Flow of the Method

Generally, the method contains two steps:

Step1: per flow schedule in the edge node. The purpose is to make sure that each critical traffic has a constant traffic model.

Step2: per interface schedule in the core node. Traffic are aggregated to ensure the scalability, and the pacing also makes sure that they do not gather. The purpose is to make the critical traffic be forwarded as the shape when outgoing the edge, not as quickly as possible. We assume that the sending rate of the buffer for the critical traffic is the same as the receiving rate (maybe an algorithm is needed here). If all work good, the buffer will be maintained with a proper depth.

Other requirements include an RSVP liked mechanism with a good scalability, which should be used to make sure the bandwidth is not exceeded on the interface.

4.2. 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.

4.3. 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 the last section.

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 would take place. In this way, the critical traffic will have a relatively low latency in the IP network with less uncertainties of micro-bursts.

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.

5. Analysis of the Proposed Method

The method proposed does not need synchronization, just as the asynchronous mechanisms studied in IEEE 802.1 Qcr. Furthermore, the method has a larger aggregation granularity, which can fulfill the requirements of simplicity and scalability. However, it has a larger uncertainty in the forwarding than the TSN mechanisms, which needs to be further studied.

6. IANA Considerations

TBD.

7. Security Considerations

TBD.

8. Acknowledgements

TBD.

9. References

9.1. 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>.

[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>.

9.2. 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.

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2021

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

February 19, 2021

Deterministic Networking (DetNet) YANG Model
draft-ietf-detnet-yang-11

Abstract

This document contains the specification for the Deterministic Networking YANG Model for configuration and operational data for DetNet Flows. The model allows for provisioning of end-to-end DetNet service along the path without dependency on any signaling protocol. It also specifies operational status for flows.

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

Requirements Language

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 23, 2021.

Table of Contents

1. Introduction 2
2. Terminology 3
3. DetNet YANG Module 3
3.1. DetNet Application Flow YANG Attributes 3
3.2. DetNet Service Sub-layer YANG Attributes 3
3.3. DetNet Forwarding Sub-layer YANG Attributes 4
4. DetNet Flow Aggregation 4
5. DetNet YANG Structure Considerations 5
6. DetNet Configuration YANG Structures 6
7. DetNet Configuration YANG Model 15
8. Open Issues 44
9. IANA Considerations 44
10. Security Considerations 44
11. Acknowledgements 44
12. References 44
12.1. Normative References 44
12.2. Informative References 45
Appendix A. Examples 45
A.1. Example JSON Configuration/Operational 45
A.2. Example XML Config: Aggregation using a Forwarding Sublayer 50
A.3. Example JSON Service Aggregation Configuration 54
Authors' Addresses 59

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. Terminology

This document uses the terminology defined in [RFC8655].

3. DetNet YANG Module

The DetNet YANG module includes DetNet App-flow, DetNet Service Sub-layer, and DetNet Forwarding Sub-layer configuration and operational objects. The corresponding attributes used in different sub-layers are defined in Section 3.1, 3.2, 3.3 respectively.

3.1. DetNet Application Flow YANG Attributes

DetNet application flow is responsible for mapping between application flows and DetNet flows at the edge node(egress/ingress node). The 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 YANG Attributes

DetNet service functions, e.g., DetNet tunnel initialization/ termination and service protection, are provided in the 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 YANG 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:

- Flow Specification and Traffic Requirements, refers to [I-D.ietf-detnet-flow-information-model]. These may used for resource reservation, flow shaping, filtering and policing by a control plane or other network management and control mechanisms.

- Since this model programs the data plane existing explicit route mechanisms can be reused. If a static MPLS tunnel is used as the transport tunnel, the configuration need to be at every transit node along the path. For an IP based path, the static configuration is similar to the static MPLS case. This document provides data-plane configuration of IP addresses or MPLS labels but it does not provide control plane mapping mapping or other aspects.

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

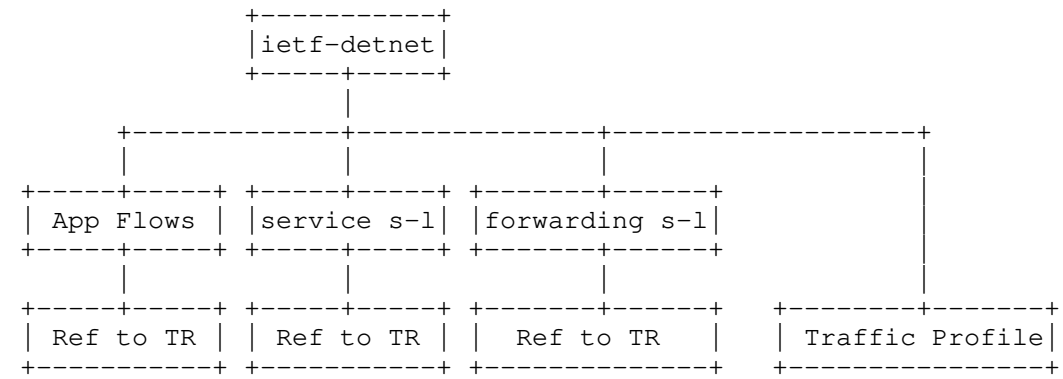
The following cases of DetNet aggregation are supported:

- Ingress node aggregates App flows into a service sub-layer of DetNet flow
- In ingress node, the service sub-layers of DetNet flows are aggregated into a forwarding sub-layer
- In ingress node, the service sub-layers of DetNet flows are aggregated into a service sub-layer of an aggregated DetNet flow
- Relay node aggregates the forwarding sub-layers DetNet flows into a forwarding sub-layer
- Relay node aggregates the service sub-layers of DetNet flows into a forwarding sub-layer
- Relay node aggregates the service sub-layers of DetNet flows into a service sub-layer of Aggregated DetNet flow
- Relay node aggregates the forwarding sub-layers of DetNet flow into a service sub-layer of Aggregated DetNet flow
- Transit node aggregates the forwarding sub-layers of DetNet flows into a forwarding sub-layer

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

5. DetNet YANG Structure Considerations

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



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

6. DetNet Configuration YANG Structures

```
module: ietf-detnet
  +--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 flow-spec
     |  |  +--rw interval?                uint32
     |  |  +--rw max-pkts-per-interval?   uint32
     |  |  +--rw max-payload-size?        uint32
     |  |  +--rw min-payload-size?        uint32
     |  |  +--rw min-pkts-per-interval?   uint32
     |  +--ro member-apps*            app-flow-ref
     |  +--ro member-services*        service-sub-layer-ref
     |  +--ro member-fwd-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 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?
|      |      |              dot1q-types:priority-type
|      |      +--:(ip-app-flow)
|      |      |  +--rw ip-app-flow
|      |      |     +--rw src-ip-prefix?          inet:ip-prefix
|      |      |     +--rw dest-ip-prefix?         inet:ip-prefix
|      |      |     +--rw protocol-next-header?   uint8
|      |      |     +--rw dscp?                   inet:dscp
|      |      |     +--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 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 interface?   if:interface-ref
```

```
      |           +--:(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* [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 app-flow-list*   app-flow-ref
|       |     +--:(service-aggregation)
|       |     |  +--rw service-aggregation
|       |     |     +--rw service-sub-layer*
|       |     |              service-sub-layer-ref
|       |     +--:(forwarding-aggregation)
|       |     |  +--rw forwarding-aggregation
|       |     |     +--rw forwarding-sub-layer*
|       |     |              forwarding-sub-layer-ref
|       |     +--:(service-id)
|       |        +--rw service-id
|       |           +--rw (detnet-flow-type)?
|       |              +--:(ip-detnet-flow)
|       |              |  +--rw src-ip-prefix?
|       |              |  |       inet:ip-prefix
|       |              |  +--rw dest-ip-prefix?
|       |              |  |       inet:ip-prefix
|       |              |  +--rw protocol-next-header?     uint8
|       |              |  +--rw dscp?                     inet:dscp
|       |              |  +--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 protocol-next-header?   uint8
      |            |        |     +--rw dscp?
      |            |        |     |       inet:dscp
      |            |        |     +--rw flow-label?
      |            |        |     |       inet:ipv6-flow-label
      |            |        |     +--rw source-port?
      |            |        |     |       inet:port-number
      |            |        |     +--rw destination-port?
      |            |        |             inet:port-number
      |            |        +--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
|           +--:(app-flow)
|           |  +--rw app-flow
|           |     +--rw app-flow-list*   app-flow-ref
|           +--:(service-disaggregation)
|           |  +--rw service-disaggregation
|           |     +--rw service-sub-layer*
|           |              service-sub-layer-ref
|           +--:(forwarding-disaggregation)
|              +--rw forwarding-disaggregation
|                 +--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
      |     |     +--rw service-sub-layer*
      |     |              service-sub-layer-ref
      |     +--:(forwarding-aggregation)
      |     |  +--rw forwarding-aggregation
      |     |     +--rw forwarding-sub-layer*
      |     |              forwarding-sub-layer-ref
      |     +--:(forwarding-id)
      |        +--rw forwarding-id
      |           +--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 protocol-next-header?     uint8
      |              |  +--rw dscp?                     inet:dscp
      |              |  +--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 protocol-next-header?
|          |      |         |       uint8
|          |      |         +--rw dscp?
|          |      |         |       inet:dscp
|          |      |         +--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* [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 protocol-next-header?
|                 |        |       uint8
|                 |        +--rw dscp?
|                 |        |       inet:dscp
```

```
|                  |        +--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-aggregation)
|  +--rw 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
+--:(forwarding-sub-layer)
|  +--rw forwarding-sub-layer
|     +--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
+--:(service-sub-layer)
|  +--rw service-sub-layer
|     +--rw service-sub-layer*
|             service-sub-layer-ref
+--:(forwarding-disaggregation)
   +--rw forwarding-disaggregation
      +--rw forwarding-sub-layer*
```

forwarding-sub-layer-ref

7. DetNet Configuration YANG Model

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

   import ietf-yang-types {
     prefix yang;
     reference
       "RFC 6021 - Common YANG Data Types.";
   }
   import ietf-inet-types {
     prefix inet;
     reference
       "RFC 6991 - Common YANG Data Types.";
   }
   import ietf-ethertypes {
     prefix ethertypes;
     reference
       "RFC 8519 - YANG Data Model for Network Access Control
                   Lists (ACLs).";
   }
   import ietf-routing-types {
     prefix rt-types;
     reference
       "RFC 8294 - Common YANG Data Types for the Routing Area.";
   }
   import ietf-packet-fields {
     prefix packet-fields;
     reference
       "RFC 8519 - YANG Data Model for Network Access Control Lists
        (ACLs).";
   }
   import ietf-interfaces {
     prefix if;
     reference
       "RFC 8343 - A YANG Data Model for Interface Management.";
   }
   import ieee802-dot1q-types {
     prefix dot1q-types;
     reference
       "IEEE 802.1Qcx-2020 - IEEE Standard for Local and Metropolitan
        Area Networks--Bridges and Bridged Networks Amendment 33: YANG
        Data Model for Connectivity Fault Management.";
```

```
   }

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

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

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

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

      Editor:   Reshad Rahman
                <mailto:reshad@yahoo.com>

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

      Editor:   Zhenqiang Li
                <mailto:lizhenqiang@chinamobile.com>";
   description
     "This YANG module describes the parameters needed
      for DetNet flow configuration and flow status
      reporting.

      Copyright (c) 2021 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject to
      the license terms contained in, the Simplified BSD License set
      forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX
      (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
      for full legal notices.

      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 (RFC 2119) (RFC 8174) when, and only when,
```

```
      they appear in all capitals, as shown here. ";

   revision 2021-02-17 {
     description
       "initial revision";
     reference
       "RFC XXXX: draft-ietf-detnet-yang-10";
   }

   identity app-status {
     description
       "Base identity from which all application-status
        status types are derived.";
     reference
       "draft-ietf-detnet-flow-information-model Section 5.8";
   }

   identity none {
     base app-status;
     description
       "This Application has no status. This type of status is
        expected when the configuration is incomplete.";
     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
      "This is an Application with 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";
    }
    description
      "This is an Application Reference.";
  }

  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";
    }
    description
      "This is a Service sub-layer Reference.";
  }

  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";
    }
    description
      "This is a Forwarding sub-layer Reference.";
  }

  typedef traffic-profile-ref {
    type leafref {
```

```
      path "/ietf-detnet:detnet"
         + "/ietf-detnet:traffic-profile"
         + "/ietf-detnet:profile-name";
    }
    description
      "This is a Traffic Profile Reference.";
  }

  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
          "This is an initiating 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.";
      }
    }
    description
      "Operation type identifies the behavior for this service
       sub-layer instance. Operations are described as unidirectional
       but a service sub-layer may combine operation types.";
  }

  typedef forwarding-operations-type {
    type enumeration {
      enum impose-and-forward {
        description
          "This operation impose outgoing label(s) and forward to
```

```
           next-hop.";
        reference
          " A YANG Data Model for MPLS Base
            draft-ietf-mpls-base-yang.";
      }
      enum pop-and-forward {
        description
          "This operation pops the incoming label and forwards to
           the next-hop.";
        reference
          " A YANG Data Model for MPLS Base
            draft-ietf-mpls-base-yang.";
      }
      enum pop-impose-and-forward {
        description
          "This operation pops the incoming label, imposes one or
           more outgoing label(s) and forwards to the next-hop.";
        reference
          " A YANG Data Model for MPLS Base
            draft-ietf-mpls-base-yang.";
      }
      enum swap-and-forward {
        description
          "This operation swaps incoming label, with an outgoing
           label and forwards to the next-hop.";
        reference
          " A YANG Data Model for MPLS Base
            draft-ietf-mpls-base-yang.";
      }
      enum forward {
        description
          "This operation forward to next-hop.";
      }
      enum pop-and-lookup {
        description
          "This operation pops incoming label and performs a
           lookup.";
      }
    }
    description
      "MPLS operations types.  This is an enum modeled after the
       MPLS enum.  The first 4 enums are the same as A YANG Data
       Model for MPLS Base.  draft-ietf-mpls-base-yang.";
  }

  typedef service-protection-type {
    type enumeration {
      enum none {
```

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

  typedef sequence-number-generation-type {
```

```
    type enumeration {
      enum copy-from-app-flow {
        description
          "This type means copy the app-flow sequence number to the
           DetNet-flow.";
      }
      enum generate-by-detnet-flow {
        description
          "This type means generate the sequence number by the
           DetNet flow.";
      }
    }
    description
      "An enumeration for the sequence number behaviors supported.";
  }

  typedef sequence-number-field {
    type enumeration {
      enum zero-sn {
        description
          "No DetNet sequence number field is used.";
      }
      enum short-sn {
        value 16;
        description
          "A 16-bit DetNet sequence number field is used.";
      }
      enum long-sn {
        value 28;
        description
          "A 28-bit DetNet sequence number field is used.";
      }
    }
    description
      "This type captures the sequence number behavior.";
  }

  grouping ip-header {
    description
      "This grouping captures the IPv4/IPv6 packet header
       information. it is modeled after existing fields.";
    leaf src-ip-address {
      type inet:ip-address;
      description
        "The source IP address in the header.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
```

```
    leaf dest-ip-address {
      type inet:ip-address;
      description
        "The destination IP address in the header.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
    leaf protocol-next-header {
      type uint8;
      description
        "Internet Protocol number.  Refers to the protocol of the
         payload.  In IPv6, this field is known as 'next-header',
         and if extension headers are present, the protocol is
         present in the 'upper-layer' header.";
      reference
        "RFC 791: Internet Protocol
         RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
    }
    leaf dscp {
      type inet:dscp;
      description
        "The traffic class value in the header.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "The flow label value of the header.IPV6 only.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
    leaf source-port {
      type inet:port-number;
      description
        "The source port number.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
    leaf destination-port {
      type inet:port-number;
      description
        "The destination port number.";
      reference
        "RFC 6021 Common YANG Data Types";
    }
  }
```

```
   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.";
       reference
         "IEEE 802.1Qcx-2020.";
     }
     leaf pcp {
       type dot1q-types:priority-type;
       description
         "The priority value of the Ethernet header.";
       reference
         "IEEE 802.1Qcx-2020.";
     }
   }

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

   grouping source-ip-port-id {
     description
       "The TCP/UDP port(source/destination) identification
```

```
        information.";
      container source-port {
        uses packet-fields:port-range-or-operator;
        description
           "This grouping captures the source port fields.";
      }
    }

    grouping ip-flow-id {
      description
        "The IPv4/IPv6 packet header identification information.";
      leaf src-ip-prefix {
        type inet:ip-prefix;
        description
          "The source IP prefix.";
        reference
          "RFC 6021 Common YANG Data Types";
      }
      leaf dest-ip-prefix {
        type inet:ip-prefix;
        description
          "The destination IP prefix.";
        reference
          "RFC 6021 Common YANG Data Types";
      }
      leaf protocol-next-header {
        type uint8;
        description
          "Internet Protocol number.  Refers to the protocol of the
           payload.  In IPv6, this field is known as 'next-header', and
           if extension headers are present, the protocol is present in
           the 'upper-layer' header.";
        reference
          "RFC 791: Internet Protocol
           RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
      }
      leaf dscp {
        type inet:dscp;
        description
          "The traffic class value in the header.";
        reference
          "RFC 6021 Common YANG Data Types";
      }
      leaf flow-label {
        type inet:ipv6-flow-label;
        description
          "The flow label value of the header.";
        reference
```

```
        "RFC 6021 Common YANG Data Types";
    }
    uses source-ip-port-id;
    uses destination-ip-port-id;
    leaf ipsec-spi {
      type ipsec-spi;
      description
        "IPsec Security Parameters Index of the Security Association.";
      reference
        "IETF RFC 6071 IP Security (IPsec) and Internet Key Exchange
         (IKE) Document Roadmap.";
    }
  }

  grouping mpls-flow-id {
    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;
          description
           "This is the case for Platform label space.";
        }
      }
    }
  }

  grouping data-flow-spec {
    description
      "app-flow identification.";
    choice data-flow-type {
      description
        "The Application flow type choices.";
      container tsn-app-flow {
        uses l2-header;
        description
          "The L2 header for application.";
      }
      container ip-app-flow {
        uses ip-flow-id;
        description
          "The IP header for application.";
```

```
      }
      container mpls-app-flow {
        uses mpls-flow-id;
        description
          "The MPLS header for application.";
      }
    }
  }

  grouping detnet-flow-spec {
    description
      "detnet-flow identification.";
    choice detnet-flow-type {
      description
        "The Detnet flow type choices.";
      case ip-detnet-flow {
        uses ip-flow-id;
      }
      case mpls-detnet-flow {
        uses mpls-flow-id;
      }
    }
  }

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

  grouping service-sub-layer-group {
    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-group {
    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 {
      description
      "The choice of DetNet 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;
          description
```

```
            "The outgoing interface, if this is a whole interface.";
          }
          choice flow-type {
            description
              "The flow type choices.";
            case ip {
              leaf next-hop-address {
                type inet:ip-address;
                description
                 "The IP next hop case.";
              }
            }
            case mpls {
              uses rt-types:mpls-label-stack;
              description
                "The MPLS Label stack next hop case.";
            }
          }
        }
        case 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
                "A user-specified identifier utilized to uniquely
                 reference the next-hop entry in the next-hop list.
                 The value of this index has no semantic meaning other
                 than for referencing the entry.";
            }
            leaf outgoing-interface {
              type if:interface-ref;
              description
                "Name of the outgoing interface.";
            }
            choice flow-type {
              description
                "The flow types supported.";
              case ip {
                leaf next-hop-address {
                  type inet:ip-address;
                  description
```

```
                    "This is the IP flow type next hop.";
                }
              }
              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;
            description
              "This is the interface as an outgoing type.";
          }
          choice flow-type {
            description
              "These are the flow type next hop choices.";
            case ip {
              choice operation-type {
                description
                  "This is the IP forwarding operation choices.";
                case ip-forwarding {
                  leaf next-hop-address {
                    type inet:ip-address;
                    description
                      "This is an IP address as a next hop.";
                  }
                }
```

```
            case mpls-over-ip-encapsulation {
              uses ip-header;
            }
          }
        }
        case mpls {
          uses rt-types:mpls-label-stack;
        }
      }
    }
    case 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 of the index for a hop.";
        }
        leaf outgoing-interface {
          type if:interface-ref;
          description
            "This is a whole interface as the next hop.";
        }
        choice flow-type {
          description
            "These are the flow type next hop choices.";
          case ip {
            choice operation-type {
              description
                "These are the next hop choices.";
              case ip-forwarding {
                leaf next-hop-address {
                  type inet:ip-address;
                  description
                    "This is an IP address as a next hop.";
                }
              }
              case mpls-over-ip-encapsulation {
                uses ip-header;
              }
            }
          }
```

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

    container detnet {
      description
        "The top level DetNet container. This contains
         applications, service sub-layers and forwarding sub-layers
         as well as the traffic profiles.";
      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.";
        }
        container traffic-requirements {
          description
            "This 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 "bps";
            description
              "This 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 "nanoseconds";
            description
              "This 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;
          units "nanoseconds";
          description
            "This 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
            "This 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
            "This 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.";
        }
      }
      container flow-spec {
        description
          "Flow-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 flow
           specification to allocate resources and adjust queue
           parameters in network nodes.";
        reference
          "draft-ietf-detnet-flow-information-model Section 5.5";
```

```
        leaf interval {
          type uint32;
          units "nanoseconds";
          description
            "The period of time in which the traffic
             specification cannot be exceeded.";
        }
        leaf max-pkts-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 min-payload-size {
          type uint32;
          description
            "The minimum payload size that the source
             will transmit.";
        }
        leaf min-pkts-per-interval {
          type uint32;
          description
            "The minimum number of packets that the
             source will transmit in one interval.";
        }
      }
      leaf-list member-apps {
        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-fwd-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;
            default false;
            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 {
          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;
            description
            "Interface is used for any service type where a whole
            interface is mapped to the applications. It may be
            further filtered by type.";
          }
          uses data-flow-spec;
        } //End of app-ingress
        container egress {
          description
            "Route's next-hop attribute.";
          leaf name {
            type string;
            description
              "Egress DetNet application.";
          }
          choice application-type {
            description
              "This is the application type choices.";
            container ethernet {
              description
              "This is TSN unaware traffic that maps to an
               interface.";
              leaf interface {
                type if:interface-ref;
                description
                  "This is an Ethernet or TSN interfaces.";
```

```
              }
            }
            container ip-mpls {
              description
                "This is IP or MPLS DetNet application types.";
              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 {
          description
            "This is the service protection type an sequence number
             options.";
          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;
          description
            "This is the service operation type for this service
             sub-layer;";
        }
        container incoming-type {
          description
            "The DetNet service sub-layer incoming configuration.";
          choice incoming-type {
            mandatory true;
            description
              "A service sub-layer may have App flows or other
               service sub-layers.";
            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-group;
            }
            container service-aggregation {
              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-group;
            }
            container forwarding-aggregation {
              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-group;
            }
            container service-id {
              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
              "The out-going type may be a forwarding Sub-layer or a
               service sub-layer or ? types need to be named.";
            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;
                  description
                    "This index allows a list of multiple outgoing
                     forwarding sub-layers";
                }
                uses detnet-header;
                uses forwarding-sub-layer-group;
              }
            }
            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 {
                description
                  "This is the MPLS service sub-layer label.";
                uses rt-types:mpls-label-stack;
              }
            }
            container 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-group;
            }
            container service-disaggregation {
              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-group;
            }
            container forwarding-disaggregation {
              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-group;
            }
          }
```

```
        }
      }
    }
    container forwarding-sub-layer {
      description
        "The DetNet forwarding sub-layer configuration.";
      list forwarding-sub-layer-list {
        key "name";
        description
          "The List is one or more DetNet Traffic types.";
        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;
          description
            "This is the forwarding operation types
             impose-and-forward, pop-and-forward,
             pop-impose-and-forward, forward, pop-and-lookup.";
        }
        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.";
              uses service-sub-layer-group;
            }
            container forwarding-aggregation {
              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-group;
          }
          container forwarding-id  {
            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
            "This is when a service connected directly to an
            interface with no forwarding sub-layer.";
          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;
          }
          container service-aggregation {
            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;
              description
                "This is reference to the service sub-layer.";
            }
            container optional-forwarding-label {
              description
```

```
                "This is the optional forwarding label for service
                 aggregation.";
              uses rt-types:mpls-label-stack;
            }
          }
          container forwarding-sub-layer {
            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;
              description
                "This is reference to the forwarding sub-layer.";
            }
            container forwarding-label {
              description
                "This is the forwarding label for forwarding
                 sub-layer aggregation.";
              uses rt-types:mpls-label-stack;
            }
          }
          container service-sub-layer {
            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-group;
          }
          container forwarding-disaggregation {
            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-group;
          }
        }
      }
    }
  }
```

```
 }
 <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 and Service Information Model", draft-ietf-detnet-flow-information-model-14 (work in progress), January 2021.

Appendix A. Examples

The following examples are provided. These examples are tested with Yanglint and use operational output to exercise both config true and config false objects

- A simple DetNet application illustrating 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 aggregation label.

A.1. Example JSON Configuration/Operational

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2020-12-18T23:59:00Z"
        }
      },
      {
        "name": "eth1",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2020-12-18T23:59:00Z"
        }
      },
      {
        "name": "eth2",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
```

```
          "statistics": {
            "discontinuity-time": "2020-12-18T23:59:00Z"
          }
        },
        {
          "name": "eth3",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2020-12-18T23:59:00Z"
          }
        },
        {
          "name": "eth4",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2020-12-18T23:59:00Z"
          }
        }
      ]
    },
    "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",
              "ip-app-flow": {
                "src-ip-prefix": "1.1.1.1/32",
                "dest-ip-prefix": "8.8.8.0/24",
                "dscp": 6
              }
            }
          },
          {
            "name": "app-1",
            "app-flow-bidir-congruent": false,
            "outgoing-service": "ssl-1",
            "traffic-profile": "pf-1",
            "ingress": {
              "app-flow-status": "ready",
              "interface": "eth0",
```

```
              "ip-app-flow": {
                "src-ip-prefix": "2.1.1.1/32",
                "dest-ip-prefix": "9.8.8.0/24",
                "dscp": 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
          },
          "flow-spec": {
            "interval": 5,
            "max-pkts-per-interval": 10,
            "max-payload-size": 1500,
            "min-payload-size": 100,
            "min-pkts-per-interval": 1
          },
          "member-apps": [
            "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
          },
          "flow-spec": {
            "interval": 5,
            "max-pkts-per-interval": 10,
            "max-payload-size": 1500,
            "min-payload-size": 100,
            "min-pkts-per-interval": 1
```

```
          },
          "member-services": [
            "ssl-1"
          ]
        },
        {
          "profile-name": "pf-3",
          "flow-spec": {
            "interval": 5,
            "max-pkts-per-interval": 10,
            "max-payload-size": 1500
          },
          "member-fwd-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": {
                "app-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
                        }
                      ]
```

```
                    },
                    "forwarding-sub-layer": [
                      "fsl-1"
                    ]
                  }
                ]
              }
            }
          }
        ]
      },
      "forwarding-sub-layer": {
        "forwarding-sub-layer-list": [
          {
            "name": "fsl-1",
            "traffic-profile": "pf-3",
            "forwarding-operation-type": "impose-and-forward",
            "incoming-type": {
              "service-sub-layer": {
                "service-sub-layer": [
                  "ssl-1"
                ]
              }
            },
            "outgoing-type": {
              "interface": {
                "outgoing-interface": "eth2",
                "mpls-label-stack": {
                  "entry": [
                    {
                      "id": 0,
                      "label": 10000
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    }
  }
```

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>2020-12-18T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth1</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-12-18T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth2</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-12-18T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth3</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-12-18T23:59:00Z</discontinuity-time>
      </statistics>
    </interface>
    <interface>
      <name>eth4</name>
      <type>ia:ethernetCsmacd</type>
      <oper-status>up</oper-status>
      <statistics>
         <discontinuity-time>2020-12-18T23:59:00Z</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>
        <app-flow-status>ready</app-flow-status>
        <interface>eth0</interface>
        <ip-app-flow>
          <src-ip-prefix>1.1.1.1/32</src-ip-prefix>
          <dest-ip-prefix>8.8.8.8/32</dest-ip-prefix>
          <dscp>6</dscp>
        </ip-app-flow>
      </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>
        <app-flow-status>ready</app-flow-status>
        <interface>eth1</interface>
        <ip-app-flow>
          <src-ip-prefix>2.1.1.1/32</src-ip-prefix>
          <dest-ip-prefix>9.8.8.8/32</dest-ip-prefix>
          <dscp>7</dscp>
        </ip-app-flow>
        <dscp>7</dscp>
      </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-apps>app-1</member-apps>
    <member-apps>app-2</member-apps>
  </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>
    <flow-spec>
      <interval>5</interval>
      <max-pkts-per-interval>10</max-pkts-per-interval>
      <max-payload-size>1500</max-payload-size>
    </flow-spec>
    <member-fwd-sublayers>afl-1</member-fwd-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>
          <app-flow-list>app-1</app-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>
            <forwarding-sub-layer>afl-1</forwarding-sub-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>
          <app-flow-list>app-2</app-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>
            <forwarding-sub-layer>afl-1</forwarding-sub-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>
          <service-sub-layer>ssl-1</service-sub-layer>
          <service-sub-layer>ssl-2</service-sub-layer>
        </service-sub-layer>
      </incoming-type>
      <outgoing-type>
        <interface>
```

```
          <outgoing-interface>eth2</outgoing-interface>
          <mpls-label-stack>
            <entry>
              <id>0</id>
              <label>10000</label>
            </entry>
          </mpls-label-stack>
         </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": {
              "app-flow-status": "ready",
              "interface": "eth0",
              "ip-app-flow": {
                "src-ip-prefix": "1.1.1.1/32",
                "dest-ip-prefix": "8.8.8.8/32",
                "dscp": 6
              }
            }
          },
          {
            "name": "app-2",
            "app-flow-bidir-congruent": false,
            "outgoing-service": "ssl-2",
            "traffic-profile": "1",
            "ingress": {
              "app-flow-status": "ready",
              "interface": "eth0",
              "ip-app-flow": {
                "src-ip-prefix": "2.1.1.1/32",
                "dest-ip-prefix": "9.8.8.8/32",
                "dscp": 7
              }
            }
          }
        ]
```

```
      },
      "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-apps": [
            "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",
          "flow-spec": {
            "interval": 5,
            "max-pkts-per-interval": 10,
            "max-payload-size": 1500
          },
          "member-fwd-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": {
                "app-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": {
                "app-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-aggregation": {
                "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
                        }
                      ]
                    },
                    "forwarding-sub-layer": [
                      "afl-1"
                    ]
                  }
                ]
              }
```

```
            }
          }
        ]
      },
      "forwarding-sub-layer": {
        "forwarding-sub-layer-list": [
          {
            "name": "afl-1",
            "traffic-profile": "3",
            "forwarding-operation-type": "impose-and-forward",
            "incoming-type": {
              "service-sub-layer": {
                "service-sub-layer": [
                  "asl-1"
                ]
              }
            },
            "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 Working Group
Internet-Draft
Intended status: Informational
Expires: August 26, 2021

Y(J) Stein
RAD

February 22, 2021

Segment Routed Time Sensitive Networking
draft-stein-srtsn-00

Abstract

Routers perform two distinct user-plane functionalities, namely forwarding (where the packet should be sent) and scheduling (when the packet should be sent). One forwarding paradigm is segment routing, in which forwarding instructions are encoded in the packet in a stack data structure, rather than programmed into the routers. Time Sensitive Networking and Deterministic Networking provide several mechanisms for scheduling under the assumption that routers are time synchronized. The most effective mechanisms for delay minimization involve per-flow resource allocation.

SRTSN is a unified approach to forwarding and scheduling that uses a single stack data structure. Each stack entry consists of a forwarding portion (e.g., IP addresses or suffixes) and a scheduling portion (deadline by which the packet must exit the router). SRTSN thus fully implements network programming for time sensitive flows, by prescribing to each router both to-where and by-when each packet should be sent.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2021.

1. Introduction

Packet Switched Networks (PSNs) use statistical multiplexing to fully exploit link data rate. On the other hand, statistical multiplexing in general leads to end-to-end propagation latencies significantly higher than the minimum physically possible, due to packets needing to reside in queues waiting for their turn to be transmitted.

Recently Time Sensitive Networking (TSN) and Deterministic Networking (DetNet) technologies have been developed to reduce this queueing latency for time sensitive packets [RFC8557]. Novel TSN mechanisms are predicated on the time synchronization of all forwarding elements (Ethernet switches, MPLS Label Switched Routers, or IP routers, to be called here simply routers). Once routers agree on time to high accuracy, it is theoretically possible to arrange for time sensitive packets to experience "green waves", that is, never to wait in queues. For example, scheduling timeslots for particular flows eliminates packet interference, but eliminates the statistical multiplexing advantage of PSNs. In addition, the scheduling calculation and programming of the network to follow this calculation doesn't scale well to large networks.

Segment Routing (SR) technologies provide a scalable method of network programming, but until now has not been applied to scheduling. The SR instructions are contained within a packet in the form of a first-in first-out stack dictating the forwarding decisions of successive routers. Segment routing may be used to choose a path sufficiently short to be capable of providing sufficiently low end-to-end latency but does not influence the queueing of individual packets in each router along that path.

2. Forwarding and Scheduling

Routers (recall that by routers we mean any packet forwarding device) perform two distinct functions on incoming packets, namely forwarding and scheduling. By forwarding we mean obtaining the incoming packet, inspecting the packet's headers, deciding on an output port, and for QoS routing a specific output queue belonging to this output port, based on the header information and a forwarding information base, optionally editing the packet (e.g., decrementing the TTL field or performing a stack operation on a MPLS label), and placing the packet into the selected output queue.

Scheduling consists of selecting which output queue and which packet from that output queue will be the next packet to be physically transmitted over the output port. In simple terms one can think of forwarding and scheduling as "which output port" and "which packet" decisions, respectively; that is, forwarding decides to which output port to send each packet, and scheduling decides which packet to send next.

Segment routing (as well as connection-oriented mechanisms) slightly simplify the meaning of forwarding to deciding "where" to send the incoming packet, while TSN slightly simplifies the meaning of scheduling to deciding "when" to send the outgoing packet.

Routers optionally perform a third user plane operation, namely per output port and/or per flow traffic conditioning. By conditioning we mean policing (discarding packets based on a token bucket algorithm), shaping (delaying packets), (W)RED, etc. Since we will only be interested in per-packet per router behavior we will neglect conditioning, which is either per router (not distinguishing between packets) or per flow (the same for all routers along the path).

As aforementioned, forwarding decisions always select an output port, but when there are QoS criteria additionally decide on an output queue belonging to that port. The use of multiple queues per output port is to aid the scheduling, which then becomes a matter of selecting an output queue and always taking the packet at the end of the queue (the packet that has waited the longest). For example, the simplest nontrivial scheduling algorithm is "strict priority". In strict priority packets are assigned to queues according to their priority (as indicated by Priority Code Point or DiffServ Code Point field). The strict priority scheduler always first checks the queue with highest priority; if there is a packet waiting there it is selected for transmission, if not the next highest priority queue is examined and so on. Undesirably strict priority may never reach packets in low priority queues (Best Effort packets), so alternative

algorithms, e.g., Weighted Fair Queueing, are used to select from priority queues more fairly.

TSN is required for networks transporting time sensitive traffic, that is, packets that are required to be delivered to their final destination by a given time. In the following we will call the time a packet is sent by the end user application (or the time it enters a specific network) the "birth time", the required delivery time to the end-user application (or the time it exists a specific network) the "final deadline" and the difference between these two times (i.e., the maximally allowed end-to-end propagation time though the network) the "delay budget".

Unlike strict priority or WFQ algorithms, TSN scheduling algorithms may directly utilize the current time of day. For example, in the TSN scheduling algorithm known as time-aware scheduling (gating), each output queue is controlled by a timed gate. At every time only certain output queues have their gates "open" and can have their packets scheduled, while packets are not scheduled from queues with "closed" gates. By appropriately timing the opening and closing of gates of all routers throughout the network, packets in time sensitive flows may be able to traverse their end-to-end path without ever needlessly waiting in output queues. In fact, time-aware gating may be able to provide a guaranteed upper bound for end-to-end delay.

However, time-aware scheduling suffers from two major disadvantages. First, opening the gates of only certain queues for a given time duration, results in this time duration being reserved even if there are very few or even no packets in the corresponding queues. This is precisely the undesirable characteristic of Time Division Multiplexing networks that led to their replacement by Packet Switched Networks. Minimizing time durations increases efficiency, but at the cost of obliging a time sensitive packet that just missed its gate to wait until the next gate opening, endangering its conforming to the delay budget.

In order to avoid such problems, one needs to know a priori the birth times of all time sensitive packets, the lengths of all links between routers, and the loading of all routers. Based on this input one can calculate optimal gating schedules for all routers in the network and distribute this information to all the routers. This calculation is computationally expensive and updating all the routers is communicationally expensive. Moreover, admitting a new time-sensitive flow requires recalculation of all the gating schedules and updating all the routers. This recalculation and communications load is practical only for small networks and a relatively small numbers of flows.

3. Stack-based Methods for Latency Control

One can envision mechanisms for reducing end-to-end propagation latency in a network with time-synchronized routers that do not suffer from the disadvantages of time sensitive scheduling. One such mechanism would be to insert the packet's birth time (time created by end-user application or time entering the network) into the packet's headers. Each router along the way could use this birth time by prioritizing packets with earlier birth times, a policy known as Longest in System (LIS). These times are directly comparable, due to our assuming the synchronization of all routers in the network. This mechanism may indeed lower the propagation delay, but at each router the decision is sub-optimal since a packet that has been in the network longer but that has a longer application delay budget will be sent before a later packet with a tighter delay budget.

An improved mechanism would insert into the packet headers the desired final deadline, i.e., the birth time plus the delay budget. Each router along the way could use this final destination time by prioritizing packets with earlier deadlines, a policy known as Earliest Deadline First (EDF). This mechanism may indeed lower the propagation delay, but at each router the decision is sub-optimal since a packet that has been in the network longer but is close to its destination will be transmitted before a later packet which still has a long way to travel.

A better solution to the problem involves precalculating individual "local" deadlines for each router, and each router prioritizing packets according to its own local deadline. As an example, a packet sent at time 10:11:12.000 with delay budget of 32 milliseconds (i.e., final deadline time of 10:11:12.032) and that needs to traverse three routers might have in its packet headers three local deadlines, 10:11:12:010, 10:11:12.020, and 10:11:12.030. The first router employs EDF using the first local deadline, the second router similarly using the second local deadline, and the ultimate router using the last local deadline.

The most efficient data structure for inserting local deadlines into the headers is a "stack", similar to that used in Segment Routing to carry forwarding instructions. The number of deadline values in the stack equals the number of routers the packet needs to traverse in the network, and each deadline value corresponds to a specific router. The Top-of-Stack (ToS) corresponds to the first router's deadline while the Bottom-of-Stack (BoS) refers to the last's. All local deadlines in the stack are later or equal to the current time (upon which all routers agree), and times closer to the ToS are always earlier or equal to times closer to the BoS.

The stack may be dynamic (as is the forwarding instruction stack in SR-MPLS) or static with an index (as is the forwarding instruction stack in SRv6).

For private networks it is possible for the stack to be inserted by the user equipment that is the source of the packet, in which case the top of stack local deadline corresponds to the first router to be encountered by the packet. However, in such a case the user equipment must also by time synchronized for its time values to be directly compatible. In an improved strategy the stack is inserted into the packet by the ingress router, and thus its deadlines are in concert with time in the network. In such case the first deadline will not explicitly appear in the stack and the initial ToS corresponds to the second router in the network to be traversed by the packet. In either case each router in turn pops from the stack the ToS local deadline and uses that local deadline in its scheduling (e.g., employing EDF).

Since the ingress router inserts the deadline stack into the packet headers, no other router needs to be aware of the requirements of the time sensitive flows. Hence admitting a new flow only requires updating the information base of the ingress router. In an efficient implementation the ingress router's information base has deadline offset vectors for each time sensitive flow. Upon receipt of a packet from user equipment, the ingress router first determines if the packet belongs to a time sensitive flow. If so, it adds the current time to the deadline offset vector belonging to the flow and inserts it as a stack into the packet headers.

An explicit example is depicted in Figure 1. Here packets of a specific time sensitive flow are required to be received by the remote user equipment within 200 microseconds of being transmitted by the source user equipment. The packets traverse a wireless link with delay 2 microseconds to reach the router R1 (the ingress router). They then travel to router R2 over an optical fiber experiencing a propagation delay of 18 microseconds, from there to router R3 experiencing an additional 38 microseconds of fiber delay, from there to router R4 (the egress router) experiencing 16 microseconds of fiber delay. Finally, they travel over a final wireless link taking again 2 microseconds.

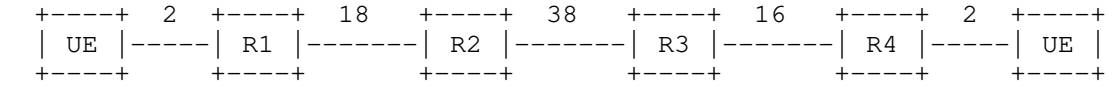


Figure 1: Example with propagation latencies

We conclude that the total constant physical propagation time is 2+18+38+16+2=76 microseconds. Moreover, assume that we know that in each router there is an additional constant time of 1 microsecond to receive the packet at the line rate and 5 microseconds to process the packet, that is, 6 microseconds per router or 24 microseconds for all four routers. We have thus reached the conclusion that the minimal time to traverse the network is 76+24=100 microseconds

Since our delay budget is 200 microseconds, we have spare time of 200-100=100 microseconds for the packets to wait in output queues. If we have no further information, we can divide this spare 100 microseconds equally among the 4 routers, i.e., 25 microseconds per router. Thus, the packet arrives at the first router after 2 microseconds, is received and processed after 2+6=8 microseconds, and is assigned a local deadline to exit the first router of 8+25=33 microseconds. The worst case times of arrival and transmission at each point along the path are depicted in Figure 2. Note that in general it may be optimal to divide the spare time in unequal fashion.

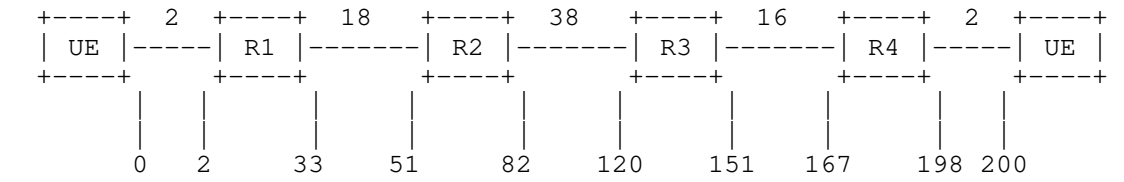


Figure 2: Example with worst case times

Assuming that the packet left router 1 the full 33 microseconds after its transmission, it will arrive at router 2 after an additional 18 microseconds, that is, after 51 microseconds. After the mandatory 6 microseconds of reception and processing and the 25 microseconds allocated for queueing, we reach the local deadline to exit router 2 by 82 microseconds. Similarly, the local deadline to exit router 3 is 151 microseconds, and the deadline to exit router 4 is 198 microseconds. After the final 2 microseconds consumed by the wireless link the packet will arrive at its destination after 200 microseconds as required

Based on these worst case times the ingress router can now build the deadline offset vector (33, 82, 151, 198) referenced to the time the packet left the source user equipment, or referenced to the time the packet arrives at the ingress router of (31, 80, 149, 196).

Now assume that a packet was transmitted at time T and hence arrives at the ingress router at time T + 2 microseconds. The ingress router R1, observing the deadline offset vector referenced to this time,

knows that the packet must be released no more than 31 microseconds later, i.e., by T + 33 microseconds. It furthermore inserts a local deadline stack [T+82, T+151, T+198] into the packet headers.

The second router R2 receives the packet with the local deadline stack, pops the ToS revealing that it must ensure that the packet exits by T + 82 microseconds. It properly prioritizes and sends the packet with the new stack [T+151, T+198]. Router R3 pops deadline T+151, and sends the packet with local deadline stack containing a single entry [T+198]. The final router pops this final local deadline and ensures that the packet is transmitted before that time The local deadline stacks are depicted in Figure 3.

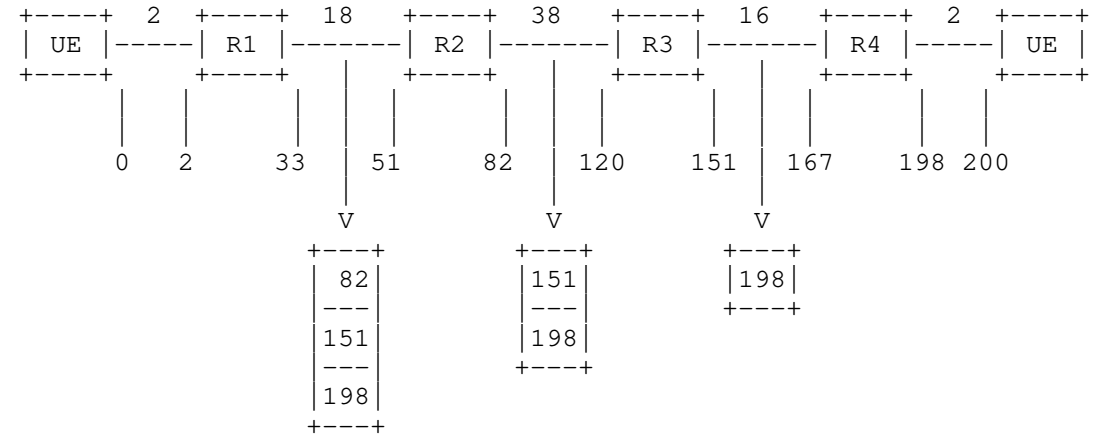


Figure 3: Example with local deadline stacks

The precise mechanism just described is by no means the only way to compute local deadlines. Furthermore, combining time-aware scheduling at the ingress router only with EDF at all the other routers can provide "green waves" with provable upper bounds to delay. However, optimizing such a scheme at scale is a challenge. A randomized algorithm for computation of the deadline offset vector is described in [AndrewsZhang].

4. The Time Sensitive Router

While a stack is the ideal data structure to hold the local deadlines in the packet, different data structures are used to hold the time sensitive packets (or their descriptors) in the routers. The standard data structure used in routers is the queue which, being a first in first out memory, is suitable for a policy of first-to-arrive first-to-exit, and not for EDF or other stack-based time sensitive mechanisms. More suitable data structures are sorted

lists, search trees, and priority heaps. While such data structures are novel in this context, efficient hardware implementations exist.

If all the time sensitive flows are of the same priority, then a single such data structure may be used for all time sensitive flows. If there are time sensitive flows of differing priorities, then a separate such data structure is required for each level of priority corresponding to a time sensitive flow, while the conventional queue data structure may be used for priority levels corresponding to flows that are not time sensitive.

For example, assume two different priorities of time sensitive flows and a lower priority for Best Effort traffic that is not time sensitive. If applying strict priority the scheduler would first check if the data structure for the highest priority contains any packets. If yes, it transmits the packet with the earliest local deadline. If not, it checks the data structure for the second priority. If it contains any packets it transmits the packet with the earliest deadline. If not, it checks the Best Effort queue. If this queue is nonempty it transmits the next packet in the queue, i.e., the packet that has waited in this queue the longest.

Separate prioritization and EDF is not necessarily the optimal strategy. An alternative (which we call Liberal EDF, or LEDF) would be for the scheduler to define a worst case (i.e., maximal) packet transmission time MAXTT (for example, the time taken for a 1500 byte packet to be transmitted at the output port's line rate). Instead of checking whether the data structure for the highest priority contains any packets at all, LEDF checks whether its earliest packet's local deadline is earlier than MAXTT from the current time. If it is, it is transmitted; if it is not the next priority is checked, knowing that even were a maximal size packet to be transmitted the scheduler will still be able to return to the higher priority packet before its local deadline.

5. Segment Routed Time Sensitive Networking

Since Segment Routing and the TSN mechanism just described both utilize stack data structures it is advantageous to combine their information into a single unified SRTSN stack. Each entry in this stack contain two subentries, the forwarding instruction (e.g., the address of the next router or the label specifying the next link) and a scheduling instruction (the local deadline).

Each SRTSN stack entry fully prescribes the forwarding and scheduling behavior of the corresponding router, both to-where and by-when the packet should be sent. The insertion of a stack into packets thus fully implements network programming for time sensitive flows.

For example, Figure 4 depicts the previous example but with the unified SRTSN stacks. Ingress router R1 inserts a SRTSN stack with three entries into the packet received. In this example the forwarding sub-entry contains the identifier or address of the next router, except for the Bottom of Stack entry that contains a special BoS code (e.g., identifier zero). The ToS entry thus contains the address of router R3 and the time by which the packet must exit router R2, namely T + 82 microseconds. Router R2 pops this ToS leaving a SRTSN stack with 2 entries. Router R3 pops the new ToS instructing it to forward the packet to router R4 by time T + 151 microseconds, leaving a stack with a single entry. Router R4 pops the ToS and sees that it has reached bottom of stack. It then forwards the packet according to the usual rules of the network (for example, according to the IP address in the IP header) by local deadline T + 198 microseconds.

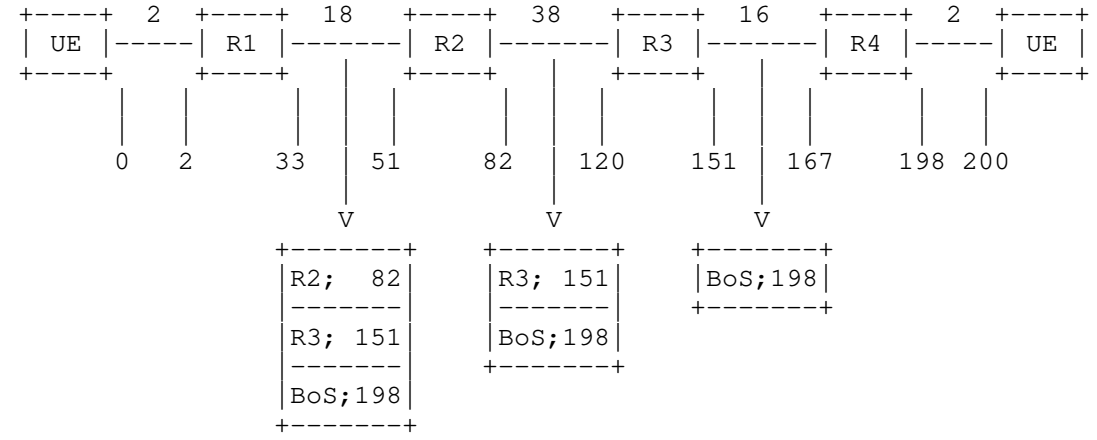


Figure 4: Example with combined SRTSN stacks

6. Stack Entry Format

A number of different time formats are in common use in networking applications and can be used to encode the local deadlines. The longest commonly utilized format is 80-bit PTP-80 timestamp defined in IEEE 1588v2 Precision Time Protocol [IEEE1588]. There are two common 64-bit time representations: the NTP-64 timestamp defined in [RFC5905] (32 bits for whole seconds and 32 bits for fractional seconds); and the PTP-64 timestamp (32 bits for whole seconds and 32 bits for nanoseconds). Finally, there is the NTP-32 timestamp (16 bits of whole seconds and 16 bits of fractional seconds) that is often insufficient due to its low resolution (15 microseconds).

However, we needn't be constrained by these common formats, since our wraparound requirements are minimal. As long as we have no ambiguity in times during the flight of a packet, which is usually much less than a second, the timestamp is acceptable. Thus, we can readily use a nonstandard 32-bit timestamp format with say 12 bits of seconds (wraparound over 1 hour) and 20 bits for microseconds, or say 8 bits for whole seconds (wraparound over 4 minutes) and 24 bits of tenths of microseconds.

For the forwarding sub-entry we could adopt like SR-MPLS standard 32-bit MPLS labels (which contain a 20-bit label and BoS bit), and thus SRTSN stack entries could be 64-bits in size comprising a 32-bit MPLS label and the aforementioned nonstandard 32-bit timestamp. Alternatively, an SR-TSN stack entry could be 96 bits in length comprising a 32-bit MPLS label and either of the standardized 64-bit timestamps.

For IPv4 networks one could employ a 32-bit IPv4 address in place of the MPLS label. Thus, using the nonstandard 32-bit timestamp the entire stack entry could be 64 bits. For dynamic stack implementations a BoS bit would have to be included.

SRv6 uses 128-bit IPv6 addresses (in addition to a 64-bit header and possibly options), and so 160-bit or 192-bit unified entries are directly derivable. However, when the routers involved are in the same network, address suffixes suffice to uniquely determine the next router.

7. Control Plane

In the above discussion we assumed that the ingress router knows the deadline offset vector for each time sensitive flow. This vector may be calculated by a centralized management system and sent to the ingress router, or may be calculated by the ingress router itself.

In the former case there is central SRTSN orchestrator, which may be based on a Network Management System, or on an SDN controller, or on a Path Computation Element server. The SRTSN orchestrator needs to be know the propagation delays for all the links in the network, which may be determined using time domain reflectometry, or via one-way delay measurement OAM, or retrieved from a network planning system. The orchestrator may additionally know basic parameters of the routers, including minimal residence time, data rate of the ports, etc. When a time sensitive path needs to be set up, the SRTSN orchestrator is given the source and destination and the delay budget. It first determines feasibility by finding the end-to-end delay of the shortest path (shortest being defined in terms of latency, not hop count). It then selects a path (usually, but not

necessarily, the shortest one) and calculates the deadline offset vector. The forwarding instructions and offset vector (as well as any other required flow-based information, such as data rate or drop precedence) are then sent to the ingress router. As in segment routing, no other router in the network needs to be informed.

In the latter case the ingress router is given the destination and the delay budget. It sends a setup message to the destination as in RSVP-TE, however, in this case arrival and departure timestamps are recorded for every router along the way. The egress router returns the router addresses and timestamps. This process may be repeated several times and minimum gating applied to approximate the link propagation times. Assuming that the path's delay does not exceed the delay budget, the path and deadline offset vector may then be determined.

The method of [AndrewsZhang] uses randomization in order to avoid the need for centralized coordination of flows entering the network at different ingress routers. However, this advantage comes at the expense of much higher achievable delay budgets.

8. Security Considerations

SRTSN concentrates the entire network programming semantics into a single stack, and thus tampering with this stack would have devastating consequences. Since each stack entry must be readable by the corresponding router, encrypting the stack would necessitate key distribution between the ingress router and every router along the path.

A simpler mechanism would be for the ingress router to sign the stack with a public key known to all routers in the network, and to append this signature to the stack. If the signature is not present or is incorrect the packet should be discarded.

9. IANA Considerations

This document requires no IANA actions.

10. Informative References

[AndrewsZhang]
Andrews, M. and L. Zhang, "Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule", Journal of Algorithms 52 57-81, 2003.

[IEEE1588] IEEE, "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE 1588-2008, DOI 10.1109/IEEESTD.2008.4579760, 2008.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010.

[RFC8557] Finn, N. and P. Thubert, "Deterministic Networking Problem Statement", RFC 8557, DOI 10.17487/RFC8557, May 2019.

Author's Address

Yaakov (J) Stein
RAD

Email: yaakov_s@rad.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: 1 October 2021

G. Mirsky
ZTE Corp.
F. Theoleyre
CNRS
G.Z. Papadopoulos
IMT Atlantique
CJ. Bernardos
UC3M
30 March 2021

Framework of Operations, Administration and Maintenance (OAM) for Deterministic Networking (DetNet) draft-tpmb-detnet-oam-framework-01

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, such as packet delay, delay variation, and packet loss ratio, assigned to each data flow.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 October 2021.

Table of Contents

1. Introduction 3
1.1. Terminology 3
1.2. Acronyms 4
1.3. Requirements Language 4
2. Role of OAM in DetNet 5
3. Operation 5
3.1. Information Collection 5
3.2. Continuity Check 6
3.3. Connectivity Verification 6
3.4. Route Tracing 6
3.5. Fault Verification/detection 6
3.6. Fault Isolation/identification 7
3.7. Use of Hybrid OAM in DetNet 7
4. Administration 7
4.1. Collection of metrics 8
4.2. Worst-case metrics 8
5. Maintenance 8
5.1. Replication / Elimination 8
5.2. Resource Reservation 9
5.3. Soft transition after reconfiguration 9
6. Requirements 9
7. IANA Considerations 10
8. Security Considerations 11
9. Acknowledgments 11
10. References 11
10.1. Normative References 11
10.2. Informative References 11
Authors' Addresses 12

1. 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.

This document lists the functional requirements toward OAM for DetNet domain. The list can further be used for gap analysis of available OAM tools to identify possible enhancements of existing or whether new OAM tools are required to support proactive and on-demand path monitoring and service validation.

1.1. 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.

* 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.

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

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

1.2. 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.

1.3. 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.

2. 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.

3. Operation

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

3.1. 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.

3.2. 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.

3.3. 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.

3.4. 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) [RFC8939]. As the result, OAM in IP ECMP environment is outside the scope of this document.

3.5. 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.

3.6. 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.

3.7. Use of Hybrid OAM in DetNet

Hybrid OAM methods are used in performance monitoring and defined in [RFC7799] as:

> Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods.

A hybrid measurement method may produce metrics as close to passive, but it still alters something in a data packet even if that is the value of a designated field in the packet encapsulation. One example of such a hybrid measurement method is the Alternate Marking method (AMM) described in [RFC8321]. One of the advantages of the use of AMM in a DetNet domain with the IP data plane is that the marking is applied to a data flow, thus ensuring that measured metrics are directly applicable to the DetNet flow.

4. 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.

4.1. 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.

4.2. 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.

5. 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.

5.1. 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.

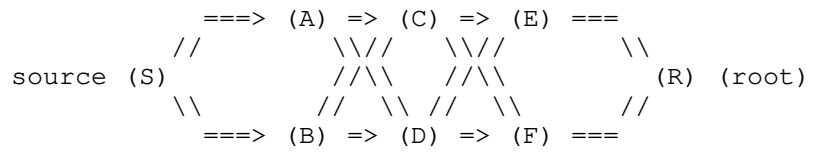


Figure 1: Packet Replication: S transmits twice the same data packet, to DP(A) and AP (B).

5.2. 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.

5.3. 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.)

6. Requirements

This section lists requirements for OAM in DetNet domain with MPLS data plane:

1. It MUST be possible to initiate DetNet OAM session from any DetNet node towards another DetNet node(s) within given domain.

2. It SHOULD be possible to initialize DetNet OAM session from a centralized controller.

3. DetNet OAM MUST support proactive and on-demand OAM monitoring and measurement methods.

4. DetNet OAM packets MUST be in-band, i.e., follow precisely the same path as DetNet data plane traffic.

5. DetNet OAM MUST support unidirectional OAM methods, continuity check, connectivity verification, and performance measurement.

6. DetNet OAM MUST support bi-directional OAM methods. Such OAM methods MAY combine in-band monitoring or measurement in the forward direction and out-of-bound notification in the reverse direction, i.e., from egress to ingress end point of the OAM test session.

7. DetNet OAM MUST support proactive monitoring of a DetNet node availability in the given DetNet domain.

8. DetNet OAM MUST support Path Maximum Transmission Unit discovery.

9. DetNet OAM MUST support Remote Defect Indication (RDI) notification to the DetNet node performing continuity checking.

10. DetNet OAM MUST support performance measurement methods.

11. DetNet OAM MAY support hybrid performance measurement methods.

12. DetNet OAM MUST support unidirectional performance measurement methods. Calculated performance metrics MUST include but are not limited to throughput, packet loss, delay and delay variation metrics. [RFC6374] provides excellent details on performance measurement and performance metrics.

13. DetNet OAM MUST support defect notification mechanism, like Alarm Indication Signal. Any DetNet node in the given DetNet domain MAY originate a defect notification addressed to any subset of nodes within the domain.

14. DetNet OAM MUST support methods to enable survivability of the DetNet domain. These recovery methods MAY use protection switching and restoration.

15. DetNet OAM MUST support the discovery of Packet Replication, Elimination, and Order preservation sub-functions locations in the domain.

16. DetNet OAM MUST support testing of Packet Replication, Elimination, and Order preservation sub-functions in the domain.

17. DetNet OAM MUST support monitoring any sub-set of paths traversed through the DetNet domain by the DetNet flow.

7. IANA Considerations

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

8. Security Considerations

This document lists the OAM requirements for a DetNet domain and does not raise any security concerns or issues in addition to ones common to networking and those specific to a DetNet discussed in [I-D.ietf-detnet-security].

9. Acknowledgments

TBD

10. References

10.1. 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>.

[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>.

10.2. Informative References

[I-D.ietf-detnet-security] Grossman, E., Mizrahi, T., and A. J. Hacker, "Deterministic Networking (DetNet) Security Considerations", Work in Progress, Internet-Draft, draft-ietf-detnet-security-16, 2 March 2021, <https://tools.ietf.org/html/draft-ietf-detnet-security-16>.

[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>.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <https://www.rfc-editor.org/info/rfc6374>.

[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>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

[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>.

[RFC8939] Varga, B., Ed., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP", RFC 8939, DOI 10.17487/RFC8939, November 2020, <https://www.rfc-editor.org/info/rfc8939>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com, gregory.mirsky@ztetx.com

Fabrice Theoleyre
CNRS
300 boulevard Sebastien Brant - CS 10413
67400 Illkirch - Strasbourg
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 Châtaigneraie
35510 Cesson-Sévigné - Rennes
France

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

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
28911 Leganes, Madrid
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: August 11, 2021

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

February 11, 2021

DetNet Control Plane Signaling
draft-trossen-detnet-control-signaling-01.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-TSN in the document).

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1. Introduction 3
1.1. Terminology 3
2. Use Cases 3
2.1. Distributed DetNet User Network Interface (ddUNI) 3
2.2. Fully Distributed Detnet Control Plane (still supports ddUNI) 4
3. Design Rationale 4
3.1. RAP Reservation in TSN vs RSVP IntServ Model 5
3.2. Similarities and Differences between RSVP and RAP 5
3.2.1. Assumptions on Network Nodes 5
3.2.2. Mapping of Latency Model 6
3.2.3. Dealing with Latency Margins 6
3.2.4. Dealing with Jitter and Non-Shaping Nodes 6
3.2.5. Mapping Resource Reservation Styles 7
3.3. Design Considerations for RSVP-TSN 7
3.3.1. Rationale 7
3.3.2. Splitting Control over Resource Style and Sender Selection 8
3.3.3. Coordinated Shared Resource Style 8
3.3.4. DnFlow DestinatinIpAddress Resolution 8
4. RSVP-TSN 9
4.1. Layer Interactions between RSVP and TSN 9
4.2. API for Deterministic QoS (gQoS) 10
4.3. RSVP-TSN upper API (uRSVP) 10
4.4. RSVP-TSN lower API (lRSVP) 12
4.5. RSVP-TSN Message Formats 14
5. Security Considerations 14
6. IANA Considerations 14
7. Conclusion 14
8. References 14
8.1. Normative References 14
8.2. Informative References 15
Authors' Addresses 15

1. Introduction

The authors in [ID.malis-detnet-controller-plane-framework] 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.

When investigating the usage of RSVP [RFC2205] for the signaling of deterministic IP connectivity in combination of underlying Layer 2 mechanisms, considerations arise for the development of the detnet-specific RSVP protocol, called RSVP-TSN in the following.

This document will outline use cases spanning the classes of control planes introduced in [ID.malis-detnet-controller-plane-framework], followed by the design rationale and specification for the proposed RSVP-TSN protocol.

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. Use Cases

Based on the detnet stack model [RFC 8938], "Resource allocation", located in the forwarding sub-layer, is split into RSVP-TSN IP flow signaling and underlying TSN subnet stream reservation. Stream reservation within TSN subnetworks can be organized with a decentralized, centralized or hybrid configuration model. The notion of TSN in these use cases and the remainder of the document assumes a Bridged-Ethernet LAN with enhancements for time-sensitive networking.

2.1. Distributed DetNet User Network Interface (ddUNI)

The following figure illustrates the principle of a hybrid DetNet using RSVP-TSN for DnFlow signaling in a TSN aware customer network. DetNet/TSN end nodes signal their DnFlows over RSVP-TSN. In parallel, the TSN control plane triggers the stream reservation within a TSN aware customer network, using e.g., LRP/RAP. The control plane solution of a TSN customer network is independent from RSVP-TSN signaling and can cover distributed, centralized or hybrid

reservation scenarios.

An RSVP detnet Edge Router supports RSVP-TSN signaling of DnFlows and covers DnFlow signaling supported by the associated detnet aware core network. Although the DetNet control plane within the DetNet core network is without support for RSVP, it still supports the DetNet Flow and Service Information Model [ID-detnet-flow-information-model] and can be organized in a decentralized or centralized (SDN-like) manner.

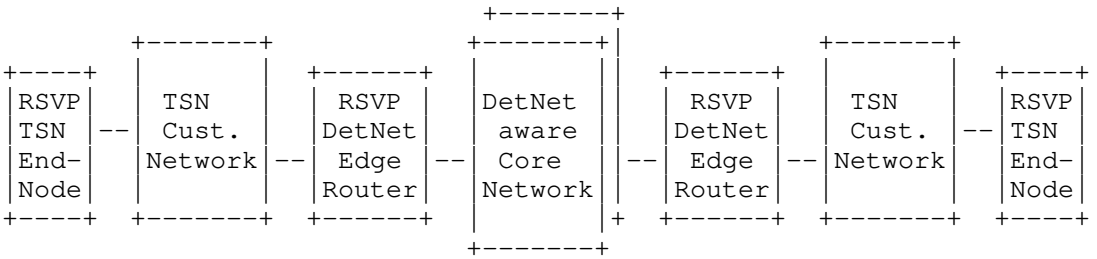


Figure 1 : Distributed DetNet UNI

2.2. Fully Distributed Detnet Control Plane (still supports ddUNI)

The following figure illustrates a fully distributed DetNet using RSVP-TSN for DnFlow signaling in TSN aware customer networks and RSVP aware core networks. In difference to the previous scenario, the detnet control plane within the detnet aware core network still supports RSVP to establish detnet end-2-end connectivity.

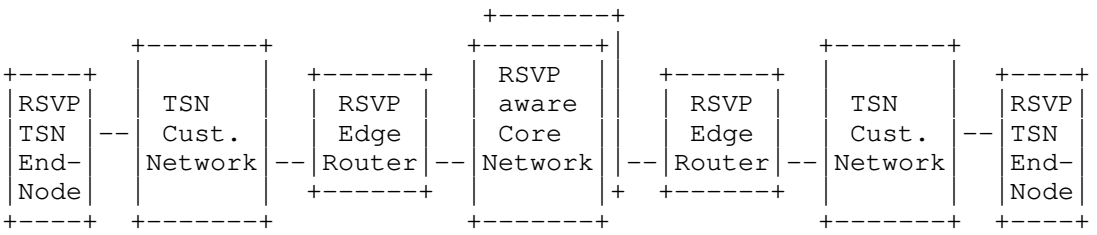


Figure 2 : Fully Distributed DetNet UNI

3. Design Rationale

This section will explore the design rationale behind the development of RSVP-TSN. The next two sub-sections outline aspects derived from the comparison of RAP, as a Layer2 mechanism, and RSVP, before highlighting key design considerations for the presentation of RSVP-TSN in Section 4.

3.1. 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-TSN onto RAP?

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

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.

3.2. Similarities and Differences between RSVP and RAP

The following sub-sections will outline various aspects to be considered when designing the interfaces between RSVP-TSN and RAP, namely the assumptions on network nodes (Section 3.2.1), the mapping of the latency model used in both models (Section 3.2.2), the dealing with latency margins (Section 3.2.3), the dealing with Jitter and non-shaping nodes (Section 3.2.4), and the mapping of resource reservation styles (Section 3.2.5).

3.2.1. 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.

3.2.2. 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.

3.2.3. 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.

3.2.4. 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.

3.2.5. 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] and Figure 3. 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, discussing the rationale for the proposed design for RSVP-TSN.

The current draft version of RAP supports only distinct explicit mode of reservation, while in principle supporting reservation between one talker and multiple listeners. Bridged Ethernet technology is also able to support the shared resource modes as specified by RSVP. Also a new resource style called Coordinated Shared Resource Style is planned.

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

Figure 3: Resource Style and Sender Selection [RFC2205]

3.3. Design Considerations for RSVP-TSN

3.3.1. Rationale

Continuing from Section 3.2.5, 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.

3.3.2. 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 3.3.1, we propose to split the control over 'resource style' and 'sender selection' in that in RSVP-TSN the sender controls the 'resource style' and the listener controls the 'sender selection'.

3.3.3. 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]

3.3.4. DnFlow DestinatinIpAddress Resolution

To support deterministic QoS Bridged Ethernet has introduced Streams. Streams differ from legacy traffic within a Bridged Ethernet sub-network because streams belong to a traffic class which is uniquely

identified by a priority value in the range of 0 through 7. Streams within an TSN aware Bridged Ethernet sub-network also need unique destination MAC-address for identification. The priority and the unique destination MAC-address indicates a Stream within a virtual LAN (VLAN). The IEEE 802.1CQ draft for "Multicast and Local Address Assignment" specifies protocols and procedures of locally unique assignment for 48-bit and 64-bit addresses in IEEE 802 networks.

Streams do not use the interface Mac-Address as destination MAC-Address within a Bridged Ethernet. Further enhancements for IP address resolutions are required within TSN and detnet aware end-systems and routers and to map one or multiple detnet IP flows to a stream destination MAC-Address. DnFlows are identified by a "6-tuple" that refers to information carried in IP and higher layer protocol headers. The 6-tuple referred to in this document is the same as that defined in [RFC3290]. Specifically, 6-tuple is DestinationIpAddress, SourceIpAddress, Protocol, SourcePort, DestinationPort, and differentiated services (DiffServ) code point (DSCP).

4. RSVP-TSN

In this section, we specify the APIs for RSVP-TSN, the message formats, as well as outline the layer and node interactions in an RSVP-TSN based system

4.1. Layer Interactions between RSVP and TSN

Figure 4 provides an overview of the interactions between L2 and L3 elements in a network deployment as an elaboration of the elements in Figure 1, also illustrating the various interfaces described in the following sections.

The application utilizes a generalized API for deterministic QoS (dQoS) that controls and signals the establishment of deterministic end-to-end DnFlow via the upper API of RSVP-TSN (uRSVP).

RSVP-TSN end nodes utilize RSVP-TSN to signal DnFlows to a detnet aware edge router. This L3 network interface is called "Distributed DetNet User Network Interface" (ddUNI).

The lower API of RSVP-TSN (lRSVP) interacts with the TSN control plane to trigger the establishment of streams in an TSN aware (e.g. customer) sub-network. The TSN control plane for the establishment of streams in a TSN sub-network can be organized decentralized, centralized or hybrid for stream reservation. For stream establishment based on centralized scheduling, a third-party protocol like RESTCONF is typically used.

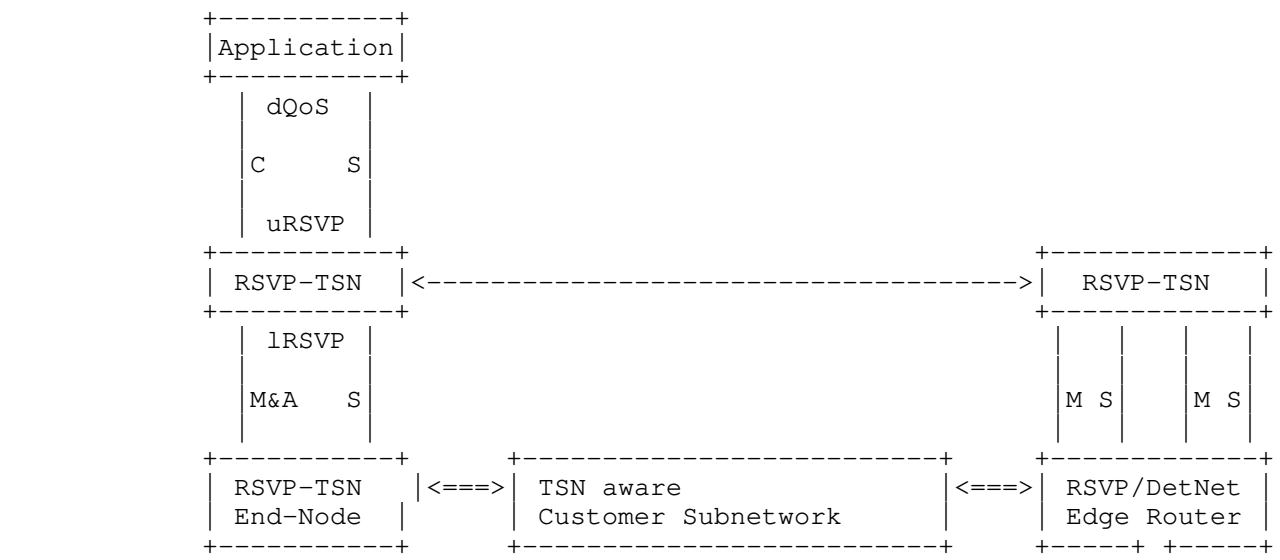


Figure 4: Layer Interactions between RSVP and TSN

4.2. API for Deterministic QoS (gQoS)

The description of a generalized API to support deterministic QoS is not part of this document.

4.3. RSVP-TSN upper API (uRSVP)

The definition of the upper and lower APIs of RSVP-TSN is based on the DetNet flow information model [ID-detnet-flow-information-model].

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

Sender

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

Request parameter (make use of pieces from the DnFlowSpecification)

- DestinationIpAddress, Protocol, DestinationPort

Response parameter:

- SessionID

Call: Add DnFlow

Request parameter (make use of pieces from the DnFlowSpecification)

- SessionID, SourceIpAddress, SourcePort, DSCP

- DnTrafficSpecification: Interval, MaxPacketsPerInterval, MaxPayloadSize, MinPayloadSize

- DnFLowRank

- Select one of the Resource Style: Distinct, Shared, CoordinatedShared

- Data TTL, PATH MTU size, LossRate

Notes for new parameter:

The DSCP is required to map DnFlows according their service class to offered service classes of the lower layer.

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

The LossRate is accumulated per DnFlow from Sender to Receiver.

Upcall: DnFlow

- Session ID

- One of the Info_type: RESV_EVENT; PATH_ERROR

Receiver

Call: Open Session

Request parameter (make use of pieces from the DnFlowSpecification)

- DestinationIpAddress, Protocol, DestinationPort

Response parameter

- SessionID

Call: Join DnFlow

Request parameter

- SessionID

- Select one of the DnFlow Source Selection: Wildcard, List of explicit sources with SourcePort

- MaximumPacketSize

- Extended Traffic Specification: MaximumExpectedLatency

Notes for new parameter:

The Source Selection is split from the RSVP-IntServ Reservation Style but still follows the rules defined by RSVP-IntServ.

The extended traffic specification MaximumExpectedLatency is propagated and merged to a minimum upstream from receiver to sender.

Upcall: DnFlow

- SessionID

- SourceIpAddress (Sender)

- SourcePort

- One of the Info_type: RESV_EVENT; PATH_ERROR

General

Call: Close Session

Request parameter

- SessionID

4.4. RSVP-TSN lower API (lRSVP)

Sender

Call: Add DnFlow

Request parameter

- SessionID, Interface, DnFlowID, DestinationIpAddress, DSCP

- DnTrafficSpecification: Interval, MaxPacketsPerInterval, MaxPayloadSize, MinPayloadSize, MinPacketsInterval

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

Response parameter

- TransportFlowID (TSN StreamID)

Notes for new parameter:

The DnFlowID is a local parameter to correlate DnFlows to transport flows (e.g., TSN Stream).

The TransportFlowID correlates the DnFlow to the lower layer transport flow, e.g., TSN Stream ID.

Upcall: DnFlow

Response parameter

- SessionID

- TransportFlowID

- One of the Info_type: RESV_EVENT, RES_MODIFY_EVENT

Receiver

Call: Join DnFlow

Request parameter

- SessionID, Interface, DnFlowID, TransportFlowID

- MaximumPacketSize

- Extended Traffic Specification: MaximumExpectedLatency

Notes for new parameter:

(see notes above)

Upcall: DnFlow

Response parameter

- SessionID, TransportFlowID

- One of the Info_type: ANNOUNCE_EVENT, ANNOUNCE_MODIFY_EVENT

4.5. RSVP-TSN Message Formats

TBD

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 this, changes to the RSVP signaling have been proposed in the form of RSVP-TSN 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.

[RFC8938] B. Varga, Ed, J. Farkas, L. Berger, A. Malis, S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC8938, November 2020.

8.2. Informative References

[ID.malis-detnet-controller-plane-framework] A. Malis, X. Geng, M. Chen, F. Qin, B. Varga, "Deterministic Networking (DetNet) Controller Plane Framework", draft-malis-detnet-controller-plane-framework-05 (work in progress), 2020.

[ID-detnet-flow-information-model] Balazs Varga, Janos Farkas, Rodney Cummings, Yuanlong Jiang, Don Fedyk, "DetNet Flow and Service Information Model", draft-ietf-detnet-flow-information-model-14 (work in progress), 2021

[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